aws

ユーザーガイド

Amazon Relational Database Service

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Relational Database Service: ユーザーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

Amazon RDS とは 1
Amazon RDS の利点 1
責任の比較 2
Amazon RDS 責任共有モデル 3
DB インスタンス 4
Amazon RDS アプリケーションアーキテクチャの例 5
DB エンジン 6
DB インスタンスクラス 7
DB インスタンスストレージ 8
Amazon Virtual Private Cloud (Amazon VPC) での DV インスタンス 9
AWS リージョン およびアベイラビリティゾーン 9
アベイラビリティーゾーン 10
マルチ AZ 配置 10
セキュリティグループによるアクセスコントロール 12
Amazon RDS のモニタリング 13
Amazon RDS のユーザーインターフェイス 14
AWS マネジメントコンソール 15
コマンドラインインターフェイス 15
Amazon RDS API 15
Amazon RDS の課金方法 16
Amazon RDS の AWS 無料利用枠 16
次のステップ 17
スタート方法 17
データベースエンジンに固有のトピック 17
DB インスタンス 18
DB インスタンスクラス 21
DB インスタンスクラスタイプ 21
サポートされる DB エンジン 31
AWS リージョン での DB インスタンスクラスのサポートを決定する 118
DB インスタンスクラスの変更 124
クロス AZ ENA Express のサポート 125
RDS for Oracle のプロセッサを設定する 126
ハードウェア仕様 152
DB インスタンスストレージ 199

ストレージタイプ 200
プロビジョンド IOPS ストレージ 201
汎用ストレージ 205
SSD のパフォーマンス特性 209
マグネティックストレージ (レガシー、非推奨) 214
追加のストレージボリューム 215
専用ログボリューム (DLV) 216
データベースのパフォーマンスのモニタリング 217
データベースのパフォーマンスに影響する要因 219
リージョン、アベイラビリティーゾーン、および Local Zones 222
AWS リージョン 223
アベイラビリティーゾーン 229
ローカルゾーン 230
リージョンのオプトアウトとオプトインの動作 232
リージョンからオプトアウトする 232
リージョンに再オプトインする 232
リージョンとエンジンでサポートされている Amazon RDS 機能 233
テーブルの規則 234
機能クイックリファレンス 234
ブルー/グリーンデプロイ 237
クロスリージョン自動バックアップ 239
クロスリージョンリードレプリカ 240
データベースアクティビティストリーミング 243
デュアルスタックモード 251
スナップショットを S3 にエクスポートする 280
IAM データベース認証 298
Kerberos 認証 305
マルチ AZ DB クラスター 325
Performance Insights 336
RDS Custom 337
Amazon RDS Proxy 360
Secrets Manager の統合 380
ゼロ ETL 統合 380
エンジンネイティブの機能 398
Amazon RDS での DB インスタンスの請求 399
オンデマンド DB インスタンス 401

リザーブド DB インスタンス 403
設定 418
AWS アカウント へのサインアップ 418
要件の確認 418
DB インスタンスへのアクセスを提供 420
スタート方法 424
MariaDB DB インスタンスの作成と接続 425
前提条件 426
EC2 インスタンスの作成 426
MariaDB DB インスタンスを作成する 432
(オプション) CloudFormation を使用して VPC、EC2 インスタンス、MariaDB インスタンスを作成する 438
MariaDB DB インスタンスに接続する 440
EC2 インスタンスと DB インスタンスを削除する 443
(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する 444
(オプション) DB インスタンスを Lambda 関数に接続する 445
Microsoft SQL Server DB インスタンスを作成して接続する 446
前提条件 447
EC2 インスタンスの作成 448
SQL Server DB インスタンスを作成する 453
(オプション) CloudFormationを使用して VPC、EC2 インスタンス、SQL Server インスタンスを作成する 459
SQL Server DB インスタンスへの接続 461
サンプル DB インスタンスを試す 463
EC2 インスタンスと DB インスタンスを削除する 465
(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する 466
(オプション) DB インスタンスを Lambda 関数に接続する 466
MySQL DB インスタンスの作成と接続 468
前提条件 469
EC2 インスタンスの作成 469
MySQL DB インスタンスを作成する 475
(オプション) CloudFormation を使用して VPC、EC2 インスタンス、MySQL インスタンスを作成する 481
MySQL DB インスタンスに接続する 483

EC2 インスタンスと DB インスタンスを削除する 486
(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する 487
(オプション) DB インスタンスを Lambda 関数に接続する 488
Oracle DB インスタンスを作成して接続する 489
前提条件 490
ステップ 1: EC2 インスタンスを作成する 491
ステップ 2: Oracle DB インスタンスを作成する 497
(オプション) CloudFormation を使用して VPC、EC2 インスタンス、Oracle DB インスタンスを作成する 502
ステップ 3: SQL クライアントを Oracle DB インスタンスに接続する 504
ステップ 4: EC2 インスタンスと DB インスタンスを削除する 508
(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する 508
(オプション) DB インスタンスを Lambda 関数に接続する 509
PostgreSQL DB インスタンスを作成して接続する 510
前提条件 511
EC2 インスタンスの作成 512
PostgreSQL DB インスタンスを作成する 518
(オプション) CloudFormation を使用して VPC、EC2 インスタンス、PostgreSQL インスタンスを作成する 523
PostgreSQL DB インスタンスへの接続 525
EC2 インスタンスと DB インスタンスを削除する 528
(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する 529
(オプション) DB インスタンスを Lambda 関数に接続する 530
チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する 531
EC2 インスタンスを起動して DB インスタンスに接続する 532
DB インスタンスを作成する 538
ウェブサーバーのインストール 559
チュートリアル: Amazon RDS DB インスタンスにアクセスするための Lambda 関数を作成する 571
前提条件 572
Amazon RDS DB インスタンスを作成する 572
Lambda 関数とプロキシを作成する 573
関数実行ロールを作成するには 575

Lambda デプロイパッケージを作成する ... 576
Lambda 関数を更新する ... 579
コンソールで Lambda 関数をテストする ... 580
Amazon SQS キュー を作成する ... 581
Lambda 関数を呼び出すためのイベントソースマッピングを作成する ... 582
セットアップのテストとモニタリング ... 583
リソースのクリーンアップ ... 584
チュートリアルとサンプルコード ... 586
このガイドのチュートリアル ... 586
他の AWS ガイドのチュートリアル ... 588
GitHub のチュートリアルとサンプルコード ... 588
AWS データベースクックブック ... 589
AWS ワークショップ - Amazon RDS PostgreSQL ... 590
AWS ワークショップ - Amazon RDS MySQL ... 590
AWS SDK の操作 ... 590
Amazon RDS のベストプラクティス ... 592
Amazon RDS の基本的な操作のガイドライン ... 592
DB インスタンスの RAM の推奨事項 ... 594
データベースエンジンのバージョンを最新の状態に保つ ... 594
AWS データベースドライバー ... 595
拡張モニタリングの使用 ... 595
メトリクスの使用 ... 595
パフォーマンスメトリクスの表示 ... 595
パフォーマンスメトリクスの評価 ... 599
クエリのチューニング ... 601
MySQL の使用のベストプラクティス ... 602
テーブルのサイズ ... 602
テーブルの数 ... 603
ストレージエンジン ... 603
MariaDB の使用のベストプラクティス ... 604
テーブルのサイズ ... 604
テーブルの数 ... 605
ストレージエンジン ... 606
Oracle を使用するためのベストプラクティス ... 606
PostgreSQL を使用するためのベストプラクティス ... 606
PostgreSQL DB インスタンスにデータをロードする ... 607

PostgreSQL Autovacuum 機能の使用 607
Amazon RDS for PostgreSQL のベストプラクティスの動画 609
SQL Server を使用するためのベストプラクティス 609
Amazon RDS for SQL Server のベストプラクティスの動画 610
DB パラメータグループを使用する 610
DB インスタンスの作成の自動化 611
新機能の動画 612
Amazon RDS へのプログラムによるアクセス 613
Console-to-Code 614
DB インスタンスの設定 615
DB インスタンスの作成 616
前提条件 616
DB インスタンスの作成 623
使用できる設定 630
での リソースの作成AWS CloudFormation 669
RDS とCloudFormationテンプレート 669
の詳細情報CloudFormation 669
DB インスタンスに接続する 670
接続情報を検索する 670
DB インスタンスにアクセスするシナリオ 674
AWS ドライバーを使用した DB インスタンスへの接続 675
特定の DB エンジンを実行する DB インスタンスに接続する 676
RDS Proxy による接続の管理 677
データベース認証オプション 677
暗号化された接続 677
オプショングループを使用する 678
オプショングループの概要 678
オプショングループを作成する 681
オプショングループをコピーする 683
オプショングループにオプションを追加する 684
オプショングループのオプションとオプション設定をリスト化する 690
オプションの設定を変更する 691
オプショングループからオプションを削除する 695
オプショングループを削除する 697
パラメータグループ 701
パラメータグループの概要 701

DB パラメータグループ 705
DB クラスターのパラメータグループ 722
DB パラメータグループを比較する 736
DB パラメータの指定 737
Amazon RDS から ElastiCache キャッシュを作成する 744
RDS DB インスタンス設定による ElastiCache キャッシュ作成の概要 744
既存の RDS DB インスタンスの設定で ElastiCache キャッシュを作成する 745
データベースの自動移行 749
概要 749
前提条件 750
制限事項 751
IAM リソースの作成 752
データ移行の設定 760
移行の管理 762
モニタリング 764
チュートリアル: カスタムパラメータと新しいオプショングループを使用した MySQL DB インスタンスの作成 766
前提条件 767
カスタムパラメータグループの作成 767
パラメータ値を変更する 768
オプショングループを作成する 768
オプションの追加 769
DB インスタンスを作成する 770
DB インスタンスの管理 772
DB インスタンスの停止 773
ユースケース 773
時間に関する考慮事項 774
サポートされている DB エンジン、クラス、リージョン 775
マルチ AZ のサポート 775
仕組み 776
制限事項 777
デフォルトのオプションとパラメータグループ 778
パブリック IP アドレス 778
DB インスタンスの停止 778
DB インスタンスのスタート 780
DB インスタンスを再起動する 782

DB インスタンスの再起動のユースケース 782
再起動の仕組み 783
マルチ AZ での再起動 783
考慮事項 784
前提条件 784
DB インスタンスの再起動: 基本的な手順 785
EC2 インスタンスの接続 787
概要 788
EC2 インスタンスの接続 793
接続中のコンピューティングリソースを表示する 796
特定の DB エンジンを実行する DB インスタンスに接続する 797
Lambda 関数を接続する 798
概要 800
Lambda 関数を接続する 811
接続中のコンピューティングリソースを表示する 813
DB インスタンスを変更する 815
スケジュールの変更 817
使用できる設定 818
DB インスタンスのメンテナンス 864
DB インスタンスのメンテナンス更新の概要 864
保留中のメンテナンスの表示 865
マルチ AZ 配置のメンテナンス 868
メンテナンスウィンドウ 869
アップデートの適用 875
オペレーティングシステムの更新 877
AWS Organizations アップグレードロールアウト 881
エンジンバージョンのアップグレード 885
エンジンバージョンの手動アップグレード 886
マイナーエンジンバージョンの自動アップグレード 888
DB インスタンスの名前を変更する 893
名前を変更して既存の DB インスタンスを置き換える 894
DB インスタンスのリードレプリカの操作 897
概要 899
DB エンジン間の違い 903
リードレプリカの作成 906
リードレプリカの昇格 910

リードレプリケーションのモニタリング 915
クロスリージョンリードレプリカ 919
RDS リソースのタグ付け 933
RDS タグを使用する理由 934
RDS タグの仕組み 935
ベストプラクティス 938
タグを DB スナップショットにコピーする 939
自動バックアップのタグ付け 940
Amazon RDS でのタグの追加と削除 941
チュートリアル: タグを使用して、停止する DB インスタンスを指定する 945
Amazon RDS の ARN 949
ARN の構築 949
既存の ARN の取得 957
ストレージの使用 960
DB インスタンスのストレージボリュームの詳細の表示 960
DB インスタンスストレージの容量を増加する 961
追加ストレージボリュームの削除 967
ストレージの自動スケーリングによる容量の自動管理 969
ストレージファイルシステムのアップグレード 978
プロビジョンド IOPS の設定を変更する 979
I/O を大量に消費するストレージの変更 981
汎用 (gp3) の設定変更 983
専用ログボリューム (DLV) を使用する 985
DB インスタンスを削除する 992
DB インスタンスの削除の前提条件 992
DB インスタンスを削除する場合の考慮事項 992
DB インスタンスを削除する 994
チュートリアル: MySQL DB インスタンスの管理 997
序章 997
前提条件 997
DB インスタンスへのタグの追加 998
DB ストレージを増やす 998
リードレプリカの作成 999
タグの更新 1000
マルチ AZ 配置の設定と管理 1003
マルチAZ DB インスタンスのデプロイ 1005

DB インスタンスをマルチ AZ 配置に変換する 1007
マルチ AZ DB インスタンスのフェイルオーバー 1009
追加のストレージボリュームによるマルチ AZ フェイルオーバー 1013
マルチ AZ DB クラスター配置 1015
マルチ AZ DB クラスターで利用できるインスタンスクラス 1016
マルチ AZ DB クラスターアーキテクチャ 1016
マルチ AZ DB クラスターのパラメータグループ 1018
マルチ AZ DB クラスターを備えた RDS Proxy 1018
レプリカの遅延とマルチ AZ DB クラスター 1018
マルチ AZ DB クラスターのスナップショット 1022
マルチ AZ DB クラスターの作成 1023
マルチ AZ DB クラスターへの接続 1049
AWS コンピューティングリソースとマルチ AZ DB クラスターを接続する 1056
マルチ AZ DB クラスターの変更 1083
マルチ AZ DB クラスターのアップグレード 1102
マルチ AZ DB クラスターの名前の変更 1105
マルチ AZ DB クラスターの再起動 1108
マルチ AZ DB クラスターをフェイルオーバーする 1110
マルチ AZ DB クラスターとの PostgreSQL 論理レプリケーション 1114
マルチ AZ DB クラスターのリードレプリカの操作 1119
マルチ AZ DB クラスターからの外部レプリケーションの設定 1131
マルチ AZ DB クラスターの削除 1132
マルチ AZ DB クラスターの制約事項 1135
RDS 延長サポート 1136
RDS 延長サポートの概要 1137
RDS 延長サポート料金 1138
RDS 延長サポート料金の回避 1139
RDS 延長サポートが適用されるバージョン 1139
RDS 延長サポートバージョンの命名規則 1140
RDS 延長サポートにおける責任 1141
Amazon RDS の責任 1141
お客様の責任 1141
DB インスタンスまたはマルチ AZ DB クラスターの作成 1142
RDS 延長サポートの動作 1142
RDS 延長サポートに関する考慮事項 1143
RDS 延長サポートで DB インスタンスまたはマルチ AZ DB クラスターを作成する 1144

RDS 延長サポート登録の確認 1145
サポート日の表示 1148
DB インスタンスまたはマルチ AZ DB クラスターの復元 1151
RDS 延長サポートの動作 1151
RDS 延長サポートに関する考慮事項 1152
RDS 延長サポートを使用した DB インスタンスまたはマルチ AZ DB クラスター を復元する 1153
データベースの更新にブルー/グリーンデプロイを使用する 1155
ブルー/グリーンデプロイの概要 1156
リージョンとバージョンの可用性 1157
利点 1157
ワークフロー 1158
アクセスの承認 1162
制約事項と考慮事項 1163
ベストプラクティス 1172
ブルー/グリーンデプロイの作成 1178
ブルー/グリーンデプロイの準備 1179
変更の指定 1181
遅延読み込みとストレージの初期化 1184
ブルー/グリーンデプロイの作成 1186
使用できる設定 1188
ブルー/グリーンデプロイの表示 1194
ブルー/グリーンデプロイの切り替え 1197
切り替えタイムアウト 1198
切り替えガードレール 1198
切り替えアクション 1199
Amazon RDS Proxy を使用する場合のスイッチオーバー 1200
切り替えのベストプラクティス 1201
切り替え前に CloudWatch メトリクスを確認する 1203
スイッチオーバー前のレプリカラグのモニタリング 1203
ブルー/グリーンデプロイの切り替え 1204
切り替え後 1206
ブルー/グリーンデプロイの削除 1208
データのバックアップ、復元、エクスポート 1212
バックアップの概要 1213
バックアップストレージ 1213

自動バックアップの管理 1215
バックアップウィンドウ 1215
バックアップの保存期間 1218
自動バックアップの有効化 1219
自動バックアップの保持 1222
保持している自動バックアップの削除 1224
サポートされていない MySQL ストレージエンジン 1228
サポートされていない MariaDB ストレージエンジン 1229
クロスリージョン自動バックアップ 1230
手動バックアップの管理 1251
シングル AZ DB インスタンスの DB スナップショットの作成 1252
マルチ AZ DB クラスターのスナップショットの作成 1255
DB スナップショットの削除 1257
DB インスタンスへの復元 1260
スナップショットからの復元 1261
考慮事項 1263
ポイントインタイムリカバリ 1268
マルチ AZ DB クラスターを指定の時点の状態に復元する 1274
スナップショットからマルチ AZ DB クラスターへの復元 1278
マルチ AZ DB クラスターのスナップショットから DB インスタンスへの復元 1281
チュートリアル: DB スナップショットからの DB インスタンスの復元 1284
DB スナップショットのコピー 1288
DB スナップショットのコピー 1288
制限事項 1298
考慮事項 1299
DB スナップショットを共有する 1311
スナップショットの共有 1313
公開スナップショットの共有 1316
暗号化されたスナップショットの共有 1318
スナップショット共有の停止 1323
Amazon S3 への DB スナップショットデータのエクスポート 1325
スナップショットデータのエクスポートの概要 1326
S3 バケットへのアクセスを設定する 1327
DB スナップショットのエクスポート 1333
リージョンとバージョンの可用性 1337
制限事項 1337

スナップショットのエクスポートのモニタリング 1338
スナップショットのエクスポートのキャンセル 1341
障害メッセージ 1342
PostgreSQL のアクセス許可エラーのトラブルシューティング 1344
ファイル命名規則 1344
データ変換 1346
AWS Backup を使用する 1356
DB インスタンスでのメトリクスのモニタリング 1357
モニタリング計画 1357
パフォーマンスのベースライン 1358
パフォーマンスガイドライン 1358
モニタリングツール 1360
自動モニタリングツール 1360
手動モニタリングツール 1362
インスタンスのステータスの表示 1364
Amazon RDS DB インスタンスのステータスの表示 1365
Amazon RDS の推奨事項 1373
推奨事項の表示 1375
推奨事項の適用 1381
推奨事項を却下する 1385
却下された推奨事項をアクティブに変更する 1387
推奨事項リファレンス 1389
Amazon RDS コンソールでのメトリクスの表示 1416
Performance Insights ダッシュボードを表示する 1417
[モニタリング] タブで新しいモニタリングビューを選択する 1418
Performance Insights ページから新しいモニタリングビューを選択する 1420
カスタムダッシュボードの作成 1422
事前設定されたダッシュボードの選択 1425
CloudWatch を使用した RDS のモニタリング 1427
Amazon RDS および Amazon CloudWatch の概要 1428
CloudWatch メトリクスの表示 1429
Performance Insights メトリクスの CloudWatch へのエクスポート 1435
CloudWatch アラームの作成 1441
チュートリアル: DB クラスターレプリカラグ用の CloudWatch アラームを作成する 1441
Database Insights によるモニタリング 1449
料金 1449

エンジン、リージョン、インスタンスクラスのサポート 1449
アドバンストモードを有効にする 1455
スタンダードモードを有効にする 1459
スロークエリのモニタリング 1465
考慮事項 1467
Performance Insights を使用した DB 負荷のモニタリング 1468
Performance Insights の概要 1469
Performance Insights の有効化と無効化 1484
MariaDB または MySQL のPerformance Schema 1489
Performance Insights のポリシー 1494
Performance Insights ダッシュボードを使用してメトリクスを分析する 1509
Performance Insights のプロアクティブ推奨事項の表示 1559
Performance Insights API によるメトリクスの取得 1562
AWS CloudTrail を使用した Performance Insights 呼び出しのログ記録 1587
VPC エンドポイント (AWS PrivateLink) 1589
DevOps Guru for RDS でパフォーマンスを分析する 1594
DevOps Guru for RDS の利点 1594
DevOps Guru for RDSはどのように機能しますか 1595
RDS 用の DevOps Guru のセットアップ 1597
拡張モニタリング を使用した OS のモニタリング 1605
Enhanced Monitoring の概要 1605
拡張モニタリングの設定と有効化 1607
RDS コンソールでの OS メトリクスの表示 1613
CloudWatch Logs を使用した OS メトリクスの表示 1617
RDS メトリクスリファレンス 1619
RDS の CloudWatch メトリクス 1619
RDS の CloudWatch ディメンション 1642
Performance Insights の CloudWatch メトリクス 1643
Performance Insights のカウンターメトリクス 1646
Performance Insights の SQL 統計 1683
拡張モニタリングの OS メトリクス 1699
イベント、ログ、およびデータベースアクティビティストリーミングのモニタリング 1715
Amazon RDS コンソールでのログ、イベント、およびストリーミングの表示 1716
RDS イベントのモニタリング 1720
Amazon RDS のイベントの概要 1720
Amazon RDS イベントの表示 1722

Amazon RDS イベント通知の操作 1726
Amazon RDS イベントでトリガーするルールの作成 1752
Amazon RDS イベントカテゴリとイベントメッセージ 1758
RDS ログのモニタリング 1810
データベースログファイルの表示とリスト化 1810
データベースログファイルのダウンロード 1811
データベースログファイルのモニタリング 1813
CloudWatch Logs への発行 1814
REST を用いたログファイルの内容の読み取り 1817
Db2 データベースログファイル 1819
MariaDB データベースのログファイル 1824
Microsoft SQL Server データベースのログファイル 1839
MySQL データベースログファイル 1845
Oracle Database のログファイル 1862
PostgreSQL データベースのログファイル 1873
CloudTrail での RDS API コールのモニタリング 1887
CloudTrail と Amazon RDS の統合 1887
Amazon RDS ログファイルエントリ 1888
データベースアクティビティストリームを使用した RDS のモニタリング 1892
概要 1892
Oracle 統合監査の設定 1899
SQL Server の監査の設定 1900
データベースアクティビティストリーミングのスタート 1902
データベースアクティビティストリーミングの変更 1904
アクティビティストリーミングのステータスの取得 1907
データベースアクティビティストリーミングの停止 1909
アクティビティストリーミングのモニタリング 1910
アクティビティストリームの IAM ポリシーの例 1953
GuardDuty RDS Protection による脅威のモニタリング 1957
Amazon RDS Custom 1958
データベースカスタマイズの課題 1958
RDS Custom の管理モデルと利点 1960
RDS Custom の責任分担モデル 1960
RDS Custom で境界と未サポートの構成をサポート 1963
RDS カスタムの主な利点 1963
RDS Custom アーキテクチャ 1964

VPC 1965
RDS Custom のオートメーションとモニタリング 1965
Amazon S3 1970
AWS CloudTrail 1971
RDS Custom セキュリティ 1971
RDS Custom がユーザーに代わってタスクを安全に管理する方法 1971
SSL 証明書 1973
Amazon S3 バケットを、「混乱した代理」問題から保護する 1973
RDS Custom for Oracle の認証情報のローテーション 1974
RDS Custom for Oracle を使用する 1979
RDS Custom for Oracle のサポート終了 1980
RDS Custom for Oracle ワークフロー 2065
Amazon RDS Custom for Oracle のデータベースアーキテクチャ 2069
RDS Custom for Oracle の機能の可用性とサポート 2071
RDS Custom for Oracle の要件と制限 2074
RDS Custom for Oracle の環境設定 2078
Oracle の RDS Custom の CEVs を使用する 2099
RDS for Oracle DB インスタンスの設定 2132
RDS Custom for Oracle DB インスタンスの管理 2152
RDS Custom for Oracle のマルチ AZ 配置の管理 2171
RDS Custom for Oracle レプリカの使用 2194
RDS Custom for Oracle DB インスタンスのバックアップと復元 2205
RDS Custom for Oracle のオプショングループ使用する 2216
RDS Custom for Oracle への移行 2226
RDS Custom for Oracle DB インスタンスのアップグレード 2227
RDS Custom for Oracle に関するトラブルシューティング 2242
RDS Custom for Oracle の既知の問題 2267
RDS Custom for SQL Server の使用 2270
RDS for SQL Server のワークフロー 2271
RDS Custom for SQL Server の要件と制限 2274
RDS Custom for SQL Server の環境設定 2367
RDS Custom for SQL Server での Bring Your Own Media 2393
RDS Custom for SQL Server の CEV の使用 2395
RDS Custom for SQL Server DB インスタンスを作成して接続する 2418
RDS Custom for SQL Server DB インスタンスの管理 2430
RDS Custom for SQL Server による Microsoft Active Directory の操作 2444

RDS Custom for SQL Server のマルチ AZ 配置の管理 ... 2472
RDS Custom for SQL Server DB インスタンスのバックアップと復元 ... 2488
RDS Custom for SQL Server DB スナップショットのコピー ... 2505
オンプレミスデータベースを RDS Custom for SQL Server へ移行する ... 2516
オペレーティングシステムの更新 ... 2520
RDS Custom for SQL Server の DB インスタンスのアップグレード ... 2523
Amazon RDS Custom for SQL Server に関する問題のトラブルシューティング ... 2527
Amazon RDS on AWS Outposts ... 2566
前提条件 ... 2567
Amazon RDS 機能のサポート ... 2568
サポートされている DB インスタンスクラス ... 2575
顧客所有の IP アドレス ... 2578
CoIP を使用する ... 2578
制限事項 ... 2580
マルチ AZ 配置 ... 2581
責任共有モデルの使用 ... 2581
可用性の向上 ... 2581
前提条件 ... 2582
Amazon EC2 許可に対する API オペレーションの使用 ... 2584
RDS on Outposts の DB インスタンスの作成 ... 2585
RDS on Outposts でのリードレプリカの作成 ... 2595
DB インスタンスの復元に関する考慮事項 ... 2598
Amazon RDS Proxy ... 2599
リージョンとバージョンの可用性 ... 2601
クォータと制限事項 ... 2601
RDS for MariaDB の制限事項 ... 2603
RDS for SQL Server の制限事項 ... 2604
MySQL の制限事項 ... 2605
PostgreSQL の制限事項 ... 2606
RDS Proxy の使用場所の計画 ... 2607
RDS Proxy の概念と用語 ... 2608
RDS Proxy の概念 ... 2609
接続プーリング ... 2610
セキュリティ ... 2611
フェイルオーバー ... 2616
取引 ... 2617

RDS Proxy のスタート方法 2617
プロキシネットワークをセットアップする 2618
データベース認証情報の設定 2621
IAM 認証の設定 2625
プロキシの作成 2632
プロキシの表示 2645
RDS Proxy を介した接続 2647
RDS Proxy の管理 2651
RDS Proxy の変更 2651
データベースユーザーの追加 2659
エンドツーエンド IAM 認証への移行 2660
RDS Proxy 接続に関する考慮事項 2663
RDS Proxy を固定しない 2667
RDS Proxy の削除 2674
RDS Proxy エンドポイントの操作 2675
プロキシエンドポイントの概要 2675
プロキシエンドポイントの制限 2676
マルチ AZ DB クラスターのプロキシエンドポイント 2677
VPC 間の RDS データベースへのアクセス 2678
プロキシエンドポイントの作成 2680
プロキシエンドポイントの表示 2683
プロキシエンドポイントの変更 2684
プロキシエンドポイントの削除 2686
CloudWatch を使用した RDS Proxy のモニタリング 2687
RDS Proxy モニタリングダッシュボード 2697
RDS Proxy イベントの使用 2708
RDS Proxy イベント 2709
RDS Proxy のトラブルシューティング 2711
プロキシでの接続の検証 2712
一般的な問題と解決策 2714
RDS for MySQL のトラブルシューティング 2715
RDS for PostgreSQL のトラブルシューティング 2717
RDS Proxy の AWS CloudFormation での使用 2723
ブルー/グリーンデプロイでの RDS Proxy の使用 2724
RDS Proxy を使用する場合のスイッチオーバー 2724
制限事項 2725

RDS Proxy のベストプラクティス ... 2725
一般的な使用シナリオ ... 2726
アプリケーションとワークロードに関する考慮事項 ... 2732
設定のガイドライン ... 2736
ゼロ ETL 統合 ... 2749
利点 ... 2751
主要なコンセプト ... 2751
制限事項 ... 2752
一般的な制限事項 ... 2752
RDS for MySQL の制限事項 ... 2753
RDS for PostgreSQL の制限事項 ... 2753
RDS for Oracle の制限事項 ... 2754
Amazon Redshift の制限事項 ... 2755
Amazon SageMaker AI Lakehouse の制限事項 ... 2755
クォータ ... 2755
サポート対象のリージョン ... 2756
ゼロ ETL 統合の開始方法 ... 2756
ステップ 1: カスタム DB のパラメータグループを作成する ... 2756
ステップ 2: ソースデータベースを選択または作成する ... 2759
ステップ 3a: ターゲットデータウェアハウスを作成する ... 2759
AWS を使用して統合をセットアップする ... 2761
ステップ 3b: Amazon SageMaker Lakehouse ゼロ ETL 統合用の AWS Glue カタログを作成する ... 2765
次のステップ ... 2769
Amazon Redshift とのゼロ ETL 統合の作成 ... 2769
前提条件 ... 2769
必要なアクセス許可 ... 2769
ゼロ ETL 統合の作成 ... 2773
統合の暗号化 ... 2777
次のステップ ... 2779
Amazon SageMaker Lakehouse とのゼロ ETL 統合の作成 ... 2779
前提条件 ... 2779
必要なアクセス許可 ... 2780
Amazon SageMaker Lakehouse とのゼロ ETL 統合の作成 ... 2782
統合の暗号化 ... 2788
次のステップ ... 2790

ゼロ ETL 統合でのデータフィルタリング 2790
データフィルターの形式 2791
フィルター論理 2793
フィルターの優先順位 2794
RDS for MySQL の例 2794
RDS for PostgreSQL の例 2795
RDS for Oracle の例 2796
データフィルターの追加 2800
データフィルターの削除 2802
データの追加とクエリ 2802
ターゲットデータベースの作成 2803
ソースデータベースへのデータの追加 2803
Amazon Redshift での Amazon RDS データのクエリ 2805
データタイプの相違点 2806
RDS for PostgreSQL の DDL オペレーション 2822
ゼロ ETL 統合の表示と監視 2825
統合の表示 2826
システムテーブルを使ったモニタリング 2827
EventBridge によるモニタリング 2828
ゼロ ETL 統合の変更 2828
ゼロ ETL 統合の削除 2829
ゼロ ETL 統合のトラブルシューティング 2831
ゼロ ETL 統合を作成できない 2831
統合が Syncing の状態でスタックしている 2832
テーブルが Amazon Redshift にレプリケートされない 2832
1 つ以上の Amazon Redshift テーブルを再同期する必要がある 2833
Amazon SageMaker AI Lakehouse ゼロ ETL 統合の統合で失敗した問題 2836
Amazon RDS の Db2 2837
Db2 の概要 2838
Db2 の機能 2839
Db2 のバージョン 2842
Db2 のライセンス 2847
Db2 インスタンスクラス 2865
Db2 のデフォルトロール 2868
Db2 パラメータ 2869
EBCDIC 照合 2875

Db2 ローカルタイムゾーン 2876
DB インスタンスの前提条件 2879
管理者アカウント 2879
その他の考慮事項 2879
複数の Db2 データベース 2881
Db2 DB インスタンスへの接続 2883
エンドポイントの検索 2883
IBM Db2 CLP 2885
IBM CLPPlus 2890
DBeaver 2893
IBM Db2 Data Management Console 2897
セキュリティグループに関する考慮事項 2906
Db2 接続の保護 2908
SSL を使用した暗号化 2908
RDS for Db2 による Active Directory の操作 2916
RDS for Db2 DB インスタンスの管理 2946
システムタスク 2948
データベースタスク 2961
S3 との統合 2991
IAM ポリシーを作成する 2991
IAM ロールを作成して IAM ポリシーをアタッチする 2996
IAM ロールを DB インスタンスに追加する 2999
RDS for Db2 へのデータの移行 3002
AWS のサービスによるデータの移行 3002
ネイティブ Db2 ツールによるデータの移行 3014
フェデレーション 3028
同種フェデレーション 3028
異種フェデレーション 3033
Db2 レプリカの使用 3038
読み取り専用レプリカとスタンバイレプリカ 3038
データベースのアクティブ化 3039
HADR 設定 3039
アーカイブログの保持 3039
Db2 レプリケーション中の停止 3040
Db2 レプリカの要件と考慮事項 3040
Db2 レプリカの作成の準備 3043

スタンバイ Db2 レプリカの作成 ... 3045
レプリカモードの変更 ... 3046
Db2 レプリカのバックアップの使用 ... 3049
Db2 レプリケーションの問題のトラブルシューティング ... 3053
RDS for Db2 DB インスタンスのオプション ... 3056
Db2 監査ログ記録 ... 3057
外部ストアドプロシージャ ... 3073
Java ベースの外部ストアドプロシージャ ... 3073
既知の問題と制限 ... 3082
認証の制限 ... 3082
フェンスされていないルーチン ... 3082
移行中の非自動ストレージテーブルスペース ... 3082
db2_compatibility_vector パラメータの設定 ... 3083
INVALID パッケージを含むデータベースの移行 ... 3083
RDS for Db2 ストアドプロシージャ ... 3085
ストアドプロシージャサポートに関する考慮事項 ... 3095
権限の付与と取り消し ... 3096
監査ポリシー ... 3114
バッファプール ... 3120
データベース ... 3126
ストレージアクセス ... 3172
テーブルスペース ... 3175
RDS for Db2 ユーザー定義関数 ... 3186
rdsadmin.get_task_status ... 3186
rdsadmin.list_databases ... 3191
rdsadmin.list_modifiable_db_params ... 3193
トラブルシューティング ... 3195
データベース接続エラー ... 3195
ファイル I/O エラー ... 3195
ストアドプロシージャエラー ... 3199
Amazon RDS 上の MariaDB ... 3212
MariaDB 機能のサポート ... 3214
MariaDB のメジャーバージョン ... 3214
サポートされているストレージエンジン ... 3226
キャッシュウォームアップ ... 3227
サポートされていない機能 ... 3229

MariaDB のバージョン 3231
バージョン最新性タイムライン 3231
サポートされている MariaDB のマイナーバージョン 3232
サポートされている MariaDB のメジャーバージョン 3242
データベースプレビュー環境 3243
データベースプレビュー環境の MariaDB バージョン 12.3.1 3247
非推奨とされた MariaDB バージョン 3247
MariaDB を実行している DB インスタンスへの接続 3248
接続情報を検索する 3249
コマンドラインクライアントからの接続 3253
AWS ドライバーとの接続 3253
トラブルシューティング 3254
MariaDB 接続の保護 3256
MariaDB のセキュリティ 3256
パスワード検証プラグイン 3259
SSL を使用した暗号化 3260
新しい SSL/TLS 証明書の使用 3264
RDS Optimized Reads によるクエリパフォーマンスの向上 3270
概要 3270
ユースケース 3271
ベストプラクティス 3271
使用 3272
モニタリング 3273
制限 3273
RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上 3275
概要 3275
新しいデータベースでの使用 3276
既存のデータベースで有効にする 3281
制限事項 3282
MariaDB DB エンジンのアップグレード 3283
考慮事項 3284
有効なアップグレードターゲットの検索 3285
MariaDB のバージョン番号 3286
RDS バージョン番号 3288
メジャーバージョンのアップグレード 3289
MariaDB DB インスタンスのアップグレード 3290

マイナーバージョンの自動アップグレード 3290
ダウンタイムの短縮によるアップグレード 3293
イベントによるモニタリング 3297
MariaDB DB スナップショットエンジンバージョンのアップグレード 3298
サポートされていないエンジンバージョンのアップグレードオプション 3300
MariaDB DB インスタンスへのデータのインポート 3302
データのインポートに関する考慮事項 3306
外部のデータベースからデータをインポートする 3312
ダウンタイムを短縮してデータをインポートする 3316
任意のソースからデータをインポートする 3334
MariaDB レプリケーション 3341
MariaDB リードレプリカ 3342
外部ソースインスタンスを使用した GTID ベースのレプリケーションを設定する 3357
外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定 3362
MariaDB のオプション 3368
MariaDB 監査プラグインのサポート 3368
MariaDB のパラメータ 3375
MariaDB パラメータの表示 3375
使用できない MySQL パラメータ 3377
MySQL DB スナップショットから MariaDB DB インスタンスへのデータ移行 3380
移行の実行 3380
MariaDB と MySQL の間の非互換性 3382
Amazon RDS SQL での MariaDB リファレンス 3384
mysql.rds_replica_status 3384
mysql.rds_set_external_master_gtid 3386
mysql.rds_kill_query_id 3389
mysql.rds_execute_operation 3390
ローカルタイムゾーン 3393
MariaDB の既知の問題と制限 3396
ファイルサイズの制限 3396
InnoDB 予約語 3398
カスタムポート 3398
Performance Insights 3398
Amazon RDS 上の Microsoft SQL Server 3399
一般的な管理タスク 3401

制限事項 3404
DB インスタンスクラスのサポート 3407
CPUの最適化 3420
制限事項 3421
DB インスタンスクラスのサポート 3422
CPU コアとスレッドを設定する 3437
セキュリティ 3441
SSL サポート 3442
SQL Server DB インスタンスでの SSL の使用 3443
SQL Server のセキュリティプロトコルおよび暗号の設定 3449
新しい SSL/TLS 証明書を反映するためのアプリケーションの更新 3456
コンプライアンスプログラム 3460
HIPAA 3460
バージョンのサポート 3461
バージョン管理 3467
バージョンポリシー 3470
メジャーバージョン 3470
マイナーバージョン 3471
新しいメジャーバージョンのサポート 3476
メジャーバージョンの利用可能性 3477
マイナーバージョンのリリース頻度 3477
マイナーバージョンの利用可能性 3477
バージョンの廃止 3483
必須アップグレード 3484
アップグレード前のテスト 3484
機能のサポート 3484
SQL Server 2025 の機能 3485
SQL Server 2022 の機能 3486
SQL Server 2019 の機能 3487
SQL Server 2017 の機能 3488
SQL Server 2016 の機能 3488
SQL Server 2014 の機能 3488
SQL Server 2012 が Amazon RDS でのサポートを終了 3489
SQL Server 2008 R2 が Amazon RDS でのサポートを終了 3489
CDC サポート 3489
サポートされていない、および制限された機能のサポート 3490

マルチ AZ 配置 3492
TDE の使用 3492
関数とストアドプロシージャ 3492
ローカルタイムゾーン 3501
サポートされているタイムゾーン 3501
SQL Server のライセンス 3514
ライセンス込み 3514
Bring Your Own Media 3514
開発とテスト 3515
マルチ AZ 配置 3515
Web Edition の使用 3515
終了したインスタンスの復元 3516
SQL Server を実行する DB インスタンスへの接続 3517
接続する前に 3517
DB インスタンスのエンドポイントとポート番号の検索 3518
SSMS を使用して DB インスタンスに接続する 3519
SQL Workbench/J を使用して DB インスタンスに接続する 3522
セキュリティグループに関する考慮事項 3524
トラブルシューティング 3525
SQL Server Developer Edition 3527
利点 3527
リージョン 3527
ライセンスと使用 3528
前提条件 3529
制限事項 3531
CEV の準備 3531
カスタムエンジンバージョンの作成 3533
Developer Edition DB インスタンスの作成 3539
データベースのマイナーバージョンアップグレードの適用 3542
カスタムエンジンバージョンの表示と管理 3542
RDS for SQL Server の SQL Server Developer Edition のトラブルシューティング 3544
Bring Your Own Media (BYOM) 3546
利点 3546
BYOM の仕組み 3547
BYOM とライセンス込み 3548
サポートされるエディション 3549

サポートされているインスタンス 3549
リージョンの可用性 3549
料金 3550
License Manager 統合 3550
制限事項 3551
BYOM エンジンバージョンの作成と管理 3552
BYOM DB インスタンスの作成 3560
BYOM DB インスタンスをアップグレードする 3563
License Manager との統合 3567
Developer Edition との比較 3570
BYOM のトラブルシューティング 3571
RDS for SQL Server による Active Directory の操作 3577
セルフマネージド型のアクティブディレクトリの操作 3578
RDS for SQL Server による AWS Managed Active Directory の操作 3602
SQL Server DB エンジンのアップグレード 3619
メジャーバージョンのアップグレード 3620
アップグレードに関する考慮事項 3623
アップグレードをテストする 3626
SQL Server DB インスタンスをアップグレードする 3627
サポート終了前に非推奨の DB インスタンスをアップグレードする 3627
SQL Server のストレージの使用 3629
考慮事項 3629
ストレージの追加、削除、変更 3630
復元オペレーション 3632
ユースケース 3634
SQL Server データベースのインポートとエクスポート 3637
制限と推奨事項 3640
セットアップ 3642
ネイティブバックアップおよび復元の使用 3649
バックアップファイルの圧縮 3669
トラブルシューティング 3670
他の方法による SQL Server データのインポートとエクスポート 3674
Linux の BCP の使用 3689
SQL Server リードレプリカ 3695
SQL Server リードレプリカの設定 3695
SQL Server でのリードレプリカの制限 3696

オプションに関する考慮事項 3697
データベースユーザーとオブジェクトの同期 3699
トラブルシューティング 3701
RDS for SQL Server 用のマルチ AZ 3702
SQL Server DB インスタンスへのマルチ AZ の追加 3704
SQL Server DB インスタンスからのマルチ AZ の削除 3705
制限事項、注意事項、および推奨事項 3705
セカンダリの場所を確認する 3712
Always On AG への移行 3713
SQL Server の追加機能 3715
SQL Server DB インスタンスでのパスワードポリシーの使用 3716
Amazon S3 統合 3723
データベースメールの使用 3745
tempdb のインスタンスストアのサポート 3761
拡張イベントの使用 3764
トランザクションログのバックアップへのアクセス 3768
SQL Server のオプション 3807
SQL Server のバージョンと エディションで使用できるオプションの一覧表示 3810
Oracle OLEDB とリンクされたサーバー 3812
Teradata ODBC を使用したリンクサーバー 3825
ネイティブバックアップおよび復元 3835
透過的なデータ暗号化 3840
SQL Server Audit 3854
SQL Server Analysis Services 3869
SQL Server Integration Services 3899
SQL Server Reporting Services と Power BI Report Server 3923
Microsoft 分散トランザクションコーディネーター 3952
SQL Server リソースガバナー 3970
一般的な DBA タスク 4002
tempdb データベースへのアクセス 4004
Database Engine Tuning Advisor を使用したデータベースワークロードの分析 4008
db_owner をデータベースの rdsa アカウントに変更する 4012
照合順序と文字セットの管理 4013
データベースユーザーの作成 4022
復旧モデルの確認 4023
最後のフェイルオーバー時間の確認 4024

LSN ギャップによる PITR 障害のトラブルシューティング 4025
データベース名の表示を拒否または許可する 4026
高速挿入の無効化 4027
SQL Server データベースの削除 4028
マルチ AZ データベースの名前を変更する 4029
マスターユーザーの db_owner ロールメンバーシップのリセット 4029
ライセンス終了した DB インスタンスの復元 4030
データベースをオフラインからオンラインに切り替える 4031
CDC の使用 4031
SQL Server エージェントの使用 4035
SQL Server のログの使用 4042
トレースファイルおよびダンプファイルの使用 4043
Amazon RDS 上の MySQL 4045
MySQL 機能のサポート 4048
MySQL メジャーバージョン 4048
サポートされているストレージエンジン 4050
memcached およびその他のオプションの使用 4051
InnoDB キャッシュウォームアップ 4051
MySQL 8.4 に対する包括的な言語変更 4053
サポートされていない機能 4056
MySQL のバージョン 4058
バージョン最新性タイムライン 4058
マイナーバージョン 4059
メジャーバージョン 4068
RDS 延長サポートバージョン 4069
データベースプレビュー環境 4077
9.7 (プレビュー) 4081
9.6 (プレビュー) 4081
9.5 (プレビュー) 4082
非推奨バージョン 4082
MySQL を実行している DB インスタンスへの接続 4083
接続情報を検索する 4084
コマンドラインクライアントのインストール 4088
コマンドラインクライアントからの接続 4088
MySQL Workbench からの接続 4089
AWS ドライバーとの接続 4091

トラブルシューティング 4092
MySQL 接続の保護 4094
パスワードの検証 4097
SSL を使用した暗号化 4098
新しい SSL/TLS 証明書の使用 4102
MySQL の Kerberos 認証の使用 4108
RDS Optimized Reads によるクエリパフォーマンスの向上 4123
概要 4123
ユースケース 4124
ベストプラクティス 4125
使用 4126
モニタリング 4126
制限 4127
RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上 4128
概要 3275
新しいデータベースでの使用 4129
既存のデータベースで有効にする 4134
制限事項 4135
MySQL DB エンジンのアップグレード 4136
考慮事項 4138
有効なアップグレードターゲットの検索 4138
MySQL バージョン番号 4139
RDS バージョン番号 4141
メジャーバージョンのアップグレード 4142
アップグレードをテストする 4148
MySQL DB インスタンスをアップグレードする 4149
マイナーバージョンの自動アップグレード 4149
ダウンタイムの短縮によるアップグレード 4152
イベントによるモニタリング 4156
MySQL DB スナップショットエンジンバージョンのアップグレード 4157
サポートされていないエンジンバージョンのアップグレードオプション 4159
MySQL DB インスタンスへのデータのインポート 4162
データのインポートに関する考慮事項 4166
MySQL DB インスタンスへのバックアップの復元 4172
外部のデータベースからデータをインポートする 4188
ダウンタイムを短縮してデータをインポートする 4192

任意のソースからデータをインポートする 4211
MySQL レプリケーション 4218
MySQL リードレプリカ 4219
GTID ベースのレプリケーション 4240
外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定 4248
マルチソースレプリケーションの設定 4254
アクティブ/アクティブクラスターの設定 4262
ユースケース 4262
アクティブ/アクティブクラスターの制限と考慮事項 4263
クロス VPC アクティブ/アクティブクラスターの準備 4267
アクティブ/アクティブクラスターに必要なパラメータ設定 4269
DB インスタンスをアクティブ/アクティブクラスターに変換する 4271
アクティブ/アクティブクラスターのセットアップ 4278
DB インスタンスをアクティブ/アクティブクラスターに追加する 4284
アクティブ/アクティブクラスターのモニタリング 4287
アクティブ/アクティブクラスター内のグループレプリケーションを停止する 4288
アクティブ/アクティブクラスター内の DB インスタンスの名前を変更する 4289
アクティブ/アクティブクラスターから DB インスタンスを削除する 4290
MySQL DB インスタンスからのデータのエクスポート 4291
外部の MySQL データベースの準備 4291
ソース MySQL DB インスタンスの準備 4292
データベースのコピー 4294
エクスポートの完了 4295
MySQL のオプション 4298
MariaDB 監査プラグイン 4299
memcached 4307
MySQL のパラメータ 4313
MySQL の一般的な DBA タスク 4315
定義済みユーザーとは 4315
ロールベースの特権モデル 4316
動的権限 4319
セッションやクエリの終了 4323
現在のレプリケーションエラーのスキップ 4324
クラッシュリカバリ時間を改善する 4326
Global Status History の管理 4329

バッファプールサイズと REDO ログ容量の設定 4331
ローカルタイムゾーン 4333
既知の問題と制限 4336
InnoDB 予約語 4336
ストレージ全体の動作 4336
InnoDB バッファプールサイズの不整合 4337
インデックスマージの最適化で誤った結果が返される 4338
Amazon RDS DB インスタンスに使用する MySQL のパラメータの例外 4339
Amazon RDS での MySQL のファイルサイズ制限 4340
MySQL キーリングプラグインがサポートされていない 4342
カスタムポート 4342
MySQL ストアドプロシージャの制限事項 4343
外部ソースインスタンスを使用した GTID ベースのレプリケーション 4343
MySQL デフォルト認証プラグイン 4343
innodb_buffer_pool_size の上書き 4343
MySQL 5.7 から MySQL 8.4 へのアップグレード 4344
InnoDB ページ圧縮 4344
RDS for MySQL のストアドプロシージャ 4345
グローバルステータス履歴の収集と維持 4346
バイナリログレプリケーションの設定、開始、停止 4349
セッションやクエリの終了 4391
アクティブ/アクティブクラスターの管理 4393
マルチソースレプリケーションの管理 4398
GTID を使用したトランザクションのレプリケーション 4421
クエリログのローテーション 4424
バイナリログ構成の設定と表示 4426
InnoDB キャッシュのウォームアップ 4430
Amazon RDS 上の Oracle 4432
Oracle の概要 4434
Oracle の機能 4434
Oracle バージョン 4440
Oracle のライセンス 4454
Oracle のユーザーと権限 4458
Oracle インスタンスクラス 4460
Oracle データベースアーキテクチャ 4470
Oracle のパラメータ 4472

Oracle 文字セット 4474
Oracle の制限事項 4478
Oracle DB インスタンスへの接続 4481
エンドポイントの検索 4481
SQL Developer 4483
SQL*Plus 4486
セキュリティグループに関する考慮事項 4488
専用サーバーと共有サーバーのプロセス 4488
トラブルシューティング 4488
Oracle sqlnet.ora パラメータの変更 4491
Oracle 接続の保護 4496
SSL を使用した暗号化 4496
新しい SSL/TLS 証明書の使用 4497
NNE を使用した暗号化 4501
Kerberos 認証の設定 4502
UTL_HTTP アクセスの設定 4521
CDB を使用する 4534
CDB の概要 4534
CDB の設定 4540
CDB のバックアップと復元 4547
非 CDB から CDB への変換 4548
シングルテナントからマルチテナント設定への変換 4551
RDS for Oracle テナントデータベースを CDB インスタンスに追加する 4554
RDS for Oracle テナントデータベースの変更 4557
RDS for Oracle テナントデータベースを CDB から削除する 4559
テナントデータベースの詳細を表示する 4562
CDB のアップグレード 4567
Oracle DB インスタンスの管理 4568
システムタスク 4583
データベースタスク 4609
データベースのチェックポイント機能 4627
分散復旧の設定 4627
データベースタイムゾーンの設定 4628
AWR の操作 4629
VPC の DB インスタンスで使用するデータベースリンクの調整 4635
デフォルトエディションの設定 4635

SYS.AUD$ テーブルの監査を有効にする 4636
SYS.AUD$ テーブルの監査を無効にする 4637
中断したオンラインインデックス構築のクリーンアップ 4637
破損ブロックのスキップ 4639
ごみ箱を空にする 4642
フルリダクションのデフォルト表示値の設定 4642
ログタスク 4644
RMAN タスク 4658
Oracle Scheduler タスク 4692
リアルタイム SQL 計画管理 4702
テーブルスペースの縮小 4706
問題の診断 4710
その他のタスク 4719
Oracle ストレージの使用 4735
考慮事項 4736
制限事項 4737
データベース管理オペレーション 4738
ストレージボリュームの変更 4738
ストレージボリュームのバックアップと復元 4740
ユースケース 4742
RDS for Oracle の高度な機能の設定 4745
インスタンスストアの設定 4745
HugePages をオンにする 4756
拡張データ型を有効にする 4760
AI 機能 4764
Bedrock の統合 4764
Oracle へのデータのインポート 4771
Oracle SQL Developer を使用したインポート 4773
Oracle トランスポータブル表領域を使用した移行 4773
Oracle Data Pump を使用したインポート 4790
Oracle エクスポート/インポートを使用したインポート 4809
Oracle SQL*Loader を使用したインポート 4810
Oracle マテリアライズドビューを使用した移行 4812
Oracle レプリカの使用 4815
Oracle レプリカの概要 4815
Oracle レプリカの要件と考慮事項 4818

Oracle レプリカの作成の準備 4823
マウントされた Oracle レプリカの作成 4825
レプリカモードの変更 4826
Oracle レプリカのバックアップの使用 4828
Oracle Data Guard のスイッチオーバー操作の実行 4830
Oracle レプリカのトラブルシューティング 4839
REDO トランスポート圧縮 4841
Oracle のオプション 4843
Oracle DB オプションの概要 4843
Amazon S3 統合 4846
Oracle Application Express (APEX) 4874
Amazon EFS の統合 4900
Java Virtual Machine (JVM) 4918
Enterprise Manager 4923
Label Security 4951
Locator 4955
ネイティブネットワーク暗号化 (NNE) 4960
OLAP 4977
Secure Sockets Layer (SSL) 4981
Spatial 4996
SQLT 5001
Statspack 5011
タイムゾーン 5015
タイムゾーンファイルの自動アップグレード 5021
Transparent Data Encryption (TDE) 5032
UTL_MAIL 5037
XML DB 5041
Oracle DB エンジンのアップグレード 5042
Oracle アップグレードの概要 5042
メジャーバージョンのアップグレード 5050
マイナーバージョンのアップグレード 5053
アップグレードに関する考慮事項 5064
アップグレードをテストする 5068
RDS for Oracle DB インスタンスのアップグレード 5069
Oracle DB スナップショットのアップグレード 5071
Oracle 用のツールおよびサードパーティーソフトウェア 5074

Oracle GoldenGate の使用 5075
Oracle Repository Creation Utility の使用 5095
CMAN の設定 5103
Amazon RDS での Oracle への Siebel Database のインストール 5106
Oracle データベースエンジンのリリース 5111
Amazon RDS 上の PostgreSQL 5112
一般的な管理タスク 5114
データベースプレビュー環境の使用 5118
データベースプレビュー環境でサポートされない機能 5119
データベースプレビュー環境の PostgreSQL バージョン 17 5119
データベースプレビュー環境での新しい DB インスタンスの作成 5119
PostgreSQL バージョン 5122
廃止予定の PostgreSQL バージョン 5122
RDS for PostgreSQL リリースプロセス 5123
RDS for PostgreSQL 増分リリースプロセスの利点 5123
リリース更新の管理 5123
PostgreSQL 拡張機能バージョン 5125
PostgreSQL エクステンションのインストールを制限する 5125
PostgreSQL 信頼できるエクステンション 5127
PostgreSQL の機能 5129
カスタムデータ型および列挙型 5130
イベントトリガー 5130
Huge ページ 5131
論理レプリケーションの実行 5132
論理レプリケーションの IAM 認証の設定 5139
stats_temp_directory の RAM ディスク 5142
テーブルスペース 5143
EBCDIC やその他のメインフレーム移行のための照合順序 5144
論理スロット同期の管理 5149
PostgreSQL インスタンスに接続する 5151
psql クライアントをインストールする 5152
接続情報を検索する 5152
pgAdmin を使用して RDS PostgreSQL DB インスタンスに接続する 5154
psql を使用したRDS for PostgreSQL DB インスタンスへの接続 5156
AWS JDBC ドライバーを使用した RDS for PostgreSQL への接続 5158
AWS Python ドライバーを使用した RDS for PostgreSQL への接続 5158

RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング 5158
SSL/TLS を使用した接続の保護 5161
PostgreSQL DB インスタンスで SSL を使用する 5161
新しい SSL/TLS 証明書を使用するためのアプリケーションの更新 5170
Kerberos 認証を使用する 5175
リージョンとバージョンの可用性 5176
Kerberos 認証の概要 5176
設定 5177
Active Directory ドメインの RDS for PostgreSQL DB インスタンスの管理 5192
Kerberos 認証に接続する 5193
アウトバウンドネットワークアクセスでカスタム DNS サーバーを使用する 5197
カスタム DNS 解決をオンにする 5197
カスタム DNS 解決をオフにする 5197
カスタム DNS サーバーのセットアップ 5197
PostgreSQL DB エンジンのアップグレード 5200
考慮事項 5202
有効なアップグレードターゲットの検索 5203
PostgreSQL のバージョン番号 5204
RDS バージョン番号 5204
メジャーバージョンアップグレードの選択 5205
メジャーバージョンのアップグレードを実施する方法 5217
マイナーバージョンの自動アップグレード 5228
PostgreSQL 拡張機能のアップグレード 5231
イベントによるモニタリング 5232
PostgreSQL DB スナップショットエンジンのバージョンのアップグレード 5233
RDS for PostgreSQL のリードレプリカ 5236
PostgreSQL でのリードレプリカの制限 5236
PostgreSQL でのリードレプリカの設定 5237
リードレプリカの論理デコード 5240
カスケードリードレプリカの使用 5243
クロスリージョンカスケーディングリードレプリカの作成 5245
RDS for PostgreSQL のバージョンが異なる場合のレプリケーションの仕組み 5247
レプリケーションプロセスのモニタリングとチューニング 5251
RDS for PostgreSQL での遅延レプリケーションの設定 5254
RDS for PostgreSQL リードレプリカのトラブルシューティング 5258
RDS Optimized Reads によるクエリパフォーマンスの向上 5260

PostgreSQL の RDS Optimized Reads の概要 5260
ユースケース 5261
ベストプラクティス 5262
使用 5262
モニタリング 5263
制限 5264
PostgreSQL にデータをインポートする 5265
Amazon EC2 インスタンスから PostgreSQL データベースをインポートする 5267
\copy コマンドを使用して PostgreSQL DB インスタンスのテーブルにデータをインポートする 5270
Amazon S3 から RDS for PostgreSQL にデータをインポートする 5271
DB インスタンス 間での PostgreSQL データベースの移行 5291
PostgreSQL データを Amazon S3 にエクスポートする 5300
拡張機能のインストール 5301
S3 へのエクスポートの概要 5302
エクスポート先の Amazon S3 ファイルパスを指定する 5303
Amazon S3 バケットへのアクセスを設定する 5304
aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート 5309
関数リファレンス 5312
Amazon S3 へのアクセスのトラブルシューティング 5317
RDS for PostgreSQL から Lambda 関数を呼び出す 5318
ステップ 1: アウトバウンド接続を設定する 5319
ステップ 2: インスタンスおよび Lambda のために IAM を設定する 5320
ステップ 3: 拡張機能をインストールする 5322
ステップ 4: Lambda のヘルパー関数を使用する 5323
ステップ 5: Lambda 関数を呼び出す 5324
ステップ 6: ユーザーに許可を付与する 5325
例: Lambda 関数の呼び出し 5325
Lambda 関数のエラーメッセージ 5328
Lambda 関数とパラメータのリファレンス 5329
RDS for PostgreSQL の一般的な DBA タスク 5335
RDS for PostgreSQL でサポートされる照合 5336
PostgreSQL のロールとアクセス権限について 5337
PostgreSQL でのデッド接続の処理 5353
PostgreSQL Autovacuum の使用 5356
Amazon RDS for PostgreSQL での高いオブジェクト数の管理 5405

TOAST OID 競合の管理 5413
ログ記録メカニズム 5421
PostgreSQL による一時ファイルの管理 5423
pgBadger を使用した PostgreSQL でのログ分析 5429
PostgreSQL をモニタリングするために PGSnapper を使用する 5429
カスタムキャストの管理 5429
RDS for PostgreSQL での並列クエリのベストプラクティス 5433
REPLICA IDENTITY FULL パフォーマンス 5438
PostgreSQL の初期トラブルシューティング 5447
パラメータの使用 5454
RDS for PostgreSQL の待機イベントでのチューニング 5475
RDS for PostgreSQL チューニングの基本概念 5476
RDS for PostgreSQL 待機イベント 5481
Client:ClientRead 5483
クライアント: ClientWrite 5487
CPU 5489
IO:BufFileRead および IO:BufFileWrite 5496
IO:DataFileRead 5504
IO:WALWrite 5512
IPC:parallel 待機イベント 5515
IPC:ProcArrayGroupUpdate 5521
Lock:advisory 5524
Lock:extend 5527
Lock:Relation 5530
Lock:transactionid 5533
Lock:tuple 5536
LWLock:BufferMapping (LWLock:buffer_mapping) 5541
LWLock:BufferIO (IPC:BufferIO) 5543
LWLock:buffer_content (BufferContent) 5545
LWLock:lock_manager (LWLock:lockmanager) 5547
LWLock:pg_stat_statements 5553
LWLock:SubtransSLRU (LWLock:SubtransControlLock) 5557
Timeout:PgSleep 5564
Timeout:VacuumDelay 5565
Amazon DevOps Guru のプロアクティブインサイトによる RDS for PostgreSQL のチューニング 5568

データベースがトランザクション接続で長時間アイドル状態になっている 5568
PostgreSQL 拡張機能の使用 5572
orafce の関数の使用 5573
PostgreSQL での Amazon RDS 委任拡張機能サポートの使用 5575
pg_partman エクステンションによるパーティションの管理 5591
pgAudit を使用してデータベースのアクティビティを記録する 5598
pg_cron エクステンションによるメンテナンスのスケジューリング 5611
pglogical を使用してデータを同期する 5621
pgactive を使用したアクティブ/アクティブレプリケーションの作成 5635
pg_repack 拡張機能を使用して膨張を抑制する 5702
PLV8 のアップグレードおよび使用 5708
PL/Rust を使って Rust 言語で関数を記述する 5711
PostGIS を使用した空間データの管理 5716
Amazon RDS for PostgreSQL でサポートされている外部データラッパー 5726
log_fdw エクステンションの使用 5726
postgres_fdw を使用した外部データへのアクセス 5728
MySQL データベースの操作 5729
Oracle データベースの操作 5733
SQL Server データベースの操作 5738
Trusted Language Extensions for PostgreSQL を使用した操作 5741
用語 5742
Trusted Language Extensions を使用するための要件 5743
Trusted Language Extensions の設定 5746
Trusted Language Extensions の概要 5750
TLE 拡張機能の作成 5752
TLE 拡張機能をデータベースから削除する 5757
Trusted Language Extensions のアンインストール 5758
TLE 拡張機能で PostgreSQL フックを使用する 5759
信頼できる言語拡張機能でのカスタムデータ型の使用 5765
Trusted Language Extensions の関数リファレンス 5766
Trusted Language Extensions のフックリファレンス 5779
コードの例 5783
基本 5784
Hello Amazon RDS 5785
基本を学ぶ 5795
アクション 5915

シナリオ 6056
Aurora Serverless 作業項目トラッカーの作成 6057
リレーショナルデータベースインスタンスの作成 6061
サーバーレスサンプル 6069
Lambda 関数での Amazon RDS データベースへの接続 6069
セキュリティ 6089
データベース認証 6091
パスワード認証 6092
IAM データベース認証 6092
Kerberos 認証 6093
RDS と Secrets Manager によるパスワード管理 6095
制限事項 6095
概要 6096
利点 6097
アクセス許可 6097
RDS 管理の強化 6098
DB インスタンスのパスワードの管理 6099
テナントデータベースのパスワードの管理 6104
マルチ AZ DB クラスターのパスワードの管理 6108
DB インスタンスのシークレットのローテーション 6112
Multi-AZ DB クラスターのシークレットのローテーション 6114
シークレットの詳細を表示する 6116
マルチ AZ DB クラスターのシークレットの詳細を表示する 6119
テナントデータベースのシークレットの詳細の表示 6122
リージョンとバージョンの可用性 6125
データ保護 6125
データの暗号化 6126
ネットワーク間のトラフィックのプライバシー 6158
Identity and Access Management 6160
対象者 6160
アイデンティティを使用した認証 6161
ポリシーを使用したアクセスの管理 6164
Amazon RDS と IAM の連携 6166
アイデンティティベースのポリシーの例 6173
AWS マネージドポリシー 6195
ポリシーの更新 6203

サービス間の混乱した代理の防止 6223
IAM データベース認証 6225
トラブルシューティング 6274
ログ記録とモニタリング 6276
コンプライアンス検証 6279
耐障害性 6280
バックアップと復元 6280
レプリケーション 6280
フェイルオーバー 6281
インフラストラクチャセキュリティ 6282
セキュリティグループ 6282
パブリックアクセシビリティ 6282
VPC エンドポイント (AWS PrivateLink) 6284
考慮事項 1590
利用可能な状況 1590
インターフェイス VPC エンドポイントの作成 1590
VPC エンドポイントポリシーの作成 1591
セキュリティのベストプラクティス 6288
セキュリティグループによるアクセス制御 6289
VPC セキュリティグループの概要 6289
セキュリティグループのシナリオ 6290
VPC セキュリティグループを作成する 6291
DB インスタンスとの関連付け 6292
マスターユーザーアカウント権限 6292
サービスにリンクされたロール 6298
Amazon RDS のサービスにリンクされたロールのアクセス許可 6298
Amazon RDS Custom のサービスにリンクされたロール許可 6302
Amazon RDS Beta のサービスにリンクされたロール許可 6303
Amazon RDS Preview のサービスリンクロール 6304
Amazon RDS を Amazon VPC に使用する 6306
VPC 内の DB インスタンスの使用 6306
DB インスタンスの VPC の更新 6326
VPC の DB インスタンスにアクセスするシナリオ 6327
チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用) 6334
チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード) 6342
DB インスタンスを VPC 内に移行する 6353

クォータと制約 6356
Amazon RDS のクォータ 6356
Amazon RDS の命名に関する制約 6363
データベース接続の最大数 6365
Amazon RDS のファイルサイズ制限 6368
トラブルシューティング 6369
DB インスタンスに接続できない 6369
DB インスタンス接続のテスト 6372
接続認証のトラブルシューティング 6373
セキュリティの問題 6373
エラーメッセージ「アカウント属性の取得に失敗しました。コンソールの特定の機能が損なわれる可能性があります。」 6373
互換性のないネットワーク状態のトラブルシューティング 6373
原因 6373
解像度 6374
DB インスタンス所有者のパスワードのリセット 6376
DB インスタンスの停止または再起動 6376
パラメータの変更が有効にならない 6377
DB インスタンスのストレージ不足 6377
利用可能な DB インスタンスが不足 6379
RDS の解放可能なメモリの問題 6380
MySQL および MariaDB の問題 6380
MySQL および MariaDB の最大接続数 6381
メモリ制限と互換性のないパラメータの状態の診断と解決 6381
リードレプリカ間の遅延の診断と解決 6384
MySQL または MariaDB のリードレプリケーションのエラーの診断と解決 6386
バイナリログが有効な場合に SUPER 権限が必要になるトリガーの作成 6387
ポイントインタイム復元のエラーの診断と解決 6389
レプリケーション停止エラー 6390
致命的なエラー 1236 によるリードレプリカの作成の失敗またはレプリケーションの中断 6391
リードレプリカのレプリケーションがメタデータ構造の初期化に失敗する 6391
バックアップ保持期間を 0 に設定できない 6392
Amazon RDS API リファレンス 6393
クエリ API の使用 6393
クエリパラメータ 6393
クエリリクエストの認証 6394

アプリケーションのトラブルシューティング 6394
エラーの取得 6394
トラブルシューティングのヒント 6395
ドキュメント履歴 6396
以前の更新 6604
AWS 用語集 6637

Amazon Relational Database Service (Amazon RDS) とは

Amazon Relational Database Service (Amazon RDS) は、AWS クラウド でリレーショナルデータベースを簡単にセットアップし、運用し、スケーリングすることのできるウェブサービスです。業界スタンダードのリレーショナルデータベース向けに、費用対効果に優れたエクステンションを備え、一般的なデータベース管理タスクを管理します。

> **Note**
>
> このガイドでは、Amazon Aurora 以外の Amazon RDS データベースエンジンについて説明しています。Amazon Aurora の使用については、Amazon Aurora ユーザーガイドを参照してください。

AWS の製品やサービスを初めて使用する場合、詳細については、以下のリソースを参照してください。

- すべての AWS 製品の概要については、「クラウドコンピューティングとは」を参照してください。
- Amazon Web Services では、数多くのデータベースサービスを提供しています。AWS で利用できるさまざまなデータベースオプションの詳細については、「AWS データベースサービスの選択」と「AWS でのデータベースの実行」を参照してください。

Amazon RDS の利点

Amazon RDS はマネージドデータベースサービスです。これは、ほとんどの管理タスクを担っています。Amazon RDS では、面倒な手動プロセスを排除することでアプリケーションとユーザーに集中することができます。

Amazon RDS は、フルマネージドではないデータベースデプロイに比べて、次のプリンシパルな利点があります。

- IBM Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle Database、PostgreSQL など既に使い慣れたデータベースエンジンを使用できます。
- Amazon RDS では、バックアップ、ソフトウェアパッチ、自動的な障害検出、および復旧を管理します。

- 自動バックアップをオンにするか、マニュアルで独自のバックアップスナップショットを作成できます。これらのバックアップを使用してデータベースを復元できます。Amazon RDS の復元は信頼性の高い効率的なプロセスです。
- プライマリ DB インスタンスと同期しているセカンダリ DB インスタンスがあると、問題が発生したときにセカンダリインスタンスにフェイルオーバーできるので、高可用性を実現できます。また、リードレプリカを使用して、読み取りスケーリングを拡張できます。
- データベースパッケージのセキュリティに加え、AWS Identity and Access Management (IAM) を使用してユーザーとアクセス許可を定義すると、アクセスを制御できます。また、仮想プライベートクラウド (VPC) に配置すると、データベースを保護することもできます。

Amazon EC2 とオンプレミスデプロイの責任の比較

ほとんどのリレーショナルデータベースデプロイでは、デフォルトの選択肢として Amazon RDS をお勧めしています。以下の代替方法には、ソフトウェアとハードウェアの管理により多くの時間を費やすという欠点があります。

オンプレミスのデプロイ

オンプレミスサーバーを購入するときは、CPU、メモリ、ストレージ、IOPS をすべて一緒にまとめて入手します。サーバー、オペレーティングシステム、およびデータベースソフトウェアに関する全責任を負うものとします。

(Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) は、でスケーラブルなコンピューティング容量を提供しますAWS クラウド オンプレミスのサーバーとは異なり、CPU、メモリ、ストレージ、IOPS は分離されているため、個別にスケールできます。AWS はハードウェアレイヤーを管理するため、オンプレミスのデータベースサーバーを管理する負担の一部を軽減します。

Amazon EC2 でデータベースを実行することの欠点は、ユーザーエラーが発生しやすいことです。例えば、OSまたはデータベースソフトウェアをマニュアルで更新すると、誤ってアプリケーションのダウンタイムが発生する可能性があります。問題の特定と修正のため、すべての変更をチェックするのに何時間も費やすことがあります。

次の表では、オンプレミスのデータベース、Amazon EC2、Amazon RDS の管理モデルを比較します。

機能	オンプレミスの管理	Amazon EC2 の管理	Amazon RDS の管理
アプリケーションの最適化	カスタマー	カスタマー	カスタマー
スケーリング	カスタマー	カスタマー	AWS
高可用性	カスタマー	カスタマー	AWS
データベースバックアップ	カスタマー	カスタマー	AWS
データベースソフトウェアのパッチ適用	カスタマー	カスタマー	AWS
データベースソフトウェアのインストール	カスタマー	カスタマー	AWS
オペレーティングシステム (OS) のパッチ適用	カスタマー	カスタマー	AWS
OS インストール	カスタマー	カスタマー	AWS
サーバーのメンテナンス	カスタマー	AWS	AWS
ハードウェアライフサイクル	カスタマー	AWS	AWS
電力、ネットワーク、冷却	お客様	AWS	AWS

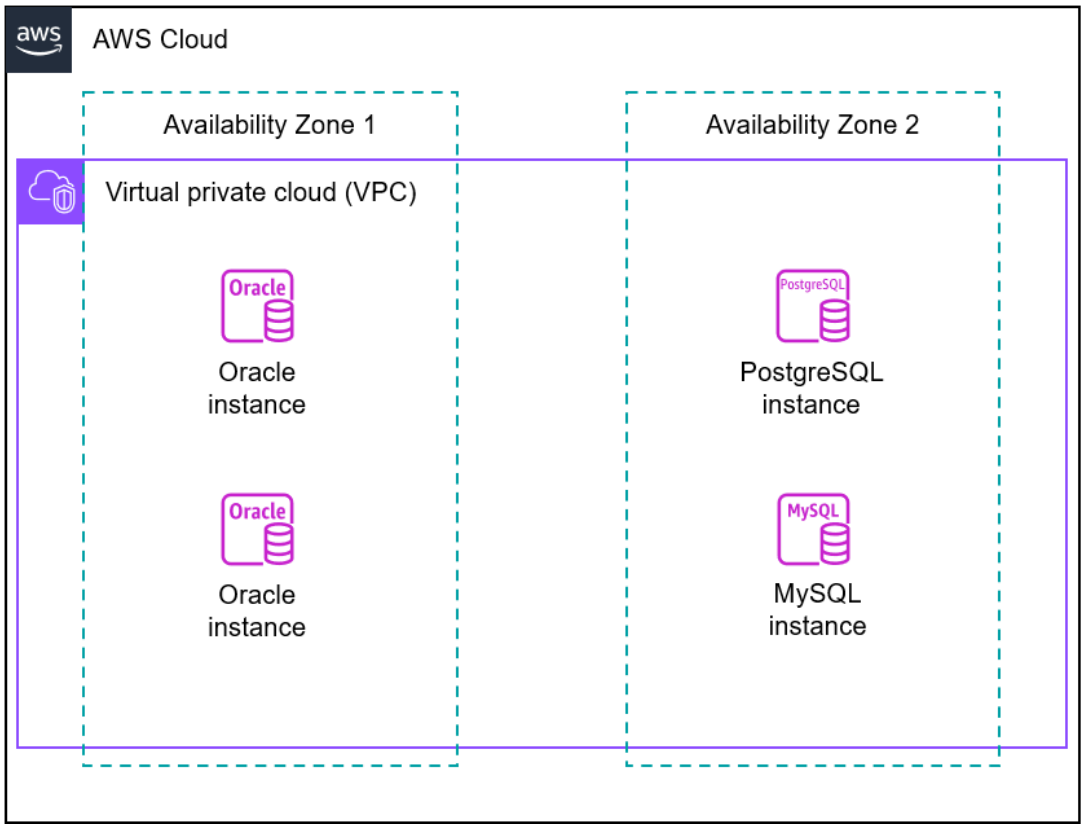
Amazon RDS 責任共有モデル

Amazon RDS は、DB インスタンスと DB クラスターのソフトウェアコンポーネントとインフラストラクチャをホストする責任があります。ユーザーはクエリチューニングに責任があります。これは、SQL クエリを調整してパフォーマンスを向上させるプロセスです。クエリのパフォーマンスは、データベースの設計、データサイズ、データ分布、アプリケーションのワークロード、クエリパターンに大きく依存しますが、これらは大きく異なる可能性があります。モニタリングとチュー

ニングは、RDS データベースごとに高度に個別化されたプロセスです。Amazon RDS Performance Insights やその他のツールを使用して、問題のあるクエリを特定できます。

Amazon RDS DB インスタンス

DB インスタンスは AWS クラウド 内の独立したデータベース環境です。Amazon RDS の基本的な構成要素は DB インスタンスです。DB インスタンスには、ユーザーが作成した1つ以上のデータベースを含めることができます。次の図は、2 つのアベイラビリティーゾーンを含む 仮想プライベートクラウド (VPC) を示しており、各 AZ には 2 つの DB インスタンスが含まれています。



DB インスタンスには、スタンドアロンデータベースインスタンスで使用するものと同じツールおよびアプリケーションを使用してアクセスすることができます。AWS Command Line Interface (AWS CLI)、Amazon RDS API、または AWS マネジメントコンソール を使用して、DB インスタンスを作成および変更することができます。

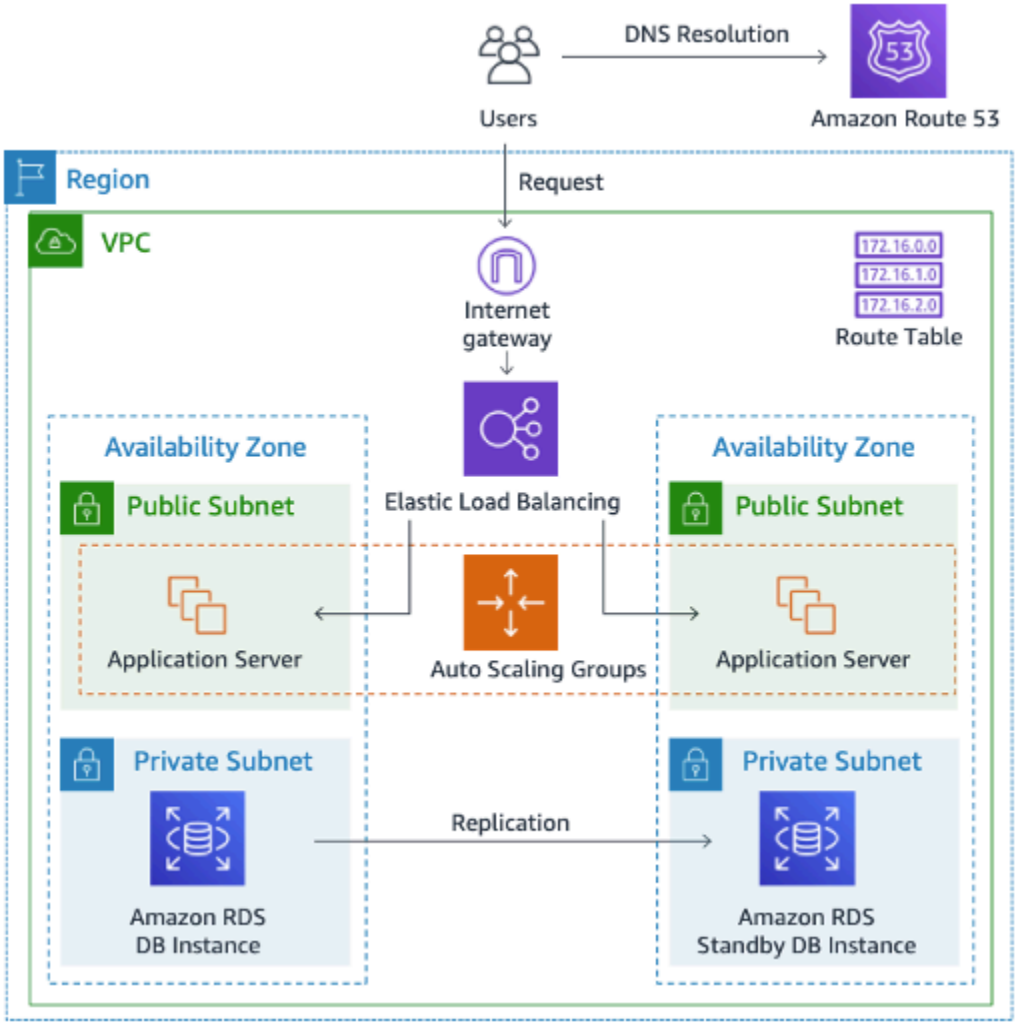
トピック

- Amazon RDS アプリケーションアーキテクチャの例
- DB エンジン
- DB インスタンスクラス

- DB インスタンスストレージ
- Amazon Virtual Private Cloud (Amazon VPC) での DV インスタンス

Amazon RDS アプリケーションアーキテクチャの例

次の図は、データベースストレージに Amazon RDS DB インスタンスを使用する動的ウェブサイトの一般的なユースケースを示しています。



上記のアーキテクチャの主なコンポーネントは次のとおりです。

Elastic Load Balancing

AWS は、Elastic Load Balancing を介してユーザートラフィックをルーティングします。ロードバランサーは、ワークロードを仮想サーバーなど複数のコンピューティングリソース間に分散さ

せます。このサンプルユースケースでは、Elastic ロードバランサーはクライアントリクエストをアプリケーションサーバーに転送します。

アプリケーションサーバー

アプリケーションサーバーは RDS DB インスタンスとやり取りします。AWS のアプリケーションサーバーは通常、スケーラブルなコンピューティング容量を提供する EC2 インスタンスでホストされます。アプリケーションサーバーは、同じ仮想プライベートクラウド (VPC) 内の異なるアベイラビリティーゾーン (AZ) を持つパブリックサブネットに存在します。

RDS DB インスタンス

EC2 アプリケーションサーバーは RDS DB インスタンスとやり取りします。DB インスタンスは、同じ仮想プライベートクラウド(VPC) 内の異なるアベイラビリティゾーン (AZ) 内にあるプライベートサブネットに存在します。サブネットはプライベートであるため、インターネットからのリクエストは許可されません。

プライマリ DB インスタンスは、リードレプリカと呼ばれる別の DB インスタンスにレプリケートします。両方の DB インスタンスは VPC 内のプライベートサブネットにあります。つまり、インターネットユーザーが直接アクセスすることはできません。

DB エンジン

DB エンジンとは、DB インスタンスで実行される特定のリレーショナルデータベースソフトウェアです。Amazon RDS は、次のデータベースエンジンをサポートします。

- IBM Db2

  詳細については、「Amazon RDS for Db2」を参照してください。
- MariaDB

  詳細については、「Amazon RDS for MariaDB」を参照してください。
- Microsoft SQL Server

  詳細については、「Amazon RDS for Microsoft SQL Server」を参照してください。
- MySQL

  詳細については、「Amazon RDS for MySQL」を参照してください。
- Oracle Database

詳細については、「Amazon RDS for Oracle」を参照してください。

- PostgreSQL

  詳細については、「Amazon RDS for PostgreSQL」を参照してください。

各 DB エンジンには、独自のサポートされている機能があり、DB エンジンの各バージョンに固有の機能が含まれている可能性があります。Amazon RDS の機能のサポートは、AWS リージョン と各 DB エンジンのバージョンによって異なります。さまざまなエンジンバージョンとリージョンでの機能のサポートを確認するには、「AWS リージョン と DB エンジンにより Amazon RDS でサポートされている機能」を参照してください。

さらに、各 DB エンジンは、DB パラメータグループに一連のパラメータを保有し、これにより管理するデータベースの動作を制御します。パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

DB インスタンスクラス

DB インスタンスクラスによって DB インスタンスのコンピューティングとメモリの容量を決定します。DB インスタンスクラスは、DB インスタンスクラスタイプとサイズの両方で構成されます。Amazon RDS では、次のインスタンスクラスタイプがサポートされています。アスタリスク (*) は世代、オプションの属性、サイズを表します。

- 凡用 – db.m*
- メモリ最適化 – db.z*、db.x*、db.r*
- コンピューティングの最適化 – db.c*
- バースト可能パフォーマンス – db.t*

インスタンスクラスごとに異なるコンピューティング、メモリ、ストレージ昨日が提供されます。例えば、db.m7g は AWS Graviton3 プロセッサを搭載した第 7 世代の汎用 DB インスタンスクラスのタイプです。DB インスタンスを作成するときは、db.m7g.2xlarge などの DB インスタンスクラスを指定します。ここで 2xlarge はサイズです。各インスタンスクラスのハードウェア仕様の詳細については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

お客様の要件に最も合う DB インスタンスクラスを選択できます。要件が時間の経過とともに変化する場合は、DB インスタンスクラスを変更できます。例えば、db.m7g.2xlarge インスタンスを db.m7g.4xlarge にスケールアップできます。詳細については、「 DB インスタンスクラス」を参照してください。

> **Note**
>
> DB インスタンスクラスの料金情報については、Amazon RDS 製品ページの料金表セクションを参照してください。

DB インスタンスストレージ

Amazon EBS は、実行中のインスタンスにアタッチできる、堅牢なブロックレベルのストレージボリュームを提供します。DB インスタンスストレージには、次のタイプがあります。

- 汎用 (SSD)

  この費用対効果の高いストレージタイプは、中規模の DB インスタンスで実行される幅広いワークロードに最適です。汎用ストレージは、開発およびテスト環境に適しています。

- プロビジョンド IOPS (PIOPS)

  このストレージタイプは、低 I/O レイテンシーおよび一貫した I/O スループットを必要とする I/O 集約型ワークロード、特にデータベースワークロードのニーズを満たすように設計されています。プロビジョンド IOPS ストレージは本稼働環境に最適です。

- マグネティック (廃止)

  マグネティックストレージは廃止され、新しい DB インスタンスでは提供されなくなります。2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。新しいストレージが必要な場合は、汎用 SSD またはプロビジョンド IOPS SSD を使用してください。

ストレージタイプは、パフォーマンス特性と価格に違いがあります。データベースの要件に応じてストレージのパフォーマンスとコストを調整できます。

各 DB インスタンスは、サポートするストレージタイプやデータベースエンジンによって最小/最大ストレージ要件が異なります。データベースの増加に対応できるように、十分なストレージを確保しておくことが大切です。また、十分なストレージがあると、DB エンジンの機能がコンテンツやログエントリを書き込むスペースが確保されます。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

Amazon Virtual Private Cloud (Amazon VPC) での DV インスタンス

Amazon Virtual Private Cloud (Amazon VPC) サービスを使用して、virtual private cloud (VPC) 上の DB インスタンスを実行できます。VPC を使用する場合、仮想ネットワーキング環境を制御できます。独自の IP アドレスの範囲を選択し、サブネットを作成してルーティングおよびアクセス制御リストを設定できます。

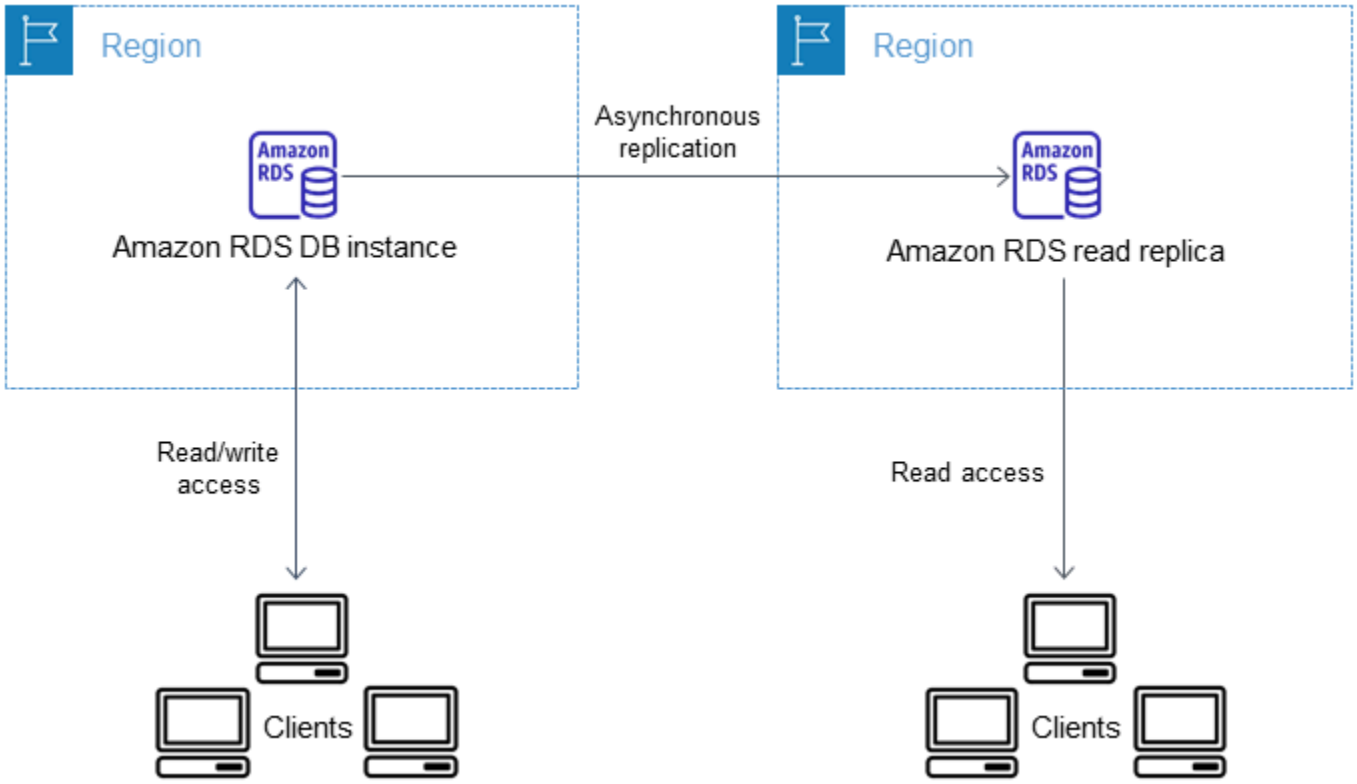
VPC で実行していてもいなくても、Amazon RDS の基本機能には違いはありません。Amazon RDS では、バックアップ、ソフトウェアパッチ、自動的な障害検出、および復旧を管理します。VPC で DB インスタンスを実行するために、追加料金はかかりません。RDS で Amazon VPC を使用する方法については、「Amazon VPC と Amazon RDS」を参照してください。

Amazon RDS は Network Time Protocol (NTP) を使用して DB インスタンスの時刻を同期します。

AWS リージョンおよびアベイラビリティーゾーン

Amazon クラウドコンピューティングリソースは、世界各地 (例えば、北米、ヨーロッパ、アジア) の高可用性のデータセンター施設に収容されています。各データセンターの場所は、AWS リージョンと呼ばれます。Amazon RDS を使用すると、複数のリージョンに DB インスタンスを作成できます。

次のシナリオでは、あるリージョンの RDS DB インスタンスが、別のリージョンのスタンバイ DB インスタンスに非同期的にレプリケートしています。一方のリージョンが使用できなくなっても、もう一方のリージョンのインスタンスは引き続き使用できます。



アベイラビリティーゾーン

各 AWS リージョンは、アベイラビリティーゾーンまたは AZ と呼ばれる複数の区切られた場所で構成されています。各アベイラビリティーゾーンは、他のアベイラビリティーゾーンの障害から分離されるように設計されています。アベイラビリティーゾーンは、同じ AWS リージョン内の他のアベイラビリティーゾーンに低価格かつ低レイテンシーのネットワーク接続を提供します。個別のアベイラビリティーゾーンで DB インスタンスを起動することにより、1 つの場所で発生した障害からアプリケーションを保護できます。詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

マルチ AZ 配置

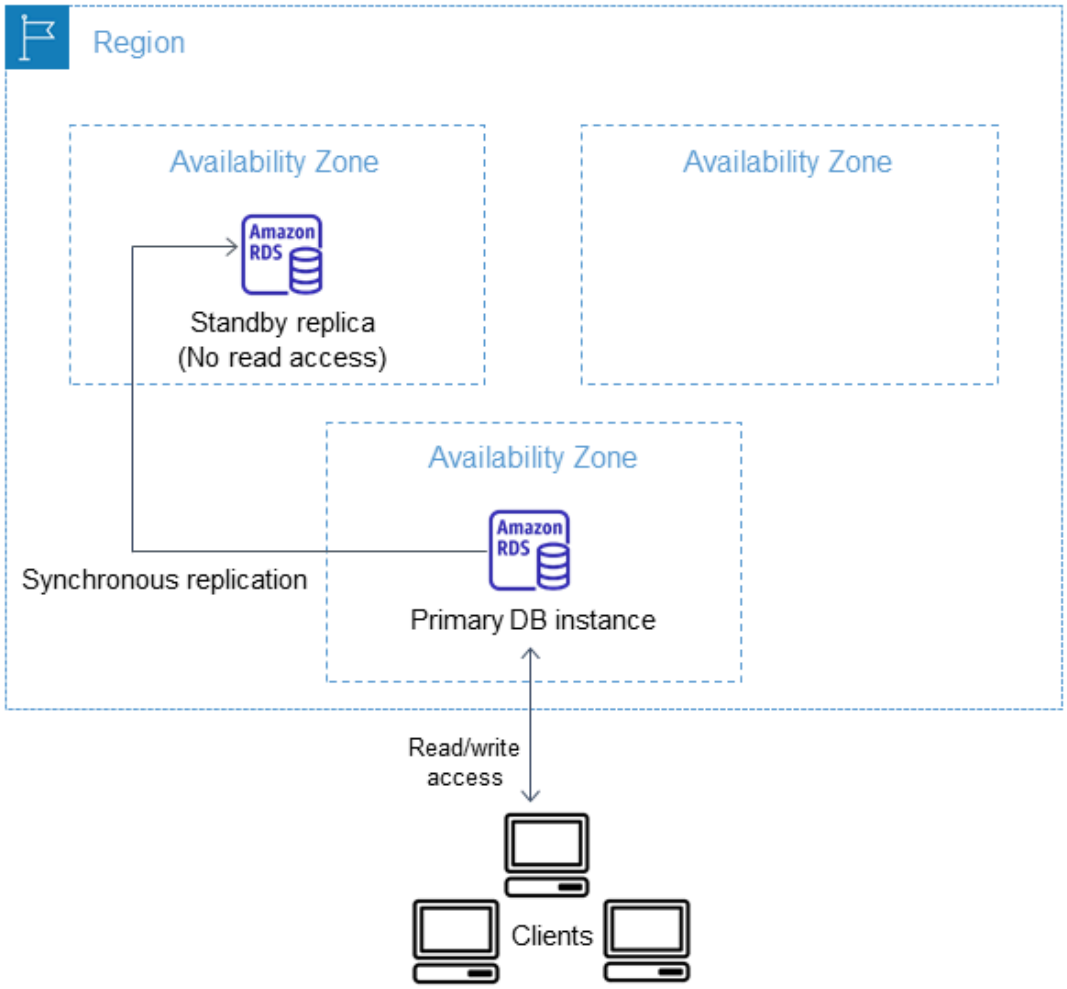
オプションで、マルチ AZ 配置と呼ばれる複数のアベイラビリティゾーンの DB インスタンスを実行できます。このオプションを選択すると、Amazon が 1 つまたは複数のセカンダリスタンバイ DB インスタンスを別の AZ で自動的にプロビジョニングして管理します。プライマリ DB インスタンスは、アベイラビリティーゾーン間で各セカンダリ DB インスタンスにレプリケートされます。

マルチ AZ 配置には、次の利点があります。

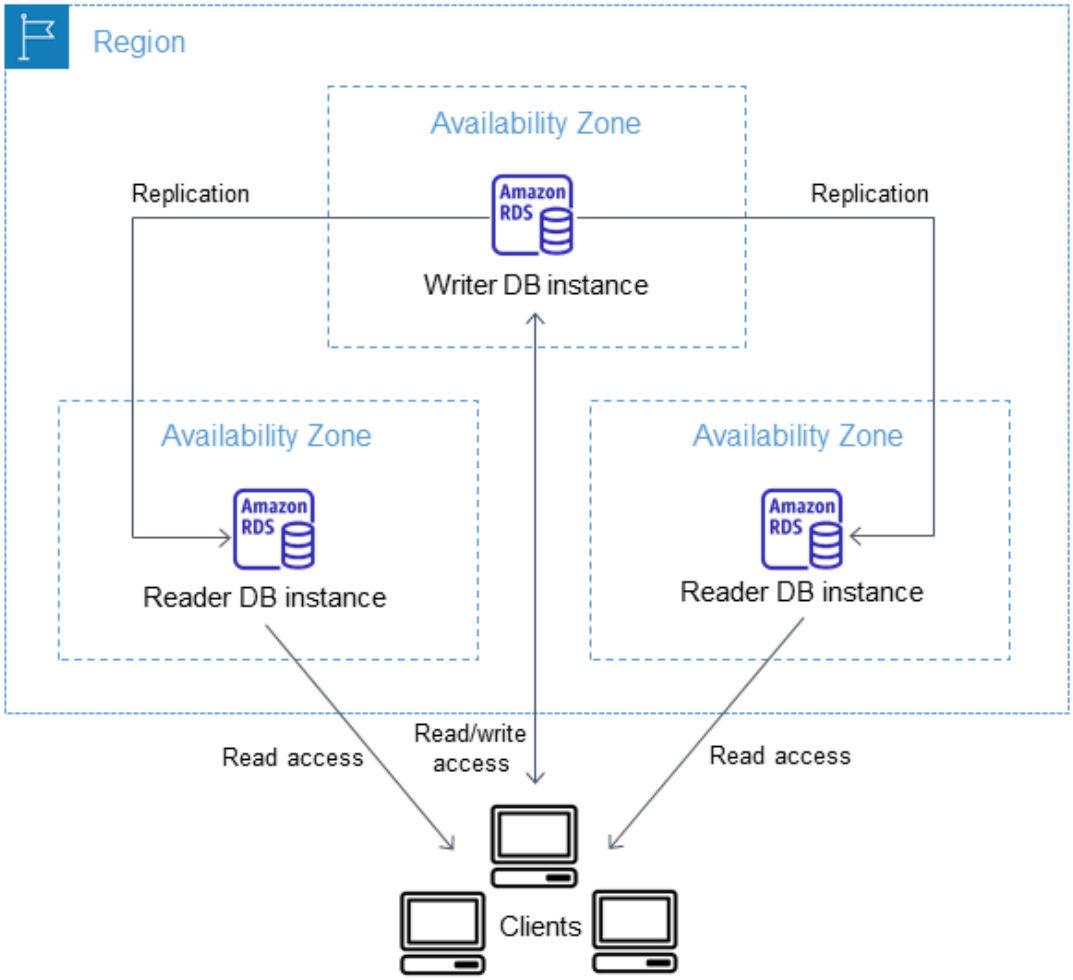
- データの冗長性とフェイルオーバーサポートの提供

- I/O フリーズの排除
- システムバックアップ中のレイテンシースパイクの最小化
- セカンダリ DB インスタンスでの読み込みトラフィックの処理 (マルチ AZ DB クラスターのデプロイのみ)

下の図は、マルチ AZ DB インスタンスのデプロイを表しており、Amazon RDS は、異なるアベイラビリティーゾーンで同期スタンバイレプリカを自動的にプロビジョンおよび維持します。レプリカデータベースは読み込みトラフィックを処理しません。



下の図は、マルチ AZ DB クラスターのデプロイを表しており、同じ AWS リージョンの 3 つの別々のアベイラビリティーゾーンに 1 つのライター DB インスタンスと 2 つのリーダー DB インスタンスがあります。3 つの DB インスタンスはすべて読み込みトラフィックを処理できます。



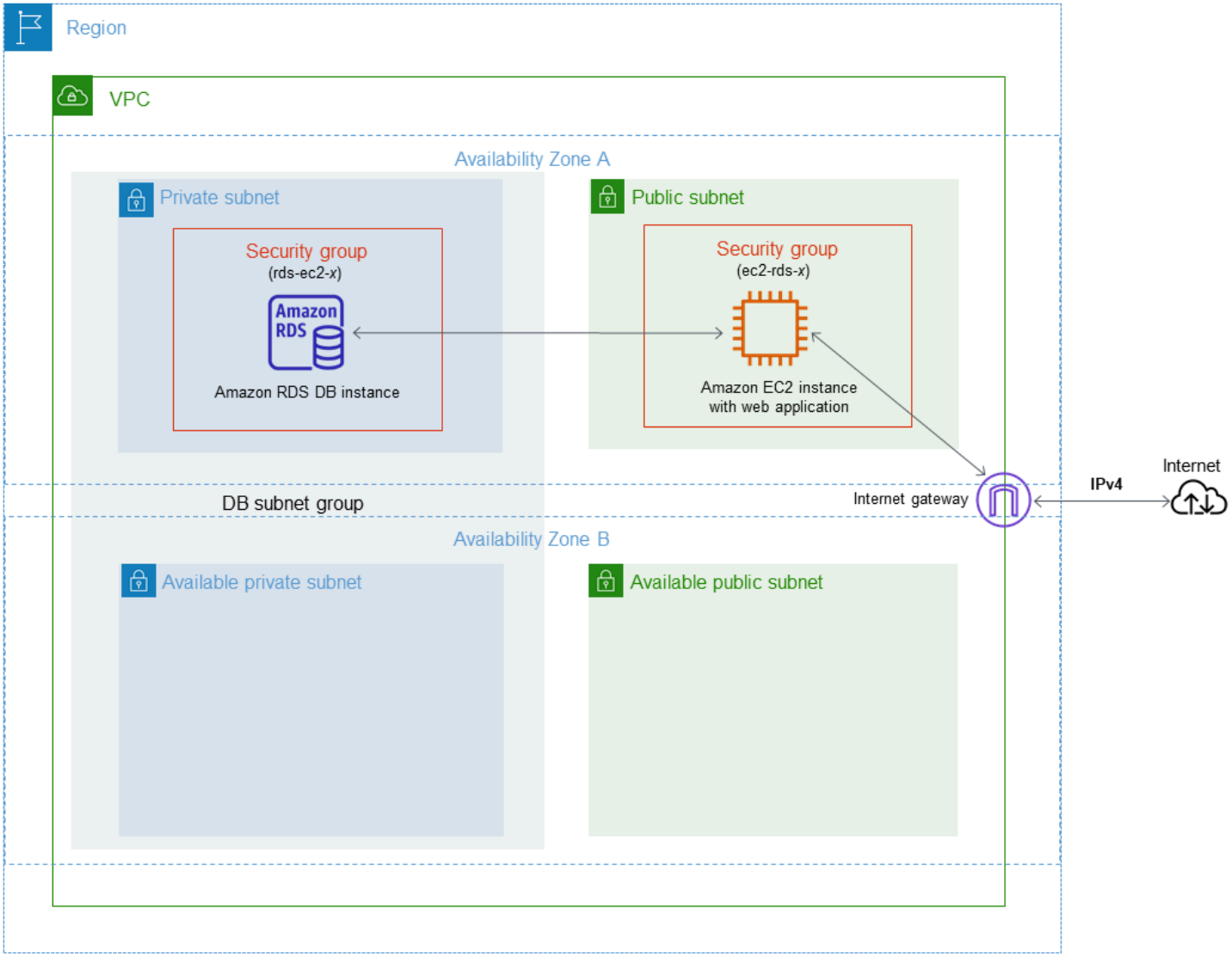
詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

セキュリティグループによるアクセスコントロール

セキュリティグループは、指定した IP アドレス範囲または Amazon EC2 インスタンスへのアクセスを許可することで、DB インスタンスへのアクセスを制御します。セキュリティグループは 1 つ以上の DB インスタンスに関連付けることができます。

VPC 内の DB インスタンスの一般的な用途は、同じ VPC 内のアプリケーションサーバーとデータを共有することです。以下の例では、VPC セキュリティグループ `ec2-rds-x` を使用して、クライアントアプリケーションの IP アドレスをソースとして使用するというインバウンドルールを定義します。アプリケーションサーバーはこのセキュリティグループに属します。`rds-ec2-x` という名前の 2 番目のセキュリティグループは、ソースとして `ec2-rds-x` を指定し、RDS DB インスタンスにアタッチします。セキュリティグループのルールによると、クライアントアプリケーションは DB イ

ンスタンスに直接アクセスできませんが、EC2 インスタンスは DB インスタンスにアクセスできます。



セキュリティグループの詳細については、Amazon RDS でのセキュリティ を参照してください。

Amazon RDS のモニタリング

モニタリングは、Amazon RDS およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。AWS には、Amazon RDS を監視したり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするためのモニタリングツールが用意されています。

さまざまな自動ツールや手動ツールを使用して、DB インスタンスのパフォーマンスとヘルスを追跡できます。

Amazon RDS インスタンスのステータスと推奨事項

Amazon RDS コンソール、AWS CLI、RDS API を使用して、インスタンスの現在のステータスに関する詳細を表示します。また、DB インスタンス、リードレプリカ、DB パラメータグループなどのデータベースリソースに対して自動化された推奨事項に対応できます。詳細については、「Amazon RDS の推奨事項」を参照してください。

Amazon RDS の Amazon CloudWatch メトリクス

Amazon CloudWatch サービスを使用して、DB インスタンスのパフォーマンスとヘルス状態をモニタリングできます。CloudWatch のパフォーマンスチャートは、Amazon RDS コンソールに表示されます。Amazon RDS は、アクティブな各データベースのメトリクスを 1 分ごとに CloudWatch に自動送信します。CloudWatch の Amazon RDS メトリクスに対する追加料金は発生しません。

Amazon CloudWatch アラームを使用すると、特定の期間にわたって 1 つの Amazon RDS メトリクスを監視できます。そのため、設定したしきい値に関連するメトリクスの値に基づいて、1 つ以上のアクションを実行できます。詳細については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

Amazon RDS Performance Insights とオペレーティングシステムのモニタリング

Performance Insights は、データベースの負荷を評価し、アクションを実行するタイミングと場所を判断します。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。Amazon RDS 拡張モニタリングは、オペレーティングシステムのメトリクスをリアルタイムで参照します。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

統合 AWS のサービス

Amazon RDS は、Amazon EventBridge、Amazon CloudWatch Logs、および Amazon DevOps Guru と統合されています。詳細については、「Amazon RDS インスタンスでのメトリクスのモニタリング」を参照してください。

Amazon RDS のユーザーインターフェイス

Amazon RDS は複数の方法でやり取りできます。

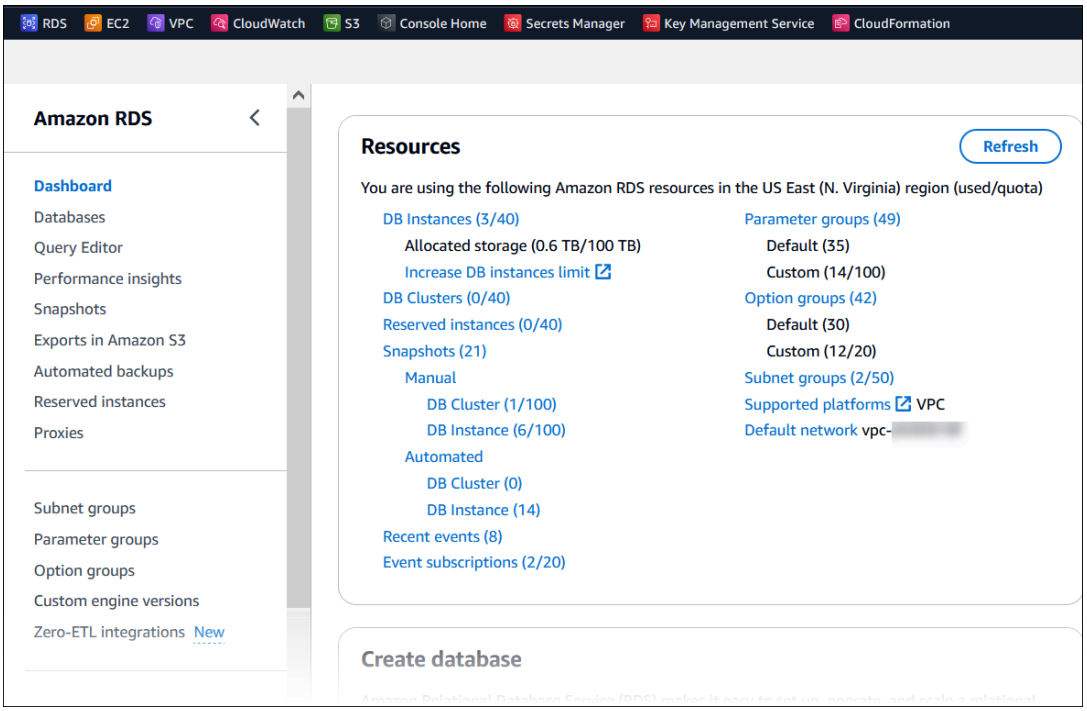
トピック

- AWS マネジメントコンソール
- コマンドラインインターフェイス

- Amazon RDS API

AWS マネジメントコンソール

AWS マネジメントコンソール はシンプルなウェブベースのユーザーインターフェイスです。プログラミングなしでコンソールから DB インスタンスを管理できます。Amazon RDS コンソールにアクセスするには、AWS マネジメントコンソールにサインインして Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。



コマンドラインインターフェイス

AWS Command Line Interface(AWS CLI) を使用して、Amazon RDS API にインタラクティブにアクセスできます。AWS CLI をインストールするには、「AWS コマンドラインインターフェイスのインストール」を参照してください。RDS 用に AWS CLI の使用をスタートするには、「Amazon RDS の AWS Command Line Interface リファレンス」を参照してください。

Amazon RDS API

開発者は、API を使用して、プログラムから Amazon RDS にアクセスできます。詳細については、「Amazon RDS API リファレンス」を参照してください。

アプリケーション開発には、AWS ソフトウェアデプロイキット (SDK) のいずれかを使用することをお勧めします。AWS SDK が、認証、再試行ロジック、エラー処理などの低レベルの詳細な処理を実

行するため、ユーザーはアプリケーションのロジックに専念することができます。AWSSDK は、さまざまな言語で利用可能です。詳細については、Amazon Web Services のツールを参照してください。

AWS では、より簡単に使用をスタートできるように、ライブラリ、サンプルコード、チュートリアルなどのリソースを提供しています。詳細については、「サンプルコードとライブラリ」を参照してください。

Amazon RDS の課金方法

Amazon RDS を使用する場合、オンデマンド DB インスタンスを使用するかリザーブド DB インスタンスを使用するかを選択できます。詳細については、「Amazon RDS DB インスタンスの請求」を参照してください。

Amazon RDS の料金情報については、Amazon RDS の製品ページを参照してください。

Amazon RDS の AWS 無料利用枠

Amazon RDS の AWS 無料利用枠 は、次のエンジンと DB インスタンスクラスと共に使用できます。

- エンジンタイプ – MariaDB、MySQL、PostgreSQL、または SQL Server Express Edition
- DB インスタンスクラス
  - t3.micro – すべてのエンジンタイプ用
  - t4g.micro – SQL Server Express Edition を除くすべてのエンジンタイプ用

Amazon RDS の AWS 無料利用枠 では、次の機能、リソース、およびアクションを使用できません。

- Amazon RDS Custom エンジンオプション
- リザーブド DB インスタンス
- シングル AZ 以外のデプロイオプション
- スナップショットの移行
- クエリエディタ
- Aurora リードレプリカの作成

これらの制限は、AWS 無料利用枠 プログラムの進化に伴って変更される可能性があります。AWS 無料利用枠 についての詳細は、「AWS 無料利用枠 で AWS を試す」を参照してください。Amazon RDS 無料利用枠の詳細については、「Amazon RDS 無料利用枠」を参照してください。

2025 年 7 月 17 日より前に利用を開始した AWS 無料利用枠 のお客様は、12 か月間の無料利用の有効期限が切れるまで Amazon RDS 無料利用枠を引き続き使用できます。この期間中は、次の使用資格が維持されます。

- Amazon RDS で MySQL、MariaDB、または PostgreSQL を実行するシングル AZ db.t3.micro および db.t4g.micro インスタンスクラスの毎月 750 時間
- Amazon RDS で SQL Server Express Edition を実行する db.t3.micro インスタンスクラスの毎月 750 時間

2025 年 7 月 17 日以降、AWS で作成した新しい AWS 無料利用枠 リソースには新しい AWS 無料利用枠 サービスが適用されます。

次のステップ

前のセクションでは、RDS が提供する基本的なインフラストラクチャのコンポーネントを紹介しました。次に実行すべきことは以下のとおりです。

スタート方法

Amazon RDS のスタート方法 の手順を使用して DB インスタンスを作成します。

データベースエンジンに固有のトピック

以下のセクションでは、DB エンジンごとに固有の情報を確認することができます。

- Amazon RDS for Db2
- Amazon RDS for MariaDB
- Amazon RDS for Microsoft SQL Server
- Amazon RDS for MySQL
- Amazon RDS for Oracle
- Amazon RDS for PostgreSQL

Amazon RDS DB インスタンス

DB インスタンスはクラウドで実行される独立したデータベース環境です。これは、Amazon RDS の基本的な構成要素です。DB インスタンスには、ユーザーが作成した複数のデータベースを含めることができ、スタンドアロンデータベースインスタンスにアクセスする場合と同じクライアントツールやアプリケーションを使用してアクセスできます。DB インスタンスは、AWS コマンドラインツール、Amazon RDS API オペレーション、または AWS マネジメントコンソール を使用して簡単に作成および変更できます。

> **Note**
>
> Amazon RDS では、任意の標準 SQL クライアントアプリケーションによるデータベースへのアクセスがサポートされています。Amazon RDS では、RDS Custom を使用する場合を除き、直接ホストアクセスは許可されません。

最大 40 個の Amazon RDS DB インスタンスを持つことができますが、以下の制限があります。

- 「ライセンス込み」のモデルでは、各 SQL Server のエディション (Enterprise、Standard、Web、および Express) ごとにインスタンスをそれぞれ最大 10 使用することができます。
- 「ライセンス込み」モデルに基づく Oracle 向けの 10
- 「Bring-Your-Own-License (BYOL)」ライセンスモデルの Db2 の場合は 40
- MySQL、MariaDB、または PostgreSQL では、40 使用できます。
- 「Bring-Your-Own-License (BYOL)」 モデルの Oracle では、40 使用できます。

> **Note**
>
> アプリケーションでそれ以上の DB インスタンスが必要な場合、このフォームを使用して追加の DB インスタンスをリクエストできます。

各 DB インスタンスには、DB インスタンス識別子があります。このお客様が指定する名前により、Amazon RDS API および AWS CLI コマンドを操作する際、DB インスタンスが一意に識別されます。DB インスタンス識別子は、AWS リージョン内でユーザー別に一意にする必要があります。

DB インスタンス識別子は、RDS によってインスタンスに割り当てられた DNS ホスト名の一部として使用されます。例えば、db1 を DB インスタンス識別子として指定した場合、RDS は、インスタンスの DNS エンドポイントを自動的に割り当てます。サンプルエンドポイントには、*db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com があります。この場合、*db1* はインスタンス ID です。

サンプルエンドポイント *db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com では、文字列 *abcdefghijkl* は AWS リージョン と AWS アカウント の一意の識別子です。この例の識別子 *abcdefghijkl* は内部で RDS によって生成され、指定されたリージョンとアカウントの組み合わせでは変わりません。したがって、このリージョンのすべての DB インスタンスは同一の固定 ID を共有します。固定識別子の以下の特徴を考えてみましょう。

- DB インスタンスの名前を変更した場合、エンドポイントは異なりますが、固定識別子は同じです。たとえば、名前を *db1* から *renamed-db1* に変更した場合、新しいインスタンスエンドポイントは *renamed-db1*.*abcdefghijkl*.*us-east-1*.rds.amazonaws.com です。
- 同じ DB インスタンス識別子を持つ DB インスタンスを削除して再作成した場合、エンドポイントは同じになります。
- 同じアカウントを使用して別のリージョンに DB インスタンスを作成した場合、内部で生成される識別子は異なります。これは、*db2*.*mnopqrstuvwx*.*us-west-1*.rds.amazonaws.com にあるようにリージョンが異なるためです。

各 DB インスタンスでは、データベースエンジンがサポートされています。Amazon RDS で現在サポートされているデータベースエンジンは、Db2、MySQL、MariaDB、PostgreSQL、Oracle、Microsoft SQL Server、および Amazon Aurora です。

DB インスタンスの作成時、一部のデータベースエンジンではデータベース名を指定する必要があります。DB インスタンスは、複数のデータベース、または複数のスキーマを使用する 1 つの Oracle データベースをホストすることができます。データベース名の値は、データベースエンジンごとに異なります。

- Db2 データベースエンジンの場合、データベース名は、DB インスタンスでホストされるデータベースの名前です。このフィールドはオプションです。後から rdsadmin.create_database ストアドプロシージャを呼び出して、データベースを作成できます。詳細については、「the section called “データベースを作成する”」を参照してください。

- MySQL および MariaDB データベースエンジンの場合、データベース名は、DB インスタンスでホストされるデータベースの名前です。同じ DB インスタンスによってホストされているデータベースは、そのインスタンス内で名前が一意である必要があります。
- Oracle データベースエンジンの場合、データベース名は、Oracle RDS インスタンスへの接続時に渡す必要のある ORACLE_SID の値を設定するために使用されます。
- Microsoft SQL Server データベースエンジンの場合、データベース名はサポートされたパラメータではありません。
- PostgreSQL データベースエンジンの場合、データベース名は、DB インスタンスでホストされるデータベースの名前です。データベース名は、DB インスタンスを作成する際は必須ではありません。同じ DB インスタンスによってホストされているデータベースは、そのインスタンス内で名前が一意である必要があります。

Amazon RDS は、作成プロセスの一環として、DB インスタンスのマスターユーザーアカウントを作成します。このマスターユーザーには、データベースを作成し、作成したテーブルに対して作成、削除、選択、更新、挿入の各オペレーションを実行するアクセス許可があります。DB インスタンスを作成する際、マスターユーザーパスワードを設定する必要があります。このパスワードは、AWS CLI、Amazon RDS API オペレーション、または AWS マネジメントコンソール を使用して後でいつでも変更できます。マスターユーザーパスワードを変更して、標準 SQL コマンドを使用してユーザーを管理できます。

> **Note**
>
> このガイドは、Aurora Amazon RDS 以外のデータベースエンジンについて説明するようになりました。Amazon Aurora の使用については、Amazon Aurora ユーザーガイドを参照してください。

DB インスタンスクラス

DB インスタンスクラスによって、Amazon RDS DB インスタンスの計算とメモリの容量を決定します。必要な DB インスタンスクラスは、処理能力とメモリの要件によって異なります。

DB インスタンスクラスは、DB インスタンスクラスタイプとサイズの両方で構成されます。例えば、db.r6g は AWS Graviton2 プロセッサを搭載したメモリ最適化DB インスタンスクラスタイプです。db.r6g インスタンスクラスタイプ内の db.r6g.2xlarge は DB インスタンスクラスです。このクラスのサイズは 2xlarge です。

インスタンスクラスの料金の詳細については、「Amazon RDS の料金」を参照してください。

DB インスタンスクラスタイプ、サポートされている DB エンジン、サポートされている AWS リージョン、DB インスタンスクラスの変更、RDS for Oracle のプロセッサの設定、または DB インスタンスクラスのハードウェア仕様の詳細については、以下のセクションを参照してください。

トピック

- DB インスタンスクラスタイプ
- DB インスタンスクラスでサポートされている DB エンジン
- AWS リージョン での DB インスタンスクラスのサポートを決定する
- DB インスタンスクラスの変更
- マルチ AZ 配置のクロスアベイラビリティーゾーン ENA Express のサポート
- RDS for Oracle で DB インスタンスクラスのプロセッサを設定する
- DB インスタンスクラスのハードウェア仕様

DB インスタンスクラスタイプ

Amazon RDS は、以下のタイプの DB インスタンスクラスをサポートしています。

- 汎用
- メモリ最適化
- コンピューティング最適化
- バースト可能パフォーマンス
- Optimized Reads

Amazon EC2 インスタンスタイプの詳細については、Amazon EC2 ドキュメントの「インスタンスタイプ」を参照してください。

汎用インスタンスクラスタイプ

以下の汎用 DB インスタンスクラスが使用可能です。

- db.m9g – AWS Graviton5 プロセッサを搭載した汎用 DB インスタンスクラス。これらのインスタンスクラスでは、幅広い汎用ワークロード向けにバランスの取れたコンピューティング、メモリ、ネットワークを提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。

  AWS Graviton5 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.m8g – AWS Graviton4 プロセッサを搭載した汎用 DB インスタンスクラス。これらのインスタンスクラスでは、幅広い汎用ワークロード向けにバランスの取れたコンピューティング、メモリ、ネットワークを提供します。第 7 世代の AWS Graviton3 ベースの M7g インスタンスと比較して、これらの新しいクラスでは、最大 3 倍の vCPU とメモリを備えたより大きなインスタンスサイズを提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。

  AWS Graviton4 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.m8i - インテル Xeon 6 プロセッサを搭載した汎用 DB インスタンスクラス。これらのインスタンスは、クラウド内の同等のインテルプロセッサの中で最高のパフォーマンスと最速のメモリ帯域幅を提供します。M8i は、前世代のインテルベースのインスタンスと比較して、料金パフォーマンスが最大 15% 向上し、メモリ帯域幅が 2.5 倍増加します。
- db.m7i – 第 4 世代インテル Xeon スケーラブルプロセッサを搭載した汎用 DB インスタンスクラス。db.m7i インスタンスは SAP 認定を受けており、より大きなインスタンスサイズや CPU の高い連続使用率が求められるエンタープライズアプリケーションをサポートするのに最適です。これらのインスタンスクラスでは、幅広い汎用ワークロード向けにバランスの取れたコンピューティング、メモリ、ネットワークを提供します。このインスタンスクラスタイプは、最大 40,000 Mbps の EBS 帯域幅と最大 50 Gbps のネットワーク帯域幅を提供します。db.m7i インスタンスは、db.m6i インスタンスと比較して、最大 15% 優れた価格パフォーマンスを提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。

- db.m7g – AWS Graviton3 プロセッサを搭載した汎用 DB インスタンスクラス。これらのインスタンスクラスでは、幅広い汎用ワークロード向けにバランスの取れたコンピューティング、メモリ、ネットワークを提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。

  AWS Graviton3 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.m6g – AWS Graviton2 プロセッサを搭載した汎用 DB インスタンスクラス。これらのインスタンスでは、幅広い汎用ワークロード向けにバランスの取れたコンピューティング、メモリ、ネットワークを提供します。db.m6gd インスタンスクラスは、高速で低レイテンシーのローカルストレージを必要とするアプリケーションのために、ローカル NVMe ベースの SSD ブロックレベルストレージを備えています。

  AWS Graviton2 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.m6i — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス。これらのインスタンスは SAP 認定を受けており、エンタープライズアプリケーションをサポートするバックエンドサーバー、ゲームサーバー、キャッシュフリート、アプリケーション開発環境などのワークロードに最適です。db.m6id および db.m6idn インスタンスクラスは最大 7.6 TB のローカル NVMe ベースの SSD ストレージを提供し、db.m6in は EBS 専用ストレージを提供します。db.m6in クラスと db.m6idn クラスは、最大 200 Gbps のネットワーク帯域幅を提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.m5 – バランスの取れたコンピューティング、メモリ、ネットワークリソースを提供し、多くのアプリケーションに適した汎用 DB インスタンスクラス。db.m5d インスタンスクラスは、ホストサーバーに物理的に接続された NVMe ベースの SSD ストレージを提供します。db.m5 インスタンスクラスは、以前の db.m4 インスタンスクラスよりも多くのコンピューティング容量を備えています。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.m4 – 以前の db.m3 インスタンスクラスより高いコンピューティング容量を備えた汎用 DB インスタンスクラス。

  RDS for Oracle DB エンジンについては、Amazon RDS は db.m4 DB インスタンスクラスをサポートしなくなりました。以前に RDS for Oracle db.m4 DB インスタンスを作成していた場合、Amazon RDS は、それらの db.m5 DB インスタンスクラスに自動的にアップグレードします。

RDS for MariaDB、RDS for MySQL、RDS for SQL Server、および RDS for PostgreSQL DB エンジンについて、Amazon RDS は、以下のスケジュールで DB インスタンスクラスのサポート終了プロセスを開始しました。このインスタンスクラスを使用しているすべての RDS DB インスタンスについては、できるだけ早期により新しい世代の DB インスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
この日付から、Amazon RDS は db.m4 を使用して新しい世代の db.m5 インスタンスクラスへのインスタンスの自動アップグレードを開始しました。db.m4 インスタンスクラスを使用した DB インスタンスの作成はサポートされなくなりました。	2024 年 6 月 1 日
Amazon RDS は db.m4 のサポートを終了します。	2024 年 12 月 31 日

- db.m3 – 以前の db.m1 インスタンスクラスより高いコンピューティング容量を備えた汎用 DB インスタンスクラス。

  RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL DB エンジンについて、Amazon RDS は、アップグレードの推奨事項を含め、以下のスケジュールで db.m3 DB インスタンスクラスを終了するプロセスを開始しています。db.m3 DB インスタンスクラスを使用しているすべての RDS DB インスタンスは、できるだけ早期により高い世代の インスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
db.m3 DB インスタンスクラスを使用する RDS DB インスタンスを作成できなくなりました。	現在
Amazon RDS は、db.m3 DB インスタンスクラスを使用する RDS DB インスタンスの同等の db.m5 DB インスタンスクラスへの自動アップグレードを開始しました。	2023 年 2 月 1 日

メモリ最適化インスタンスクラスタイプ

メモリ最適化 Z ファミリーは、以下のインスタンスクラスをサポートします。

- db.z1d - メモリを大量に使用するアプリケーション用に最適化されているインスタンスクラス。これらのインスタンスクラスでは、優れたコンピューティング性能と大きなメモリフットプリントの両方を提供します。高周波 z1d インスタンスでは、最大 4.0 GHz の持続的な全コア周波数を提供します。

メモリ最適化 X ファミリーは、以下のインスタンスクラスをサポートします。

- db.x2g - メモリを大量に消費するアプリケーション用に最適化され、AWS Graviton2 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、メモリの GiB あたりのコストを削減します。

  AWS Graviton2 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.x2i – メモリを大量に使用するアプリケーション用に最適化されているインスタンスクラス。db.x2iedn および db.x2idn インスタンスクラスタイプは、第 3 世代インテル Xeon スケーラブルプロセッサ (Ice Lake) を搭載しています。これらには、最大 3.8 TB のローカル NVMe SSD ストレージ、最大 100 Gbps のネットワーク帯域幅、および最大 4 TiB (db.x2iden) または 2 TiB (db.x2idn) のメモリが含まれています。db.x2iezn タイプは、最大 4.5 GHz の全コアターボ周波数と最大 1.5 TiB のメモリを備えた第 2 世代インテル Xeon スケーラブルプロセッサ (Cascade Lake) と、専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を搭載しています。
- db.x1 - メモリを大量に使用するアプリケーション用に最適化されているインスタンスクラス。これらのインスタンスクラスは、すべての DB インスタンスクラスの中で、RAM の GiB あたりの料金が最も低く、最大 1,952 GiB の DRAM ベースのインスタンスメモリを提供します。db.x1e インスタンスクラスタイプは、最大 3,904 GiB の DRAM ベースのインスタンスメモリを提供します。

メモリ最適化 R ファミリーは、次のインスタンスクラスタイプをサポートします。

- db.r8g - AWS Graviton4 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、MySQL や PostgreSQL などのオープンソースデータベースでメモリ消費の高いワークロードを実行するのに最適です。これらのインスタンスは、第 7 世代の AWS Graviton3 ベースの db.r7g インスタンスの最大 3 倍の vCPU とメモリを備えた、より大きなインスタンスサイズを提供しま

す。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。

- AWS Graviton4 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.r8i – インテル Xeon 6 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、インメモリデータベース、リアルタイムのビッグデータ分析、大規模なインメモリキャッシュ、スクラッチスペースを必要とする科学計算アプリケーション、高速ローカルストレージを必要とするデータ処理アプリケーションなど、高性能なローカルストレージから恩恵を受けるメモリを大量に消費するワークロードの実行に最適です。
- db.r7g - AWS Graviton3 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、MySQL や PostgreSQL などのオープンソースデータベースでメモリ消費の高いワークロードを実行するのに最適です。

  AWS Graviton3 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.r7i — 第 4 世代インテル Xeon スケーラブルプロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは SAP 認定であり、MySQL や PostgreSQL などのオープンソースデータベースでメモリ消費の高いワークロードを実行するのに最適です。第 4 世代インテル Xeon スケーラブルプロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.r6g - AWSGraviton2 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、MySQL や PostgreSQL などのオープンソースデータベースでメモリ消費の高いワークロードを実行するのに最適です。db.r6gd タイプは、高速で低レイテンシーのローカルストレージを必要とするアプリケーションのために、ローカル NVMe ベースの SSD ブロックレベルストレージを備えています。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- AWS Graviton2 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。

- db.r6i — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは SAP 認定であり、MySQL や PostgreSQL などのオープンソースデータベースでメモリ消費の高いワークロードを実行するのに最適です。db.r6id、db.r6in、db.r6idn インスタンスクラスのメモリと vCPU の比率は 8:1 で、最大メモリは 1 TiB です。db.r6id クラスと db.r6idn クラスは、最大 7.6 TB の直接接続型 NVMe ベースの SSD ストレージを提供し、db.r6in クラスは EBS 専用ストレージを提供します。db.r6idn クラスと db.r6in クラスは、最大 200 Gbps のネットワーク帯域幅を提供します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.r5b – スループットを重視するアプリケーション向けにメモリが最適化されたインスタンスクラス。AWS Nitroシステムを搭載する db.r5b インスタンスは、最大 60 Gbps の帯域幅と 260,000 IOPSのEBSパフォーマンスを提供します。これは、EC2 上で最速のブロックストレージパフォーマンスです。
- db.r5d - 低レイテンシー、非常に高いランダム I/O パフォーマンス、および高いシーケンシャル読み取りスループットに向けて最適化されたインスタンスクラス。
- db.r5 – メモリを大量に使用するアプリケーション用に最適化されたインスタンスクラス。これらのインスタンスクラスは、ネットワーキングと のパフォーマンスを強化します。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.r4 — 以前の db.r3 インスタンスクラスよりもネットワークが改善されたインスタンスクラス。

  RDS for Oracle DB エンジンについて、Amazon RDS は、アップグレードの推奨事項を含め、以下のスケジュールで db.r4 DB インスタンスクラスの製品終了プロセスを開始しています。db.r4 DB インスタンスクラスを使用している RDS for Oracle DB インスタンスは、できるだけ早期により高い世代のインスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
db.r4 DB インスタンスクラスを使用する RDS for Oracle DB インスタンスを作成できなくなりました。	現在
Amazon RDS は、db.r4 DB インスタンスクラスを使用する RDS for Oracle DB インスタンスの同等の db.r5 DB インスタンスクラスへの自動アップグレードを開始しました。	2023 年 4 月 17 日

RDS for MariaDB、RDS for MySQL、RDS for SQL Server、および RDS for PostgreSQL DB エンジンについて、Amazon RDS は、以下のスケジュールで DB インスタンスクラスのサポート終了プロセスを開始しました。このインスタンスクラスを使用しているすべての RDS DB インスタンスについては、できるだけ早期により新しい世代の DB インスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
この日付から、Amazon RDS は db.r4 を使用して新しい世代の db.r5 インスタンスクラスへのインスタンスの自動アップグレードを開始しました。db.m4 インスタンスクラスを使用した DB インスタンスの作成はサポートされなくなりました。	2024 年 6 月 1 日
Amazon RDS は db.r4 のサポートを終了します。	2024 年 12 月 31 日

- db.r3 - メモリの最適化を提供するインスタンスクラス。

  RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL DB エンジンについて、Amazon RDS は、アップグレードの推奨事項を含め、以下のスケジュールで db.r3 DB インスタンスクラスを終了するプロセスを開始しています。db.r3 DB インスタンスクラスを使用しているすべての RDS DB インスタンスは、できるだけ早期により高い世代の インスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
db.r3 DB インスタンスクラスを使用する RDS DB インスタンスを作成できなくなりました。	現在
Amazon RDS は、db.r3 DB インスタンスクラスを使用する RDS DB インスタンスの同等の db.r5 DB インスタンスクラスへの自動アップグレードを開始しました。	2023 年 2 月 1 日

コンピューティング最適化インスタンスクラスタイプ

次のコンピューティング最適化インスタンスクラスタイプが利用できます。

- db.c6gd — 高度な計算集約型ワークロードの実行に最適なインスタンスクラス。AWS Graviton2 プロセッサを搭載したこれらのインスタンスクラスは、高速で低レイテンシーのローカルストレージを必要とするアプリケーションに、ローカル NVMe ベースの SSD ブロックレベルストレージを提供します。

  > **Note**
  >
  > c6gd インスタンスクラスは、マルチ AZ DB クラスター配置でのみサポートされます。これらは、`medium` インスタンスサイズを提供するマルチ AZ DB クラスターでサポートされる唯一のインスタンスクラスです。詳細については、「the section called “マルチ AZ DB クラスター配置”」を参照してください。

バースト可能パフォーマンスインスタンスクラスタイプ

以下のバースト可能パフォーマンス DB インスタンスクラスタイプが使用可能です。

- db.t4g – ARM ベースの AWS Graviton2 プロセッサを搭載した汎用インスタンスクラス。これらのインスタンスクラスでは、幅広いバースト汎用ワークロードに対して、以前のバーストパフォーマンス DB インスタンスクラスよりも優れた料金パフォーマンスを提供します。Amazon RDS db.t4g インスタンスは、Unlimited モードに設定されています。したがって、追加料金を支払えば、24 時間にわたり、ベースラインを超えてバーストできることになります。

  AWS Graviton2 プロセッサを搭載した DB インスタンスクラスの 1 つを使用するように DB インスタンスを変更できます。これを行うには、他の DB インスタンスを変更する場合と同じ手順を実行します。
- db.t3 – ベースラインのパフォーマンスレベルを提供しながら、CPU の最大使用率までバーストすることも可能なインスタンスクラス。db.t3 インスタンスは、Unlimited モードに設定されています。これらのインスタンスクラスには、前世代の db.t2 インスタンスクラスよりも多くのコンピューティング容量を備えています。専用ハードウェアと軽量ハイパーバイザーが組み合わされた AWS Nitro System を使用します。
- db.t2 – ベースラインのパフォーマンスレベルを提供しながら、CPU の最大使用率までバーストすることも可能なインスタンスクラス。db.t2 インスタンスは、Unlimited モードに設定されていま

す。これらのインスタンスクラスは、開発/テストサーバーなどの本稼働用以外のサーバーでのみ使用することをお勧めします。

RDS for MariaDB、RDS for MySQL、RDS for SQL Server、および RDS for PostgreSQL DB エンジンについて、Amazon RDS は、以下のスケジュールで DB インスタンスクラスのサポート終了プロセスを開始しました。このインスタンスクラスを使用しているすべての RDS DB インスタンスについては、できるだけ早期により新しい世代の DB インスタンスクラスにアップグレードすることをお勧めします。

アクションまたは推奨事項	日付
この日付から、Amazon RDS は db.t2 を使用して新しい世代の db.t3 インスタンスクラスへのインスタンスの自動アップグレードを開始しました。db.t2 インスタンスクラスを使用した DB インスタンスの作成はサポートされなくなりました。	2024 年 6 月 1 日
Amazon RDS は db. t 2 のサポートを終了します。	2024 年 12 月 31 日

> **Note**
>
> AWS Nitro System (db.m5、db.r5、db.t3) を使用する DB インスタンスクラスは、読み取りと書き込みを組み合わせたワークロードに対して調整されます。

DB インスタンスクラスのハードウェア仕様については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

Optimized Reads インスタンスクラスタイプ

次の Optimized Reads インスタンスクラスタイプが利用可能です。

- db.m8gd - AWS Graviton4 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、高速で低レイテンシーのストレージを必要とする汎用ワークロードに最適です。最大 786 GiB のメモリと、最大 11.4 TB の直接アタッチされた NVMe ベースの SSD ストレージを備えています。

- db.r8gd – Graviton4 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、メモリ負荷の高いワークロードの実行に最適で、高速で低レイテンシーのローカルストレージを必要とするアプリケーションに、ローカル NVMe ベースの SSD ブロックレベルストレージを提供します。最大 1.5 TiB のメモリと、最大 11.4 TB の直接アタッチされた NVMe ベースの SSD ストレージを備えています。
- db.r6gd – AWS Graviton2 プロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは、メモリ負荷の高いワークロードの実行に最適で、高速で低レイテンシーのローカルストレージを必要とするアプリケーションに、ローカル NVMe ベースの SSD ブロックレベルストレージを提供します。
- db.r6id – 第 3 世代 Intel Xeon スケーラブルプロセッサを搭載したインスタンスクラス。これらのインスタンスクラスは SAP 認定であり、メモリ負荷の高いワークロードに最適です。最大 1 TiB のメモリと、最大 7.6 TB の直接アタッチされた NVMe ベースの SSD ストレージを備えています。

DB インスタンスクラスでサポートされている DB エンジン

DB インスタンスクラスの DB エンジン固有の考慮事項は次のとおりです。

Db2

DB インスタンスクラスのサポートは、Db2 のバージョンとエディションによって異なります。バージョンやエディションでサポートされているインスタンスクラスについては、「Amazon RDS for Db2 インスタンスクラス」を参照してください。

Microsoft SQL Server

DB インスタンスクラスのサポートは、SQL Server のバージョンとエディションによって異なります。バージョンやエディションでサポートされているインスタンスクラスについては、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。

Oracle

DB インスタンスクラスのサポートは、Oracle Database のバージョンとエディションによって異なります。RDS for Oracle では、追加のメモリ最適化インスタンスクラスがサポートされています。これらのクラスには db.r5.*instance_size*.tpc*threads_per_core*.mem*ratio* という形式の名前があります。最適化された各クラスの vCPU 数とメモリ割り当てについては、「サポートされている RDS for Oracle DB インスタンスクラス」を参照してください。

RDS Custom

RDS Custom でサポートされている DB インスタンスクラスの詳細については、「RDS Custom for Oracle での DB インスタンスクラスのサポート」および「RDS Custom for SQL Server の DB インスタンスクラスでのサポート」を参照してください。

Amazon RDS DB エンジンごとにサポートされている Amazon RDS DB インスタンスに関する詳細を以下の表に示します。各エンジンのセルには、次のいずれかの値が含まれます。

はい

インスタンスクラスは DB エンジンのすべてのバージョンでサポートされています。

いいえ

インスタンスクラスは DB エンジンではサポートされません。

specific-versions

インスタンスクラスは、DB エンジンの、指定されたデータベースバージョンでサポートされています。

Amazon RDS では、DB エンジンのメジャーバージョンとマイナーバージョンが定期的に非推奨になります。すべての AWS リージョン が以前のエンジンバージョンをサポートしているわけではありません。現在サポートされているバージョンについては、個々の DB エンジンのトピック (Db2 バージョン、MariaDB バージョン、Microsoft SQL Server バージョン、MySQL バージョン、Oracle バージョン、PostgreSQL バージョン) を参照してください。

トピック

- 汎用インスタンスクラスでサポートされている DB エンジン
- メモリ最適化インスタンスクラスでサポートされている DB エンジン
- コンピューティング最適化インスタンスクラスでサポートされている DB エンジン
- バーストパフォーマンスインスタンスクラスでサポートされている DB エンジン
- Optimized Reads インスタンスクラスでサポートされている DB エンジン

汎用インスタンスクラスでサポートされている DB エンジン

以下の表は、汎用インスタンスクラスでサポートされているデータベースとデータベースバージョンを示しています。

db.m9g – AWS Graviton5 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m9g.4 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.2 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.1 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.1 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m9g.8x large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.4x large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.2x large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m9g.xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m9g.large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

db.m8g – AWS Graviton4 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8g.4 xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.2 xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.1 xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
						降、14.9 以降、および 13.11 以降
db.m8g.1 xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.8 large	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.4 large	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.2 large	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8g.xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m8g.large	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

db.m8i — インテル Xeon 6 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8i.96xlarge	いいえ	なし	はい	いいえ	BYO のみ、rise Editi のみ	いいえ
db.m8i.48xlarge	いいえ	なし	はい	いいえ	BYO のみ、rise Editi	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
					のみ	
db.m8i.32xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.m8i.24xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.m8i.16xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.m8i.12xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8i.8xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.m8i.4xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.m8i.2xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.m8i.xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.m8i.large	いいえ	なし	はい	いいえ	はい	いいえ
db.m8i.metal-96xl	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ
db.m8i.metal-48xl	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ

db.m8a – 第 5 世代 AMD EPYC プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8a.4 xlarge	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
db.m8a.2 xlarge	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
db.m8a.1 xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.m8a.1 xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.m8a.8 large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.m8a.4 large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.m8a.2 large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.m8a.x arge	いいえ	なし	はい	いいえ	いいえ	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m8a.large	いいえ	なし	はい	いいえ	いいえ	いいえ

db.m7i – 第 4 世代インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.m7i.48xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、Enrise Edition のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、Enrise Edition のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、Enrise Edition のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.12xlarge	はい	MariaDB 12.3、11.8、11.4、10.	はい	MySQL 8.0.32 以降	BYOL の	PostgreSQL バージョン 17.1

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
		11、10.6、10.5、および 10.4			み、Enrise Editionのみ	以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOLのみ、Enrise Editionのみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOLのみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOLのみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.m7i.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.m7i.metal-48xl	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ
db.m7i.metal-24xl	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.m7g – AWS Graviton3 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m7g.16 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.m7g.12 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.m7g.8x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.m7g.4x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m7g.2x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.m7g.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.m7g.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン

db.m6g – AWS Graviton2 プロセッサを搭載した汎用インスタンスクラス。

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6g.16 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
						ン、12.7 以降の 12 バージョン
db.m6g.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.m6g.8 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.m6g.4 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.m6g.2 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.m6g.xl arge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6g.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

db.m6gd — AWS Graviton2 プロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6gd.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン
db.m6gd.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン
db.m6gd.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m6gd.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン
db.m6gd.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン
db.m6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン
db.m6gd.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16 , 15、14 のすべてのバージョン、13.7 以降の 13 バージョン、13.4 バージョン

db.m6id — 第 3 世代インテル Xeon スケーラブルプロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.m6id.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョ

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
						ン、13.7 以降の 13 バージョン
db.m6id.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6id.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.m6idn — 第 3 世代インテル Xeon スケーラブルプロセッサ、SSD ストレージ、ネットワーク最適化を搭載した汎用インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m6idn.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.m6idn.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.m6idn.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

db.m6in — 第 3 世代インテル Xeon スケーラブルプロセッサとネットワーク最適化を搭載した汎用インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.m6in.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.m6in.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.m6in.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOのみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.m6in.metal	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ

db.m6in — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Oracle	PostgreSQL
db.m6i.32xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.m6i.12 xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.8x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.4x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.2x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.xl arge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.la rge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	Oracle Databa 19c	使用可能なすべてのバージョン
db.m6i.m tal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.m5d — インテル Xeon Platinum プロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m5d.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m5d.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.m5d.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

db.m5 — 2.5 GHz インテル Xeon Platinum プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db2	Maria	Microsoft SQL Server	MyS	Oracl	PostgreSQL
db.m5.24xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.16xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン

インスタンスクラス	Db2	Maria	Microsoft SQL Server	MyS	Oracl	PostgreSQL
db.m5.12x large	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.8xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.4xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.2xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.m5.large	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン

db.m4 — インテル Xeon プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	Db	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m4.16 large	いいえ	廃止	廃止	廃止	廃止	廃止

インスタンスクラス	Db	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m4.10large	いいえ	廃止	廃止	廃止	廃止	廃止
db.m4.4xarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.m4.2xarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.m4.xlarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.m4.large	いいえ	廃止	廃止	廃止	廃止	廃止

db.m3 – 汎用インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m3.2xlarge	いいえ	いいえ	廃止	はい	廃止	廃止
db.m3.xlarge	いいえ	いいえ	廃止	はい	廃止	廃止

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.m3.large	いいえ	いいえ	廃止	はい	廃止	廃止
db.m3.medium	いいえ	いいえ	廃止	はい	廃止	廃止

メモリ最適化インスタンスクラスでサポートされている DB エンジン

以下の表は、メモリ最適化インスタンスクラスでサポートされているデータベースとデータベースバージョンを示しています。

db.z1d – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.z1d.1xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.z1d.6large	いいえ	なし	はい	いいえ	はい	いいえ
db.z1d.3large	いいえ	なし	はい	いいえ	はい	いいえ
db.z1d.2large	いいえ	なし	はい	いいえ	はい	いいえ
db.z1d.xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.z1d.large	いいえ	なし	はい	いいえ	はい	いいえ

db.x2g – AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.x2g.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.x2g.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.x2g.8 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.x2g.4 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.x2g.2 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.x2g.x arge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.x2g.la rge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

db.x2idn — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Oracle	PostgreSQL
db.x2idn.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	Enterp e Editior のみ	PostgreSQL バージョン 15、14.6、および 13.9
db.x2idn.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	Enterp e Editior のみ	PostgreSQL バージョン 15、14.6、および 13.9
db.x2idn.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	Enterp e Editior のみ	PostgreSQL バージョン 15、14.6、および 13.9
db.x2idn.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.x2iedn - 第 3 世代インテル Xeon スケーラブルプロセッサを搭載し、ローカル NVMe ベースの SSD を備えたメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iedn.32xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition の	MySQL 8.4 および 8.0	Enterprise Edition のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョ

インスタンスクラス	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
			み、SQL Server 2014 12.00 以降			ン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition および Standard Edition 2 (SE2)	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition および Standard Edition 2 (SE2)	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

インスタンスクラス	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iedn.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	Enterprise および Standard Edition のみ、SQL Server 2014 12.00 以降	MySQL 8.4 および 8.0	Enterprise Edition および Standard Edition 2 (SE2)	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.x2iedn.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.x2m - 第 3 世代インテル Xeon スケーラブルプロセッサを搭載し、ローカル NVMe ベースの SSD を備えたメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2m.3xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.4xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.16xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ

インスタンスクラス	DI	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2m.8x large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.4x large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.2x large	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.x2m.metal	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ

db.x2iezn — 第 2 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iezn.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ
db.x2iezn.8xlarge	いいえ	いいえ	いいえ	いいえ	Enterprise Edition のみ	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x2iezn.6xlarge	いいえ	いいえ	いいえ	いいえ	Enterprise Edition のみ	いいえ
db.x2iezn.4xlarge	いいえ	いいえ	いいえ	いいえ	Enterprise Edition および Standard Edition 2 (SE2)	いいえ
db.x2iezn.2xlarge	いいえ	いいえ	いいえ	いいえ	Enterprise Edition および Standard Edition 2 (SE2)	いいえ

db.x1e – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x1e.32xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1e.16xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1e.8xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1e.4xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1e.2xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1e.xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ

db.x1 – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.x1.32xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ
db.x1.16xlarge	いいえ	いいえ	廃止	いいえ	廃止	いいえ

db.r8g – AWS Graviton4 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8g.48xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.24xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.16xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
						降、14.9 以降、および 13.11 以降
db.r8g.12xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.8xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.4xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.2xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8g.xlarge	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r8g.large	いいえ	MariaDB 12.3.2 以降、11.8.3 以降、11.4.3 以降、10.11.7 以降、10.6.13 以降、10.5.20 以降、および 10.4.29 以降	いいえ	MySQL 8.0.32 以降	いいえ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

db.r8i – インテル Xeon 6 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8i.96xlarge	いいえ	なし	はい	いいえ	BYO のみ、rise Editi のみ	いいえ
db.r8i.48xlarge	いいえ	なし	はい	いいえ	BYO のみ、rise Editi	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
					のみ	
db.r8i.32xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.r8i.24xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.r8i.16xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.r8i.12xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8i.8xlarge	いいえ	なし	はい	いいえ	BYOのみ、rise Editiのみ	いいえ
db.r8i.4xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.r8i.2xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.r8i.xlarge	いいえ	なし	はい	いいえ	はい	いいえ
db.r8i.large	いいえ	なし	はい	いいえ	はい	いいえ
db.r8i.metal-96xl	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ
db.r8i.metal-48xl	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ

db.r8a – 第 5 世代 AMD EPYC プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8a.48xlarge	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
db.r8a.24xlarge	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
db.r8a.16xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.r8a.12xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.r8a.8xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.r8a.4xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.r8a.2xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ
db.r8a.xlarge	いいえ	なし	はい	いいえ	いいえ	いいえ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r8a.large	いいえ	なし	はい	いいえ	いいえ	いいえ

db.r7i – 高メモリ、ストレージ、および I/O 用に事前設定されたメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.8xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.8xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.6xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
					Standard Edition 2	
db.r7i.6xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.4xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.4xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.4xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.3xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.2xlarge.tpc2.mem8x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.2xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ
db.r7i.xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	BYOL のみ、Enterprise Edition および Standard Edition 2	いいえ

db.r7i – 第 4 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.48xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
						降、14.9 以降、および 13.11 以降
db.r7i.12xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.0.32 以降	BYOL のみ、すべてのエディション	PostgreSQL バージョン 17.1 以降、16.1 以降、15.4 以降、14.9 以降、および 13.11 以降
db.r7i.metal-48xl	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Oracle	PostgreSQL
db.r7i.metal-24xl	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.r7g – AWS Graviton3 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r7g.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.8 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.4 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
						の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.2 large	い いえ	MariaDB 12.3、11.8 、11.4、10.11、10.6、1 0.5、および 10.4	いいえ	MySQL 8.4 および 8.0	い いえ	PostgreSQL 17、16、15 のすべての バージョン、14.5 以降 の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.xl arge	い いえ	MariaDB 12.3、11.8 、11.4、10.11、10.6、1 0.5、および 10.4	いいえ	MySQL 8.4 および 8.0	い いえ	PostgreSQL 17、16、15 のすべての バージョン、14.5 以降 の 14 バージョン、13.4 以降の 13 バージョン
db.r7g.la rge	い いえ	MariaDB 12.3、11.8 、11.4、10.11、10.6、1 0.5、および 10.4	いいえ	MySQL 8.4 および 8.0	い いえ	PostgreSQL 17、16、15 のすべての バージョン、14.5 以降 の 14 バージョン、13.4 以降の 13 バージョン

db.r6g – AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6g.16 xlarge	い いえ	MariaDB 12.3、11.8 、11.4、10. 11、10.6、10.5、およ び 10.4	いいえ	MySQL 8.4 および 8.0	い いえ	PostgreSQL 17、16、15、14、13 のすべてのバージョ ン、12.7 以降の 12 バー ジョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6g.12 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r6g.8x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r6g.4x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r6g.2x large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r6g.xl arge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6g.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

db.r6gd – AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6gd.6xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バー

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
						ジョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

db.r6id — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microso SQL Server	MySQL	Oracle	PostgreSQL
db.r6id.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
						ン、13.7 以降の 13 バージョン
db.r6id.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6id.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

db.r6idn — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6idn.32xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.12xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6idn.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6idn.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

db.r6in — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.32xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.24xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.16xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.12xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.8xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.4xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.2xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン
db.r6in.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYO のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.3 以降の 14 バージョン、13.7 以降の 13 バージョン、12.11 以降の 12 バージョン、11.16 以降の 11 バージョン

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6in.metal	いいえ	いいえ	いいえ	いいえ	BYO のみ	いいえ

db.r6i – 高メモリ、ストレージ、および I/O 用に事前設定されたメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6i.8xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	Enterprise Edition のみ	いいえ
db.r6i.8xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	いいえ	Enterprise Edition のみ	いいえ
db.r6i.6xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	Enterprise Edition のみ	いいえ
db.r6i.4xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.4xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.4xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6i.2xlarge.tpc2.mem8x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.2xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.2xlarge.tpc1.mem2x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ
db.r6i.large.tpc1.mem2x	いいえ	いいえ	いいえ	いいえ	EE と SE2 BYOL	いいえ

db.r6i – メモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6i.3 xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.2 xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.1 xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.1 xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6i.8x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.4x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.2x large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.xlarge	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6i.large	はい	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14 のすべてのバージョン、13.4 以降の 13 バージョン、12.8 以降の 12 バージョン、11.13 以降の 11 バージョン、10.21 以降の 10 バージョン
db.r6i.metal	いいえ	いいえ	いいえ	いいえ	BYOのみ	いいえ

db.r5d – メモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r5d.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r5d.1xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
						以降の 13 バージョン、および 13.4
db.r5d.1 xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r5d.8 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r5d.4 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r5d.2 large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r5d.x arge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

インスタンスクラス	Db	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r5d.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

db.r5b – 高メモリ、ストレージ、I/O 用に事前設定されたメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.8xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.6xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.4xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.4xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.4xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.2xlarge.tpc2.mem8x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.2xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.2xlarge.tpc1.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5b.large.tpc1.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ

db.r5b – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5b.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョ

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
						ン、12.7 以降の 12 バージョン
db.r5b.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r5b.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r5b.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	> はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r5b.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r5b.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

インスタンスクラス	Db2	MariaDB	Micros SQL Server	MySQL	Orac	PostgreSQL
db.r5b.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.r5b.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	はい	MySQL 8.4 および 8.0	はい	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

db.r5 – 高メモリ、ストレージ、I/O 用に事前設定されたメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5.12xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.8xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.6xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.4xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r5.4xlarge.tpc2.mem3x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.4xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.2xlarge.tpc2.mem8x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.2xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.2xlarge.tpc1.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.xlarge.tpc2.mem4x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.xlarge.tpc2.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ
db.r5.large.tpc1.mem2x	いいえ	いいえ	いいえ	なし	はい	いいえ

db.r5 – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaD	Microsoft SQL Server	MySC	Oracle	PostgreSQL
db.r5.24xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.16xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.12xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.8xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.4xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.2xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン
db.r5.xlarge	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン

インスタンスクラス	Db2	MariaD	Microsoft SQL Server	MySC	Oracle	PostgreSQL
db.r5.large	いいえ	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11 のすべてのバージョン、10.17 以降の 10 バージョン、9.6.22 以降の 9 バージョン

db.r4 – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r4.16xlarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.r4.8xlarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.r4.4xlarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.r4.2xlarge	いいえ	廃止	廃止	廃止	廃止	廃止
db.r4.xlarge	いいえ	廃止	廃止	廃止	廃止	廃止

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r4.large	いいえ	廃止	廃止	廃止	廃止	廃止

db.r3 – メモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQ	Oracle	PostgreSQL
db.r3.8xlarge**	いいえ	MariaDB 10.6、10.5、10.4、および 10.3 のすべてのバージョン	廃止	はい	廃止	廃止
db.r3.4xlarge	いいえ	MariaDB 10.6、10.5、10.4、および 10.3 のすべてのバージョン	廃止	はい	廃止	廃止
db.r3.2xlarge	いいえ	MariaDB 10.6、10.5、10.4、および 10.3 のすべてのバージョン	廃止	はい	廃止	廃止
db.r3.xlarge	いいえ	MariaDB 10.6、10.5、10.4、および 10.3 のすべてのバージョン	廃止	はい	廃止	廃止
db.r3.large	いいえ	MariaDB 10.6、10.5、10.4、および 10.3 のすべてのバージョン	廃止	はい	廃止	廃止

コンピューティング最適化インスタンスクラスでサポートされている DB エンジン

以下の表は、コンピューティング最適化インスタンスクラスでサポートされているデータベースとデータベースバージョンを示しています。

db.c6gd — コンピューティング最適化インスタンスクラス (マルチ AZ DB クラスター配置のみ)

インスタンスクラス	Db2	Maria	Microsoft SQL Server	MySQL	Orac	PostgreSQL
db.c6gd.16xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.12xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.8xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.4xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.2xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.xlarge	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14

インスタンスクラス	Db2	Maria	Microsoft SQL Server	MySQL	Orac	PostgreSQL
						バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.large	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン
db.c6gd.medium	いいえ	いいえ	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.4 および 13.7 以降の 13 バージョン

バーストパフォーマンスインスタンスクラスでサポートされている DB エンジン

以下の表は、バーストパフォーマンスインスタンスクラスでサポートされているデータベースとデータベースバージョンを示しています。

db.t4g – AWS Graviton2 プロセッサを搭載したバーストパフォーマンスインスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Orac	PostgreSQL
db.t4g.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.t4g.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョ

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Orac	PostgreSQL
						ン、12.7 以降の 12 バージョン
db.t4g.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.t4g.medium	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.t4g.small	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン
db.t4g.micro	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15、14、13 のすべてのバージョン、12.7 以降の 12 バージョン

db.t3 – バーストパフォーマンスインスタンスクラス

インスタンスクラス	Db2	Maria	Microsoft SQL Server	MySQ	Oracle	PostgreSQL
db.t3.2xlarge	はい	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン
db.t3.xlarge	はい	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン
db.t3.large	はい	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン
db.t3.medium	はい	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン
db.t3.small	はい	はい	はい	はい	はい	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン
db.t3.micro	いいえ	はい	はい	はい	Oracle Database 12c Release 1 (12.1.0.2) のみ (廃止)	PostgreSQL 17、16、15、14、13、12、11、10 のすべてのバージョン、9.6.22 以降の 9 バージョン

db.t2 – バーストパフォーマンスインスタンスクラス

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.t2.2xlarge	いいえ	廃止	いいえ	廃止	廃止	廃止
db.t2.xlarge	いいえ	廃止	いいえ	廃止	廃止	廃止
db.t2.large	いいえ	廃止	廃止	廃止	廃止	廃止
db.t2.medium	いいえ	廃止	廃止	廃止	廃止	廃止
db.t2.small	いいえ	廃止	廃止	廃止	廃止	廃止
db.t2.micro	いいえ	廃止	廃止	廃止	廃止	廃止

Optimized Reads インスタンスクラスでサポートされている DB エンジン

以下の表は、Optimized Reads インスタンスクラスでサポートされているデータベースとデータベースバージョンを示しています。

db.m8gd — Optimized Reads をサポートし、AWS Graviton4 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.m8gd 8xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd 4xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd 6xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd 2xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
				降、8.4.3 以降		
db.m8gd xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.m8gd arge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

db.r8gd — Optimized Reads をサポートし、AWS Graviton4 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r8gd.4 8xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r8gd.2 4xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.1 6xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.1 2xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.8 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.4 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r8gd.2 xlarge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.x large	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降
db.r8gd.l arge	いいえ	MariaDB 10.5.27 以降、10.6.19 以降、10.11.9 以降、11.4.3 以降、11.8.3 以降、および 12.3.2 以降	いいえ	MySQL 8.0.40 以降、8.4.3 以降	いいえ	PostgreSQL 13.18 以降、14.17 以降、15.10 以降、16.6 以降、17.2 および 18.1 以降

db.r6gd — Optimized Reads をサポートし、AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	DI	MariaDB	Micros SQL Serve	MySQL	Ora	PostgreSQL
db.r6gd.1 6xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

インスタンスクラス	DI	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6gd.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4
db.r6gd.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

インスタンスクラス	Db	MariaDB	Micros SQL Server	MySQL	Ora	PostgreSQL
db.r6gd.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	いいえ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン、および 13.4

db.r6id — Optimized Reads をサポートし、第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	Db2	MariaDB	Microso SQL Server	MySQL	Oracle	PostgreSQL
db.r6id.32xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.24xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.16xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョ

インスタンスクラス	Db2	MariaDB	Microso SQL Server	MySQL	Oracle	PostgreSQL
						ン、13.7 以降の 13 バージョン
db.r6id.12xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.8xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	EE および BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.4xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.2xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン

インスタンスクラス	Db2	MariaDB	Microsoft SQL Server	MySQL	Oracle	PostgreSQL
db.r6id.xlarge	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.large	いいえ	MariaDB 12.3、11.8、11.4、10.11、10.6、10.5、および 10.4	いいえ	MySQL 8.4 および 8.0	BYOL のみ	PostgreSQL 17、16、15 のすべてのバージョン、14.5 以降の 14 バージョン、13.7 以降の 13 バージョン
db.r6id.metal	いいえ	いいえ	いいえ	いいえ	BYOL のみ	いいえ

AWS リージョン での DB インスタンスクラスのサポートを決定する

特定の AWS リージョン で各 DB エンジンがサポートしている DB インスタンスクラスを決定するため、複数のアプローチの中から 1 つを選択できます。AWS マネジメントコンソール で、「Amazon RDS 料金表」ページ、または AWS Command Line Interface (AWS CLI) の describe-orderable-db-instance-options コマンドを使用できます。

> **Note**
>
> AWS マネジメントコンソール でオペレーションを実行すると、特定の DB エンジン、DB エンジンバージョン、および AWS リージョン でサポートされる DB インスタンスクラスが自動的に表示されます。実行できるオペレーションの例には、DB クラスターの作成や変更などが含まれます。

目次

- Amazon RDS 料金ページを使用して、AWS リージョン での DB インスタンスクラスのサポートを決定する
- AWS CLI を使用して、AWS リージョン 内での DB インスタンスクラスのサポートを決定する
  - AWS リージョン 内の特定の DB エンジンバージョンでサポートされている DB インスタンスクラスの一覧表示
  - AWS リージョン 内で特定の DB インスタンスクラスをサポートする DB エンジンのバージョンの一覧表示
  - 特定の DB エンジンおよびインスタンスクラスをサポートする AWS リージョンの一覧表示

Amazon RDS 料金ページを使用して、AWS リージョン での DB インスタンスクラスのサポートを決定する

Amazon RDS 料金 ページを使用して、特定の AWS リージョン 内で各 DB エンジンがサポートしている、DB インスタンスクラスを決定できます。

料金ページを使用して、リージョンの各エンジンでサポートされる DB インスタンスクラスを決定するには

1. Amazon RDS 料金に移動します。
2. [Amazon RDS 用の AWS 料金見積りツール] セクションで、[カスタム見積りを今すぐ作成] を選択します。
3. [リージョンを選択] で、[AWS リージョン] を選択します。
4. [サービスを検索] に、「**Amazon RDS**」と入力します。
5. 設定オプションと DB エンジンの [設定] を選択します。
6. 互換性のあるインスタンスのセクションを使用して、サポートされている DB インスタンスクラスを確認します。
7. (オプション) 見積りツールで他のオプションを選択し、[概要を保存して表示] または [サービスを保存して追加] を選択します。

AWS CLI を使用して、AWS リージョン 内での DB インスタンスクラスのサポートを決定する

AWS CLI を使用して、AWS リージョン 内の特定の DB エンジンおよび DB エンジンバージョンでサポートされる DB インスタンスクラスを決定できます。次の表は、有効な DB エンジンの値を示しています。

エンジン名	CLI コマンドのエンジン値	バージョンの詳細

Db2	`db2-ae`
	`db2-ce`
	`db2-se`
		Amazon RDS での Db2 のバージョン

MariaDB	`mariadb`	Amazon RDS の MariaDB のバージョン

Microsoft SQL Server	`sqlserver-ee`
	`sqlserver-se`
	`sqlserver-ex`
	`sqlserver-web`
		Amazon RDS での Microsoft SQL Server バージョン

MySQL	`mysql`	Amazon RDS での MySQL のバージョン
Oracle	`oracle-ee` `oracle-se2`	Amazon RDS for Oracle リリースノート
PostgreSQL	`postgres`	利用可能な PostgreSQL データベースのバージョン

AWS リージョン 名については、「AWS リージョン」を参照してください。

次の例は、describe-orderable-db-instance-options AWS CLI コマンドを使用して、AWS リージョンでの DB インスタンスクラスのサポートを決定する方法を示しています。

> **Note**
>
> 出力を制限するために、以下の例は、汎用 SSD (gp2) ストレージタイプのみの結果を示しています。必要に応じて、コマンドでストレージタイプを汎用 SSD (gp3)、プロビジョンド IOPS (io1)、プロビジョンド IOPS Block Express (io2) またはマグネティック (standard) に変更できます。

トピック

- AWS リージョン 内の特定の DB エンジンバージョンでサポートされている DB インスタンスクラスの一覧表示
- AWS リージョン 内で特定の DB インスタンスクラスをサポートする DB エンジンのバージョンの一覧表示
- 特定の DB エンジンおよびインスタンスクラスをサポートする AWS リージョンの一覧表示

AWS リージョン 内の特定の DB エンジンバージョンでサポートされている DB インスタンスクラスの一覧表示

AWS リージョン 内の特定の DB エンジンバージョンでサポートされている DB インスタンスクラスを一覧表示するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 \
    --query "*[].{DBInstanceClass:DBInstanceClass,StorageType:StorageType}|[?
StorageType=='gp2']|[].{DBInstanceClass:DBInstanceClass}" \
    --output text \
    --region region
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 ^
    --query "*[].{DBInstanceClass:DBInstanceClass,StorageType:StorageType}|[?
StorageType=='gp2']|[].{DBInstanceClass:DBInstanceClass}" ^
    --output text ^
    --region region
```

例えば、次のコマンドは、米国東部 (バージニア北部) の RDS for PostgreSQL DB エンジンのバージョン 13.6 でサポートされている DB インスタンスクラスを一覧表示します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options --engine postgres --engine-version 15.4
 \
    --query "*[].{DBInstanceClass:DBInstanceClass,StorageType:StorageType}|[?
StorageType=='gp2']|[].{DBInstanceClass:DBInstanceClass}" \
    --output text \
    --region us-east-1
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options --engine postgres --engine-version 15.4
 ^
    --query "*[].{DBInstanceClass:DBInstanceClass,StorageType:StorageType}|[?
StorageType=='gp2']|[].{DBInstanceClass:DBInstanceClass}" ^
    --output text ^
    --region us-east-1
```

AWS リージョン 内で特定の DB インスタンスクラスをサポートする DB エンジンのバージョンの一覧表示

AWS リージョン 内で特定の DB インスタンスクラスをサポートしている DB エンジンのバージョンを一覧表示するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class \
    --query "*[].{EngineVersion:EngineVersion,StorageType:StorageType}|[?
StorageType=='gp2']|[].{EngineVersion:EngineVersion}" \
    --output text \
    --region region
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class ^
```

```
    --query "*[].{EngineVersion:EngineVersion,StorageType:StorageType}|[?
StorageType=='gp2']|[].{EngineVersion:EngineVersion}" ^
    --output text ^
    --region region
```

例えば、次のコマンドは、米国東部 (バージニア北部) の db.r5.large DB インスタンスクラスをサポートする RDS for PostgreSQL DB エンジンの DB エンジンのバージョンを一覧表示します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options --engine postgres --db-instance-class
 db.m7g.large \
    --query "*[].{EngineVersion:EngineVersion,StorageType:StorageType}|[?
StorageType=='gp2']|[].{EngineVersion:EngineVersion}" \
    --output text \
    --region us-east-1
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options --engine postgres --db-instance-class
 db.m7g.large ^
    --query "*[].{EngineVersion:EngineVersion,StorageType:StorageType}|[?
StorageType=='gp2']|[].{EngineVersion:EngineVersion}" ^
    --output text ^
    --region us-east-1
```

特定の DB エンジンおよびインスタンスクラスをサポートする AWS リージョンの一覧表示

次の bash スクリプトは、DB エンジンとインスタンスクラスの指定された組み合わせをサポートするすべての AWS リージョンを一覧表示します。

```
#!/usr/bin/env bash
# Usage: check_region_support.sh <db-engine> <db-instance-class>

if [ $# -ne 2 ]; then
  echo "Usage: $0 <db-engine> <db-instance-class>"
  exit 1
fi
ENGINE="$1"
INSTANCE_CLASS="$2"
REGIONS=$(aws ec2 describe-regions --query "Regions[].RegionName" --output text)
```

```
for region in $REGIONS; do
  supported_count=$(aws rds describe-orderable-db-instance-options \
    --region "$region" \
    --engine "$ENGINE" \
    --db-instance-class "$INSTANCE_CLASS" \
    --query 'length(OrderableDBInstanceOptions)' \
    --output text 2>/dev/null || echo "0")
  if [ "$supported_count" -gt 0 ]; then
    echo "$region supports $INSTANCE_CLASS for $ENGINE."
  else
    echo "$region doesn't support $INSTANCE_CLASS for $ENGINE."
  fi
done
```

次の出力例では、db.r8g.large インスタンスクラスを使用して、RDS for MySQL のリージョンサポートをチェックします。

```
./check_region_support.sh mysql db.r8g.large
ap-south-1 doesn't support db.r8g.large for mysql.
eu-north-1 doesn't support db.r8g.large for mysql.
eu-west-3 doesn't support db.r8g.large for mysql.
eu-west-2 doesn't support db.r8g.large for mysql.
eu-west-1 doesn't support db.r8g.large for mysql.
ap-northeast-3 doesn't support db.r8g.large for mysql.
ap-northeast-2 doesn't support db.r8g.large for mysql.
ap-northeast-1 doesn't support db.r8g.large for mysql.
ca-central-1 doesn't support db.r8g.large for mysql.
sa-east-1 doesn't support db.r8g.large for mysql.
ap-southeast-1 doesn't support db.r8g.large for mysql.
ap-southeast-2 doesn't support db.r8g.large for mysql.
eu-central-1 supports db.r8g.large for mysql.
us-east-1 supports db.r8g.large for mysql.
us-east-2 supports db.r8g.large for mysql.
us-west-1 doesn't support db.r8g.large for mysql.
us-west-2 supports db.r8g.large for mysql.
```

DB インスタンスクラスの変更

DB インスタンスで利用可能な CPU やメモリを変更するには、その DB インスタンスクラスを変更します。DB インスタンスクラスを変更するには、「Amazon RDS DB インスタンスを変更する」の手順に従って DB インスタンスを変更します。

マルチ AZ 配置のクロスアベイラビリティーゾーン ENA Express のサポート

Amazon RDS は、AWS の Scalable Reliable Datagram (SRD) プロトコルを使用するマルチ AZ 配置において、対象となる DB インスタンスクラスのクロスアベイラビリティーゾーン ENA Express をサポートしています。クロスアベイラビリティーゾーン ENA Express は、シングルフローのネットワーク帯域幅を増やし、プライマリ DB インスタンスとスタンバイ DB インスタンス間のブロックレベルのデータレプリケーションを高速化します。

SRD の詳細と、クロスアベイラビリティーゾーン ENA Express をサポートするインスタンスクラスのリストについては、Amazon EC2 ドキュメントの「ENA Express」を参照してください。

> **Note**
>
> Amazon RDS for SQL Server では、クロスアベイラビリティーゾーン ENA Express はサポートされていません。

利点

クロスアベイラビリティーゾーン ENA Express を有効にすると、マルチ AZ 配置では、アベイラビリティーゾーン間のデータレプリケーションのシングルフロー帯域幅が増加します。高度な輻輳制御とマルチパス機能により、クロスアベイラビリティーゾーン ENA Express はレプリケーションレイテンシーとレプリケーションレイテンシーの変動性を低減し、書き込み集約型ワークロードのフェイルオーバーの信頼性を向上させます。

実際のスループットは、DB インスタンスクラスの合計帯域幅制限、データベースエンジン、データベース設定によって異なります。

クロスアベイラビリティーゾーン ENA Express の有効化

クロスアベイラビリティーゾーン ENA Express は、対象となるインスタンスクラスを使用する新しく作成されたすべてのマルチ AZ DB インスタンスでデフォルトで有効になっています。

既存のマルチ AZ DB インスタンスでクロスアベイラビリティーゾーン ENA Express を有効にするには、次のいずれかを実行します。

- DB インスタンスの停止と起動。詳細については、「一時的に Amazon RDS DB インスタンスを停止する」を参照してください。

- 対象となるインスタンスクラスを使用するように DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- 対象となるインスタンスクラスでスケールコンピューティングを使用します。詳細については、「スケールコンピューティング」を参照してください。

RDS for Oracle で DB インスタンスクラスのプロセッサを設定する

Amazon RDS DB インスタンスクラスは、単一のインテル Xeon CPU コアで同時に複数のスレッドを実行できるインテルハイパースレッディングテクノロジーをサポートしています。各スレッドは、DB インスタンスの仮想 CPU (vCPU) として表されます。DB インスタンスには、デフォルトの CPU コア数があります。これは、DB インスタンスタイプによって異なります。例えば、db.m4.xlarge DB インスタンスタイプには 2 つの CPU コアがあり、デフォルトではコアごとに 2 つのスレッドの合計で 4 つの vCPU があります。

> **Note**
>
> 各 vCPU は、インテルXeon CPU コアのハイパースレッドです。

トピック

- RDS for Oracle のプロセッサ設定の概要
- プロセッサ設定をサポートする DB インスタンスクラス
- DB インスタンスクラスの CPU コア数と CPU コアあたりのスレッド数の設定

RDS for Oracle のプロセッサ設定の概要

RDS for Oracle を使用すると、通常、ワークロードに適したメモリと vCPU 数を組み合わせた DB インスタンスクラスを見つけることができます。ただし、特定のワークロードまたはビジネスのニーズに合わせて、RDS for Oracle DB インスタンスを最適化するために、以下のプロセッサ機能を指定することもできます。

- CPU コア数 - DB インスタンスの CPU コア数をカスタマイズできます。これによって、大量のメモリを使用するワークロード用に十分な RAM 量がありながら、少ない CPU コアの DB インスタンスのソフトウェアのライセンスコストを最適化することにつながります。

- コア別のスレッド - Intel ハイパースレッディングテクノロジーを無効化するには、CPU コアごとに 1 つのスレッドを指定できます。高性能コンピューティング (HPC) のワークロードのような特定のワークロードでこれを使用できます。

各コアで、CPU コア数とスレッド数を個別に制御できます。1 つのリクエストでどちらか片方または両方を設定できます。設定は DB インスタンスに関連付けられると、変更するまで維持されます。

DB インスタンスのプロセッサ設定は、DB インスタンスのスナップショットに関連付けられます。スナップショットを復元されると、復元された DB インスタンスは、スナップショットが作成されたときに使用されたプロセッサ機能設定を使用します。

デフォルト以外のプロセッサ設定を持つ DB インスタンスの DB インスタンスクラスを変更する場合は、デフォルトのプロセッサ設定を指定するか、変更時にプロセッサ設定を明示的に指定します。この要件により、DB インスタンスを変更する場合に発生する可能性があるサードパーティーのライセンスコストを確認できます。

RDS for Oracle DB インスタンスでプロセッサ機能を指定しても、課金の追加や割引はありません。デフォルトの CPU 設定で起動した DB インスタンスと同じように課金されます。

プロセッサ設定をサポートする DB インスタンスクラス

次の条件が満たされている場合にのみ、CPU コア数およびコアあたりのスレッド数を設定できます。

- RDS for Oracle DB インスタンスを設定しています。さまざまな Oracle データベースエディションによってサポートされる DB インスタンスクラスについては、「RDS for Oracle DB インスタンスクラス」を参照してください。
- DB インスタンスで RDS for Oracle の Bring-Your-Own-License (BYOL) ライセンスオプションを使用しています。Oracle ライセンスのオプションの詳細については、「RDS for Oracle のライセンスオプション」を参照してください。
- DB インスタンスは、事前定義されたプロセッサ設定を持つ db.r5 または db.r5b インスタンスクラスには属していません。これらのインスタンスクラスの名前は、db.r5.*instance_size*.tpc*threads_per_core*.mem*ratio* または db.r5b.*instance_size*.tpc*threads_per_core*.mem*ratio* という形式です。例えば、db.r5b.xlarge.tpc2.mem4x は、コアあたり 2 スレッド (tpc2)、スタンダードの db.r5b.xlarge インスタンスクラスの 4 倍のメモリで事前設定されています。これらの最適化されたインスタンスクラスのプロセッサ機能を設定することはできません。詳細については、「サポートされている RDS for Oracle DB インスタンスクラス」を参照してください。

次の AWS CLI コマンドを使用して、インスタンスクラスのデフォルトの vCPU、コア、コアあたりのスレッド、およびコアの有効数を表示できます。サンプルコマンドの *r7i.48xlarge* をインスタンスクラスの名前に置き換えます。

```
aws ec2 describe-instance-types \
    --instance-types r7i.48xlarge \
    --query '{
        DefaultVCPUs: InstanceTypes[0].VCpuInfo.DefaultVCpus,
        DefaultCores: InstanceTypes[0].VCpuInfo.DefaultCores,
        DefaultThreadsPerCore: InstanceTypes[0].VCpuInfo.DefaultThreadsPerCore,
        ValidCores: InstanceTypes[0].VCpuInfo.ValidCores
    }' \
    --output json
```

次の表では、CPU コア数とコアあたりの CPU スレッド数の設定をサポートする DB インスタンスクラスを確認できます。また、DB インスタンスクラスごとに CPU コア数とコアあたりの CPU スレッド数のデフォルト値と有効値も確認できます。

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.m6i - メモリ最適化インスタンスクラス					
db.m6i.large	2	1	2	1	1、2
db.m6i.xlarge	4	2	2	2	1、2
db.m6i.2xlarge	8	4	2	2、4	1、2
db.m6i.4xlarge	16	8	2	2、4、6、8	1、2
db.m6i.4xlarge	16	8	2	2、4、6、8	1、2
db.m6i.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.m6i.12 xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.m6i.16 xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.m6i.24 xlarge	96	48	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.m6i.32 xlarge	128	64	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1、2
db.m5 – 汎用インスタンスクラス					
db.m5.large	2	1	2	1	1、2
db.m5.xlarge	4	2	2	2	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.m5.2xlarge	8	4	2	2、4	1、2
db.m5.4xlarge	16	8	2	2、4、6、8	1、2
db.m5.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.m5.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.m5.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.m5.24xlarge	96	48	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.m5d – 汎用インスタンスクラス					
db.m5d.large	2	1	2	1	1、2
db.m5d.xlarge	4	2	2	2	1、2
db.m5d.2xlarge	8	4	2	2、4	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.m5d.4xlarge	16	8	2	2、4、6、8	1、2
db.m5d.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.m5d.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.m5d.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.m5d.24xlarge	96	48	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.m4 – 汎用インスタンスクラス					
db.m4.10xlarge	40	20	2	2、4、6、8、10、12、14、16、18、20	1、2
db.m4.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.r7i – メモリ最適化インスタンスクラス					

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r7i.large	2	1	2	1	1、2
db.r7i.xlarge	4	2	2	1、2	1、2
db.r7i.2xlarge	8	4	2	1、2、3、4	1、2
db.r7i.4xlarge	16	8	2	1、2、3、4、5、6、7、8	1、2
db.r7i.8xlarge	32	16	2	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16	1、2
db.r7i.12xlarge	48	24	2	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24	1、2
db.r7i.16xlarge	64	32	2	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r7i.24xlarge	96	48	2	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32、33、34、35、36、37、38、39、40、41、42、43、44、45、46、47、48	1、2
db.r7i.48xlarge	192	96	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64、66、68、70、72、74、76、78、80、82、84、86、88、90、92、94、96	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r6i – メモリ最適化インスタンスクラス					
db.r6i.large	2	1	2	1	1、2
db.r6i.xlarge	4	2	2	1、2	1、2
db.r6i.2xlarge	8	4	2	2、4	1、2
db.r6i.4xlarge	16	8	2	2、4、6、8	1、2
db.r6i.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.r6i.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.r6i.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.r6i.24xlarge	96	48	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r6i.32xlarge	128	64	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1、2
db.r5 – メモリ最適化インスタンスクラス					
db.r5.large	2	1	2	1	1、2
db.r5.xlarge	4	2	2	2	1、2
db.r5.2xlarge	8	4	2	2、4	1、2
db.r5.4xlarge	16	8	2	2、4、6、8	1、2
db.r5.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.r5.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.r5.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r5.24xlarge	96	48	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.r5 – メモリ最適化インスタンスクラス					
db.r5b.large	2	1	2	1	1、2
db.r5b.xlarge	4	2	2	2	1、2
db.r5b.2xlarge	8	4	2	2、4	1、2
db.r5b.4xlarge	16	8	2	2、4、6、8	1、2
db.r5b.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.r5b.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.r5b.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r5b.24xlarge	96	48	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.r5d – メモリ最適化インスタンスクラス					
db.r5d.large	2	1	2	1	1、2
db.r5d.xlarge	4	2	2	2	1、2
db.r5d.2xlarge	8	4	2	2、4	1、2
db.r5d.4xlarge	16	8	2	2、4、6、8	1、2
db.r5d.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.r5d.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.r5d.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r5d.24xlarge	96	48	2	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.r4 – メモリ最適化インスタンスクラス					
db.r4.large	2	1	2	1	1、2
db.r4.xlarge	4	2	2	1、2	1、2
db.r4.2xlarge	8	4	2	1、2、3、4	1、2
db.r4.4xlarge	16	8	2	1、2、3、4、5、6、7、8	1、2
db.r4.8xlarge	32	16	2	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16	1、2
db.r4.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.r3 – メモリ最適化インスタンスクラス					
db.r3.large	2	1	2	1	1、2
db.r3.xlarge	4	2	2	1、2	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.r3.2xlarge	8	4	2	1、2、3、4	1、2
db.r3.4xlarge	16	8	2	1、2、3、4、5、6、7、8	1、2
db.r3.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.x2idn – メモリ最適化インスタンスクラス					
db.x2idn.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.x2idn.24xlarge	96	48	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2
db.x2idn.32xlarge	128	64	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.x2iedn – メモリ最適化インスタンスクラス					
db.x2iedn.xlarge	4	2	2	1、2	1、2
db.x2iedn.2xlarge	8	4	2	2、4	1、2
db.x2iedn.4xlarge	16	8	2	2、4、6、8	1、2
db.x2iedn.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.x2iedn.16xlarge	64	32	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32	1、2
db.x2iedn.24xlarge	96	48	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.x2iedn.32xlarge	128	64	2	2、4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1、2
db.x2iezn – メモリ最適化インスタンスクラス					
db.x2iezn.2xlarge	8	4	2	2、4	1、2
db.x2iezn.4xlarge	16	8	2	2、4、6、8	1、2
db.x2iezn.6xlarge	24	12	2	2、4、6、8、10、12	1、2
db.x2iezn.8xlarge	32	16	2	2、4、6、8、10、12、14、16	1、2
db.x2iezn.12xlarge	48	24	2	2、4、6、8、10、12、14、16、18、20、22、24	1、2
db.z1d – メモリ最適化インスタンスクラス					
db.z1d.large	2	1	2	1	1、2
db.z1d.xlarge	4	2	2	2	1、2

DB インスタンスクラス	デフォルト vCPU	デフォルトの CPU コア	コアごとのデフォルトのスレッド	CPU コアの有効数	コアごとのスレッドの有効数
db.z1d.2xlarge	8	4	2	2、4	1、2
db.z1d.3xlarge	12	6	2	2、4、6	1、2
db.z1d.6xlarge	24	12	2	2、4、6、8、10、12	1、2
db.z1d.12 xlarge	48	24	2	4、6、8、10、12、14、16、18、20、22、24	1、2

> **Note**
>
> Amazon RDS for Oracle DB インスタンスの設定を処理するには、AWS CloudTrail を使用して、変更のモニタリングと監査を行います。CloudTrail の使用の詳細については、「AWS CloudTrail での Amazon RDS API コールのモニタリング」を参照してください。

DB インスタンスクラスの CPU コア数と CPU コアあたりのスレッド数の設定

次のオペレーションを実行するときに、DB インスタンスクラスの CPU コア数とコアあたりのスレッド数を設定できます。

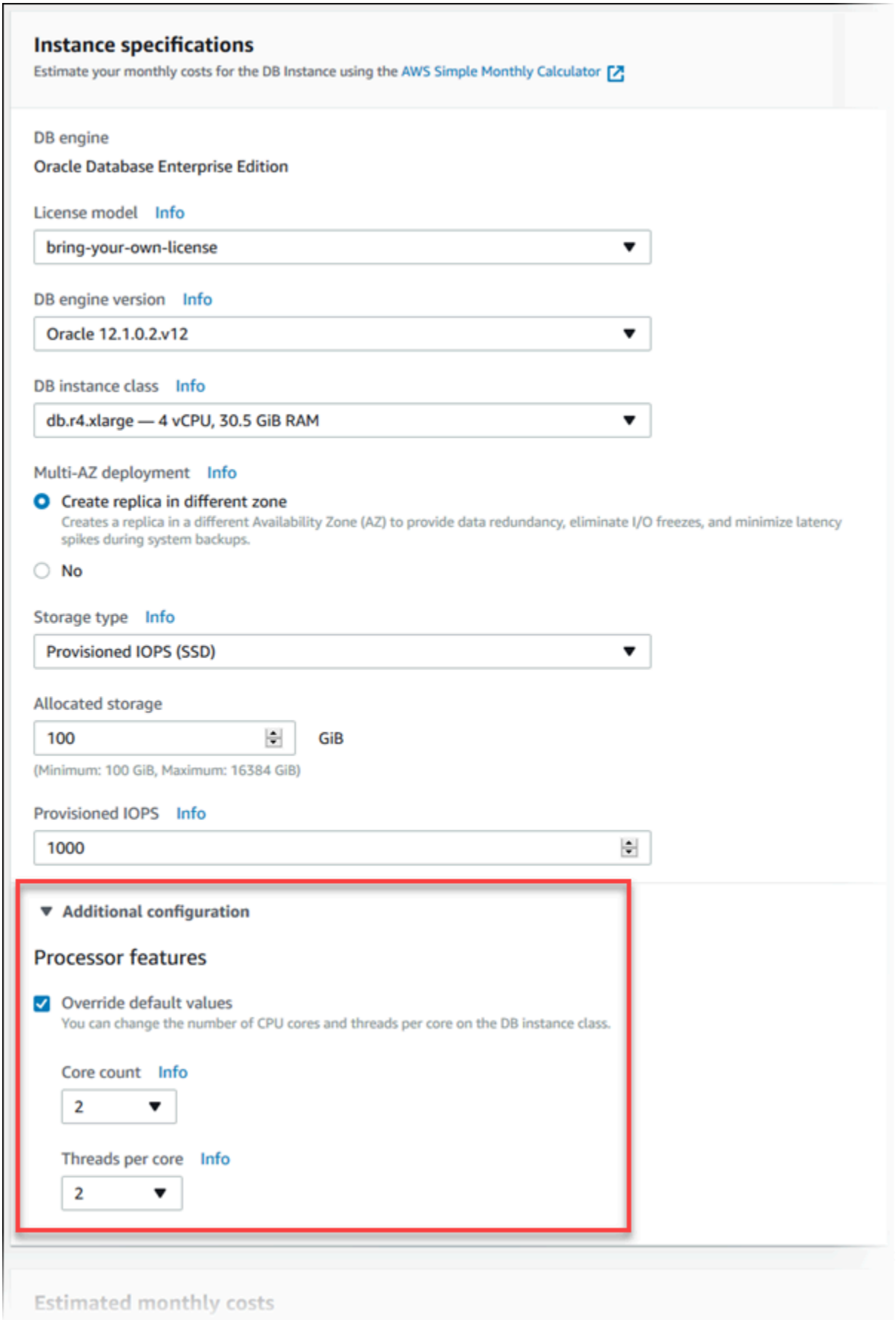
- Amazon RDS DB インスタンスの作成
- Amazon RDS DB インスタンスを変更する
- DB インスタンスへの復元
- Amazon RDS の DB インスタンスを特定の時点に復元する

> **Note**
>
> DB インスタンスを変更してコアごとの CPU 数とスレッド数を指定する際に、DB インスタンスが短時間停止します。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB インスタンスクラスの CPU コアごとに CPU コアとスレッドを設定できます。

コンソール

DB インスタンスを作成、変更、または復元するときに、AWS マネジメントコンソール で DB インスタンスクラスを設定します。[Instance specifications (インスタンスの仕様)] セクションに、プロセッサのオプションが表示されます。次のイメージにプロセッサ機能オプションを示します。

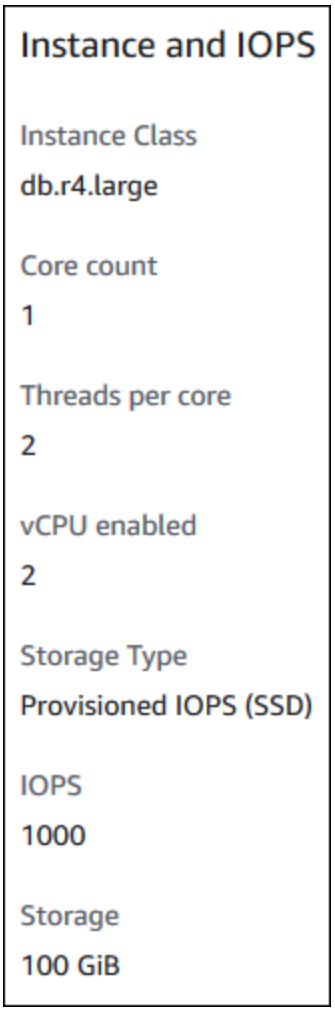

Instance specifications
Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator
DB engine
Oracle Database Enterprise Edition
License model Info
bring-your-own-license
DB engine version Info
Oracle 12.1.0.2.v12
DB instance class Info
db.r4.xlarge — 4 vCPU, 30.5 GiB RAM
Multi-AZ deployment Info
Create replica in different zone
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.
No
Storage type Info
Provisioned IOPS (SSD)
Allocated storage
100
GiB
(Minimum: 100 GiB, Maximum: 16384 GiB)
Provisioned IOPS Info
1000
Additional configuration
Processor features
Override default values
You can change the number of CPU cores and threads per core on the DB instance class.
Core count Info
2
Threads per core Info
2
Estimated monthly costs

[Processor features (プロセッサ機能)] の下で、次のオプションを、使用している DB インスタンスクラスに適切な値に設定します。

- Core count (コア数)- CPU コアの数を設定するにはこのオプションを使用します。値は、その DB インスタンスクラスの CPU コアの最大数以下である必要があります。
- Threads per core (コアあたりのスレッド) - コアごとに複数のスレッドを有効にするには 2 を指定します。コアごとの複数スレッドを無効にするには 1 を指定します。

DB インスタンスを変更、または復元する際に、CPU コアごとの CPU コアとスレッドも、インスタンスクラスのデフォルト値に設定できます。

コンソールで DB インスタンスの詳細を表示すると、DB インスタンスクラスのプロセッサ情報を [設定] タブで確認できます。次のイメージは 1 つの CPU コアと複数のスレッドが有効になっている DB インスタンスクラスを示します。



Oracle DB インスタンスの場合、Bring-Your-Own-License (BYOL) DB インスタンスの場合のみプロセッサ情報が表示されます。

AWS CLI

次の AWS CLI コマンドのいずれかを実行する際に DB インスタンスのプロセッサ機能を設定できます。

- create-db-instance
- modify-db-instance
- DBスナップショットからDBインスタンスを復元する
- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

AWS CLI を使用して DB インスタンスで DB インスタンスクラスのプロセッサを設定するには、コマンドに `--processor-features` オプションを含めます。`coreCount` 機能名で CPU コアの数を指定し、`threadsPerCore` 機能名で複数のスレッドを有効にするかどうかを指定します。

オプションの構文は次のとおりです。

```
--processor-features "Name=coreCount,Value=<value>" "Name=threadsPerCore,Value=<value>"
```

プロセッサの設定の例を以下に示します。

例

- DB インスタンスの CPU コア数の設定
- DB インスタンスの CPU コアの数を設定し、複数のスレッドを無効にする
- DB インスタンスクラスの有効なプロセッサ値を確認する
- DB インスタンスのデフォルトのプロセッサ設定に戻す
- DB インスタンスのデフォルトの CPU コア数に戻す
- DB インスタンスのデフォルトのコアあたりのスレッド数に戻す

DB インスタンスの CPU コア数の設定

Example

次の例では、CPU コアの数を 4 に設定して mydbinstance を変更します。--apply-immediately を使用すると変更はすぐに適用されます。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately オプションを省略します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --processor-features "Name=coreCount,Value=4" \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --processor-features "Name=coreCount,Value=4" ^
    --apply-immediately
```

DB インスタンスの CPU コアの数を設定し、複数のスレッドを無効にする

Example

次の例では、mydbinstance を変更して CPU コアの数を 4 に設定し、コアごとの複数のスレッドを無効にします。--apply-immediately を使用すると変更はすぐに適用されます。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately オプションを省略します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --processor-features "Name=coreCount,Value=4" "Name=threadsPerCore,Value=1" \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --processor-features "Name=coreCount,Value=4" "Name=threadsPerCore,Value=1" ^
    --apply-immediately
```

DB インスタンスクラスの有効なプロセッサ値を確認する

Example

特定の DB インスタンスクラスの有効なプロセッサ値を確認するには、describe-orderable-db-instance-options コマンドを実行し、--db-instance-class オプションでインスタンスクラスを指定します。例えば、次のコマンドの出力は db.r3.large インスタンスクラスのプロセッサオプションを示します。

```
aws rds describe-orderable-db-instance-options --engine oracle-ee --db-instance-class
 db.r3.large
```

以下はコマンドの JSON 形式の出力サンプルです。

```
    {
                "SupportsIops": true,
                "MaxIopsPerGib": 50.0,
                "LicenseModel": "bring-your-own-license",
                "DBInstanceClass": "db.r3.large",
                "SupportsIAMDatabaseAuthentication": false,
                "MinStorageSize": 100,
                "AvailabilityZones": [
                    {
                        "Name": "us-west-2a"
                    },
                    {
                        "Name": "us-west-2b"
                    },
                    {
                        "Name": "us-west-2c"
                    }
                ],
                "EngineVersion": "12.1.0.2.v2",
                "MaxStorageSize": 32768,
                "MinIopsPerGib": 1.0,
                "MaxIopsPerDbInstance": 40000,
                "ReadReplicaCapable": false,
```

```
            "AvailableProcessorFeatures": [
                {
                    "Name": "coreCount",
                    "DefaultValue": "1",
                    "AllowedValues": "1"
                },
                {
                    "Name": "threadsPerCore",
                    "DefaultValue": "2",
                    "AllowedValues": "1,2"
                }
            ],
            "SupportsEnhancedMonitoring": true,
            "SupportsPerformanceInsights": false,
            "MinIopsPerDbInstance": 1000,
            "StorageType": "io1",
            "Vpc": false,
            "SupportsStorageEncryption": true,
            "Engine": "oracle-ee",
            "MultiAZCapable": true
    }
```

さらに、次のコマンドを実行して DB インスタンスのクラスのプロセッサ情報を取得できます。

- describe-db-instances - 指定された DB インスタンスのプロセッサ情報を示します。
- describe-db-snapshots - 指定された DB スナップショットのプロセッサ情報を示します。
- describe-valid-db-instance-modifications - 指定された DB インスタンスのプロセッサに対する有効な変更を示します。

上記のコマンドの出力では、次の条件が満たされている場合にのみ、プロセッサ機能の値は null ではありません。

- RDS for Oracle DB インスタンスを使用しています。
- RDS for Oracle DB インスタンスは、プロセッサ値の変更をサポートしています。
- 現在の CPU コアとスレッドの設定は、デフォルト以外の値に設定されます。

上記の条件が満たされない場合、describe-db-instances を使用してインスタンスタイプを取得できます。EC2 オペレーション describe-instance-types を実行することで、このインスタンスタイプのプロセッサ情報を取得できます。

DB インスタンスのデフォルトのプロセッサ設定に戻す

Example

次の例では、mydbinstance を変更して DB インスタンスクラスをデフォルトのプロセッサ値に戻します。--apply-immediately を使用すると変更はすぐに適用されます。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately オプションを省略します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --use-default-processor-features \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --use-default-processor-features ^
    --apply-immediately
```

DB インスタンスのデフォルトの CPU コア数に戻す

Example

次の例では、mydbinstance を変更して DB インスタンスクラスをデフォルトの CPU コア数に戻します。コアごとスレッドの設定は変更されません。--apply-immediately を使用すると変更はすぐに適用されます。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately オプションを省略します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --processor-features "Name=coreCount,Value=DEFAULT" \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
```

```
    --processor-features "Name=coreCount,Value=DEFAULT" ^
    --apply-immediately
```

DB インスタンスのデフォルトのコアあたりのスレッド数に戻す

Example

次の例では、mydbinstance を変更して DB インスタンスクラスをデフォルトのコアあたりのスレッド数に戻します。CPU コア数の設定は変更されません。--apply-immediately を使用すると変更はすぐに適用されます。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately オプションを省略します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --processor-features "Name=threadsPerCore,Value=DEFAULT" \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --processor-features "Name=threadsPerCore,Value=DEFAULT" ^
    --apply-immediately
```

RDS API

次の Amazon RDS API オペレーションのいずれかを呼び出す際に DB インスタンスのプロセッサ機能を設定できます。

- CreateDBInstance
- ModifyDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

Amazon RDS API を使用して DB インスタンスで DB インスタンスクラスのプロセッサ機能を設定するには、コマンドに ProcessFeatures パラメータを含めます。

パラメータの構文は次のとおりです。

```
ProcessFeatures "Name=coreCount,Value=<value>" "Name=threadsPerCore,Value=<value>"
```

`coreCount` 機能名で CPU コアの数を指定し、`threadsPerCore` 機能名で複数のスレッドを有効にするかどうかを指定します。

特定の DB インスタンスクラスの有効なプロセッサ値を確認するには、DescribeOrderableDBInstanceOptions アクションを実行し、`DBInstanceClass` パラメータでインスタンスクラスを指定します。以下のオペレーションを使用することもできます。

- DescribeDBInstances - 指定された DB インスタンスのプロセッサ情報を示します。
- DescribeDBSnapshots - 指定された DB スナップショットのプロセッサ情報を示します。
- DescribeValidDBInstanceModifications - 指定された DB インスタンスのプロセッサに対する有効な変更を示します。

上記のオペレーションの出力では、次の条件が満たされている場合にのみ、プロセッサ機能の値は null ではありません。

- RDS for Oracle DB インスタンスを使用しています。
- RDS for Oracle DB インスタンスは、プロセッサ値の変更をサポートしています。
- 現在の CPU コアとスレッドの設定は、デフォルト以外の値に設定されます。

上記の条件が満たされない場合、DescribeDBInstances を使用してインスタンスタイプを取得できます。EC2 オペレーション DescribeInstanceTypes を実行することで、このインスタンスタイプのプロセッサ情報を取得できます。

DB インスタンスクラスのハードウェア仕様

このセクションの表で、の Amazon RDS DB インスタンスクラスに関するハードウェアの詳細を示します。

DB インスタンスクラスごとの Amazon RDS DB エンジンサポートについては、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。

トピック

- DB インスタンスクラスのハードウェア用語

- 汎用インスタンスクラスのハードウェア仕様
- メモリ最適化インスタンスクラスのハードウェア仕様
- コンピューティング最適化インスタンスクラスのハードウェア仕様
- バーストパフォーマンスインスタンスクラスのハードウェア仕様

> **Note**
>
> RDS for SQL Server は、第 7 世代のインスタンスクラス (db.m7i や db.r7i など) から CPU の最適化をサポートします。以下に示す vCPU 数は、これらのインスタンスクラスで異なる場合があります。正確な vCPU 数については、「CPU の最適化をサポートする DB インスタンスクラス」を参照してください。

DB インスタンスクラスのハードウェア用語

以下の用語を使用して、DB インスタンスクラスのハードウェア仕様について説明します。

vCPU

仮想中央演算装置 (CPU) の数。仮想 CPU は、DB インスタンスクラスの比較に使用できる容量の単位です。特定のプロセッサを購入またはリースして数か月から数年間使用する代わりに、時間単位で処理能力をレンタルすることができます。私たちの目標は、実際の基盤となるハードウェアの範囲内で、一貫して特定の容量の CPU 能力を使用できるようにすることです。

ECU

Amazon EC2 インスタンスの整数処理能力の相対的測定基準。異なるインスタンスクラス間で開発者が簡単に CPU 能力値を比較できるように、Amazon EC2 コンピュート単位が定義されています。特定のインスタンスに配分されている CPU 量は、これらの EC2 コンピュート単位で明示されます。現在のところ、1 つの ECU で、1.0 - 1.2 GHz 2007 Opteron または 2007 Xeon プロセッサと同等の CPU 能力が提供されます。

メモリ (GiB)

DB インスタンスに割り当てられる RAM (ギガバイナリバイト単位)。通常、メモリと vCPU の比率は一定です。例として、db.r4 インスタンスクラスを使用します。このインスタンスクラスのメモリと vCPU の比率は db.r5 インスタンスクラスと同じですが、db.r5 インスタンスクラスのパフォーマンスは、ほとんどのユースケースで db.r4 インスタンスクラスより安定して優れています。

EBS 最適化

DB インスタンスは、最適化された設定スタックを使用し、I/O 用に専用の容量を追加で提供します。このように最適化することで、I/O と、インスタンスからのその他のトラフィックとの間の競合を最小に抑え、最高のパフォーマンスを実現します。Amazon EBS 最適化インスタンスの詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」を参照してください。

EBS 最適化インスタンスには、ベースライン IOPS レートと最大 IOPS レートがあります。最大 IOPS レートは、DB インスタンスレベルで適用されます。IOPS レートが最大値を超えるように組み合わされた EBS ボリュームのセットは、インスタンスレベルのしきい値を超えることはできません。例えば、特定の DB インスタンスクラスの最大 IOPS が 40,000 であり、64,000 IOPS EBS ボリュームを 4 つアタッチした場合、最大 IOPS は 256,000 ではなく 40,000 になります。各 EC2 インスタンスタイプに固有の IOPS の最大値については、「Amazon EC2 Linux インスタンス用ユーザーガイドの「インスタンスタイプ」を参照してください。

最大 EBS 帯域幅 (Mbps)

最大 EBS 帯域幅 (メガビット/秒)。8 で割ると、メガバイト/秒でのスループットが得られます。

> **Important**
>
> 通常、Amazon RDS DB インスタンスの汎用 SSD (gp2) ボリュームには 250 MiB/秒のスループット制限があります。ただし、スループット制限はボリュームサイズに応じて異なる場合があります。詳細については、Amazon EC2 ユーザーガイドの「Amazon EBS ボリュームの種類」を参照してください。

ネットワーク帯域幅

他の DB インスタンスクラスとの相対的なネットワーク速度。

汎用インスタンスクラスのハードウェア仕様

次の表は、汎用インスタンスクラスのコンピューティング、メモリ、ストレージ、帯域幅の仕様を示しています。

db.m9g – AWS Graviton5 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m9g.48xlarge	192	—	768	EBS 最適化のみ	72,000	100
db.m9g.24xlarge	96	—	384	EBS 最適化のみ	36,000	50
db.m9g.16xlarge	64	—	256	EBS 最適化のみ	24,000	34
db.m9g.12xlarge	48	—	192	EBS 最適化のみ	18,000	25
db.m9g.8xlarge	32	—	128	EBS 最適化のみ	12,000	17
db.m9g.4xlarge*	16	—	64	EBS 最適化のみ	最大 12,000	最大 17
db.m9g.2xlarge*	8	—	32	EBS 最適化のみ	最大 12,000	最大 17
db.m9g.xlarge*	4	—	16	EBS 最適化のみ	最大 12,000	最大 15
db.m9g.large*	2	—	8	EBS 最適化のみ	最大 12,000	最大 15

db.m8g – AWS Graviton4 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m8g.48xlarge	192	—	768	EBS 最適化のみ	40,000	50
db.m8g.24xlarge	96	—	384	EBS 最適化のみ	30,000	40
db.m8g.16xlarge	64	—	256	EBS 最適化のみ	20,000	30
db.m8g.12xlarge	48	—	192	EBS 最適化のみ	15,000	22.5
db.m8g.8xlarge	32	—	128	EBS 最適化のみ	10,000	15
db.m8g.4xlarge*	16	—	64	EBS 最適化のみ	最大 10,000	最大 15
db.m8g.2xlarge*	8	—	32	EBS 最適化のみ	最大 10,000	最大 15
db.m8g.xlarge*	4	—	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.m8g.large*	2	—	8	EBS 最適化のみ	最大 10,000	最大 12.5

db.m8i — インテル Xeon 6 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m8i.metal-96xl	384	192	2	—	1,536	EBS のみ	80,000	100
db.m8i.metal-48xl	192	96	1	—	768	EBS のみ	60,000	75
db.m8i.96xlarge	384	—	—	—	1,536	EBS のみ	80,000	100
db.m8i.48xlarge	192	—	—	—	768	EBS のみ	60,000	75
db.m8i.32xlarge	128	—	—	—	512	EBS のみ	40,000	50
db.m8i.24xlarge	96	—	—	—	384	EBS のみ	30,000	40
db.m8i.16xlarge	64	—	—	—	256	EBS のみ	20,000	30
db.m8i.12xlarge	48	—	—	—	192	EBS のみ	15,000	22.5
db.m8i.8xlarge	32	—	—	—	128	EBS のみ	10,000	15
db.m8i.4xlarge	16	—	—	—	64	EBS のみ	最大 10,000	最大 15
db.m8i.2xlarge	8	—	—	—	32	EBS のみ	最大 10,000	最大 15
db.m8i.xlarge	4	—	—	—	16	EBS のみ	最大 10,000	最大 12.5
db.m8i.large	2	—	—	—	8	EBS のみ	最大 10,000	最大 12.5

db.m7i – 第 4 世代インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m7i.metal-48xl	192	96	2	—	768	EBS 最適化のみ	40,000	50
db.m7i.metal-24xl	96	48	1	—	384	EBS 最適化のみ	30,000	37.5
db.m7i.48xlarge	192	—	—	—	768	EBS 最適化のみ	40,000	50
db.m7i.24xlarge	96	—	—	—	384	EBS 最適化のみ	30,000	37.5
db.m7i.16xlarge	64	—	—	—	256	EBS 最適化のみ	20,000	25
db.m7i.12xlarge	48	—	—	—	192	EBS 最適化のみ	15,000	18.75
db.m7i.8xlarge	32	—	—	—	128	EBS 最適化のみ	10,000	12.5
db.m7i.4xlarge	16	—	—	—	64	EBS 最適化のみ	最大 10,000	最大 12.5
db.m7i.2xlarge	8	—	—	—	32	EBS 最適化のみ	最大 10,000	最大 12.5
db.m7i.xlarge	4	—	—	—	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.m7i.large	2	—	—	—	8	EBS 最適化のみ	最大 10,000	最大 12.5

db.m7g – AWS Graviton3 プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m7g.16xlarge	64	—	256	EBS 最適化のみ	20,000	30
db.m7g.12xlarge	48	—	192	EBS 最適化のみ	15,000	22.5
db.m7g.8xlarge	32	—	128	EBS 最適化のみ	10,000	15
db.m7g.4xlarge	16	—	64	EBS 最適化のみ	最大 10,000	最大 15
db.m7g.2xlarge*	8	—	32	EBS 最適化のみ	最大 10,000	最大 15
db.m7g.xlarge*	4	—	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.m7g.large*	2	—	8	EBS 最適化のみ	最大 10,000	最大 12.5

db.m6g – AWS Graviton2 プロセッサを搭載した汎用インスタンスクラス。

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6g.16xlarge	64	—	256	EBS 最適化のみ	19,000	25
db.m6g.12xlarge	48	—	192	EBS 最適化のみ	14,250	20

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6g.8xlarge	32	—	128	EBS 最適化のみ	9,500	12
db.m6g.4xlarge	16	—	64	EBS 最適化のみ	4,750	最大 10
db.m6g.2xlarge*	8	—	32	EBS 最適化のみ	最大 4,750	最大 10
db.m6g.xlarge*	4	—	16	EBS 最適化のみ	最大 4,750	最大 10
db.m6g.large*	2	—	8	EBS 最適化のみ	最大 4,750	最大 10

db.m6gd — AWS Graviton2 プロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6gd.16xlarge	64	—	256	2 x 1,900 NVMe SSD	19,000	25
db.m6gd.12xlarge	48	—	192	2 x 1,425 NVMe SSD	14,250	20
db.m6gd.8xlarge	32	—	128	1 x 1,900 NVMe SSD	9,500	12
db.m6gd.4xlarge*	16	—	64	1 x 950 NVMe SSD	4,750	最大 10

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6gd.2xlarge*	8	—	32	1 x 474 NVMe SSD	最大 4,750	最大 10
db.m6gd.xlarge*	4	—	16	1 x 237 NVMe SSD	最大 4,750	最大 10
db.m6gd.large*	2	—	8	1 x 118 NVMe SSD	最大 4,750	最大 10

db.m6id — 第 3 世代インテル Xeon スケーラブルプロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	vCP	物理コア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6id.metal	128	64	2	—	512	4 x 1,900 NVMe SSD	40,000	50
db.m6id.32xlarge	128	—	—	—	512	4 x 1,900 NVMe SSD	40,000	50
db.m6id.24xlarge	96	—	—	—	384	4 x 1,425 NVMe SSD	30,000	37.5
db.m6id.16xlarge	64	—	—	—	256	2 x 1,900 NVMe SSD	20,000	25
db.m6id.12xlarge	48	—	—	—	192	2 x 1,425 NVMe SSD	15,000	18.75
db.m6id.8xlarge	32	—	—	—	128	1 x 1,900 NVMe SSD	10,000	12.5

インスタンスクラス	vCP	物理コア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6id.4xlarge*	16	—	—	—	64	1 x 950 NVMe SSD	最大 10,000	最大 12.5
db.m6id.2xlarge*	8	—	—	—	32	1 x 474 NVMe SSD	最大 10,000	最大 12.5
db.m6id.xlarge*	4	—	—	—	16	1 x 237 NVMe SSD	最大 10,000	最大 12.5
db.m6id.large*	2	—	—	—	8	1 x 118 NVMe SSD	最大 10,000	最大 12.5

db.m6idn — 第 3 世代インテル Xeon スケーラブルプロセッサ、SSD ストレージ、ネットワーク最適化を搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6idn.32xlarge	128	—	512	4 x 1,900 NVMe SSD	100,000	200
db.m6idn.24xlarge	96	—	384	4 x 1,425 NVMe SSD	75,000	150
db.m6idn.16xlarge	64	—	256	2 x 1,900 NVMe SSD	50,000	100
db.m6idn.12xlarge	48	—	192	2 x 1,425 NVMe SSD	37,500	75
db.m6idn.8xlarge	32	—	128	1 x 1,900 NVMe SSD	25,000	50

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6idn.4xlarge*	16	—	64	1 x 950 NVMe SSD	最大25,000	最大 50
db.m6idn.2xlarge*	8	—	32	1 x 474 NVMe SSD	最大25,000	最大 40
db.m6idn.xlarge*	4	—	16	1 x 237 NVMe SSD	最大25,000	最大 30
db.m6idn.large*	2	—	8	1 x 118 NVMe SSD	最大25,000	最大 25

db.m6in — 第 3 世代インテル Xeon スケーラブルプロセッサとネットワーク最適化を搭載した汎用インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6in.metal	128	64	2	—	512	EBS 最適化のみ	100,000	200
db.m6in.32xlarge	128			—	512	EBS 最適化のみ	100,000	200
db.m6in.24xlarge	96			—	384	EBS 最適化のみ	75,000	150
db.m6in.16xlarge	64			—	256	EBS 最適化のみ	50,000	100

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6in.12xlarge	48			—	192	EBS 最適化のみ	37,500	75
db.m6in.8xlarge	32			—	128	EBS 最適化のみ	25,000	50
db.m6in.4xlarge*	16			—	64	EBS 最適化のみ	最大25,000	最大 50
db.m6in.2xlarge*	8			—	32	EBS 最適化のみ	最大25,000	最大 40
db.m6in.xlarge*	4			—	16	EBS 最適化のみ	最大25,000	最大 30
db.m6in.large*	2			—	8	EBS 最適化のみ	最大25,000	最大 25

db.m6in — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6i.metal	128	64	2	—	512	EBS 最適化のみ	40,000	50

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m6i.32xlarge	128			—	512	EBS 最適化のみ	40,000	50
db.m6i.24xlarge	96			—	384	EBS 最適化のみ	30,000	37.5
db.m6i.16xlarge	64			—	256	EBS 最適化のみ	20,000	25
db.m6i.12xlarge	48			—	192	EBS 最適化のみ	15,000	18.75
db.m6i.8xlarge	32			—	128	EBS 最適化のみ	10,000	12.5
db.m6i.4xlarge*	16			—	64	EBS 最適化のみ	最大 10,000	最大 12.5
db.m6i.2xlarge*	8			—	32	EBS 最適化のみ	最大 10,000	最大 12.5
db.m6i.xlarge*	4			—	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.m6i.large*	2			—	8	EBS 最適化のみ	最大 10,000	最大 12.5

db.m5d — インテル Xeon Platinum プロセッサと SSD ストレージを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m5d.24xlarge	96	345	384	4 x 900 NVMe SSD	19,000	25
db.m5d.16xlarge	64	262	256	4 x 600 NVMe SSD	13,600	20
db.m5d.12xlarge	48	173	192	2 x 900 NVMe SSD	9,500	12
db.m5d.8xlarge	32	131	128	2 x 600 NVMe SSD	6,800	10
db.m5d.4xlarge	16	61	64	2 x 300 NVMe SSD	4,750	最大 10
db.m5d.2xlarge*	8	31	32	1 x 300 NVMe SSD	最大 4,750	最大 10
db.m5d.xlarge*	4	15	16	1 x 150 NVMe SSD	最大 4,750	最大 10
db.m5d.large*	2	10	8	1 x 75 NVMe SSD	最大 4,750	最大 10

db.m5 — インテル Xeon Platinum プロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m5.24xlarge	96	345	384	EBS 最適化のみ	19,000	25

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m5.16xlarge	64	262	256	EBS 最適化のみ	13,600	20
db.m5.12xlarge	48	173	192	EBS 最適化のみ	9,500	12
db.m5.8xlarge	32	131	128	EBS 最適化のみ	6,800	10
db.m5.4xlarge	16	61	64	EBS 最適化のみ	4,750	最大 10
db.m5.2xlarge*	8	31	32	EBS 最適化のみ	最大 4,750	最大 10
db.m5.xlarge*	4	15	16	EBS 最適化のみ	最大 4,750	最大 10
db.m5.large*	2	10	8	EBS 最適化のみ	最大 4,750	最大 10

db.m4 – インテル Xeon スケーラブルプロセッサを搭載した汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m4.16xlarge	64	188	256	EBS 最適化のみ	10,000	25
db.m4.10xlarge	40	124.5	160	EBS 最適化のみ	4,000	10

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m4.4xlarge	16	53.5	64	EBS 最適化のみ	2,000	高
db.m4.2xlarge	8	25.5	32	EBS 最適化のみ	1,000	高
db.m4.xlarge	4	13	16	EBS 最適化のみ	750	高
db.m4.large	2	6.5	8	EBS 最適化のみ	450	中

db.m3 – 汎用インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m3.2xlarge	8	26	30	EBS 最適化のみ	1,000	高
db.m3.xlarge	4	13	15	EBS 最適化のみ	500	高
db.m3.large	2	6.5	7.5	EBS のみ	—	中
db.m3.medium	1	3	3.75	EBS のみ	—	中

* これらのインスタンスクラスは、24 時間に最低 1 回、30 分間にわたって、最大パフォーマンスをサポートできます。基礎となる EC2 インスタンスタイプのベースラインパフォーマンスの詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」を参照してください。

メモリ最適化インスタンスクラスのハードウェア仕様

次の表は、メモリ最適化インスタンスクラスのコンピューティング、メモリ、ストレージ、帯域幅の仕様を示しています。

db.z1d – メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.z1d.12xlarge	48	271	384	2 x 900 NVMe SSD	19,000	25
db.z1d.6xlarge	24	134	192	1 x 900 NVMe SSD	9,500	12
db.z1d.3xlarge	12	75	96	1 x 450 NVMe SSD	4,750	最大 10
db.z1d.2xlarge	8	53	64	1 x 300 NVMe SSD	3,170	最大 10
db.z1d.xlarge*	4	28	32	1 x 150 NVMe SSD	最大 3,170	最大 10
db.z1d.large*	2	15	16	1 x 75 NVMe SSD	最大 3,170	最大 10

db.x2g – AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2g.16xlarge	64	—	1024	EBS 最適化のみ	19,000	25

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2g.12xlarge	48	—	768	EBS 最適化のみ	14,250	20
db.x2g.8xlarge	32	—	512	EBS 最適化のみ	9,500	12
db.x2g.4xlarge	16	—	256	EBS 最適化のみ	4,750	最大 10
db.x2g.2xlarge	8	—	128	EBS 最適化のみ	最大 4,750	最大 10
db.x2g.xlarge	4	—	64	EBS 最適化のみ	最大 4,750	最大 10
db.x2g.large	2	—	32	EBS 最適化のみ	最大 4,750	最大 10

db.x2idn – 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2idn.metal	128	64	2	—	2,048	2 x 1,900 NVMe SSD	80,000	100
db.x2idn.32xlarge	128	—	—	—	2,048	2 x 1,900 NVMe SSD	80,000	100
db.x2idn.24xlarge	96	—	—	—	1,536	2 x 1,425 NVMe SSD	60,000	75

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2idn.16xlarge	64	—	—	—	1,024	1 x 1,900 NVMe SSD	40,000	50

db.x2iedn – 第 3 世代インテル Xeon スケーラブルプロセッサを搭載し、ローカル NVMe ベースの SSD を備えたメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2iedn.metal	128	64	2	—	4,096	2 x 1,900 NVMe SSD	80,000	100
db.x2iedn.32xlarge	128	—	—	—	4,096	2 x 1,900 NVMe SSD	80,000	100
db.x2iedn.24xlarge	96	—	—	—	3,072	2 x 1,425 NVMe SSD	60,000	75
db.x2iedn.16xlarge	64	—	—	—	2,048	1 x 1,900 NVMe SSD	40,000	50
db.x2iedn.8xlarge	32	—	—	—	1,024	1 x 950 NVMe SSD	20,000	25

インスタンスクラス	vCPU	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2iedn.4xlarge	16	—	—	—	512	1 x 475 NVMe SSD	最大 20,000	最大 25
db.x2iedn.2xlarge	8	—	—	—	256	1 x 237 NVMe SSD	最大 20,000	最大 25
db.x2iedn.xlarge	4	—	—	—	128	1 x 118 NVMe SSD	最大 20,000	最大 25

db.x2iezn – 第 2 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2iezn.metal	48	24	2	—	1,536	EBS 最適化のみ	19,000	100
db.x2iezn.12xlarge	48	—	—	—	1,536	EBS 最適化のみ	19,000	100
db.x2iezn.8xlarge	32	—	—	—	1,024	EBS 最適化のみ	12,000	75
db.x2iezn.6xlarge	24	—	—	—	768	EBS 最適化のみ	最大 9,500	50

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x2iezn.4xlarge	16	—	—	—	512	EBS 最適化のみ	最大 4,750	最大 25
db.x2iezn.2xlarge	8	—	—	—	256	EBS 最適化のみ	最大 3,170	最大 25

db.x1e – メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x1e.32xlarge	128	340	3,904	EBS 最適化のみ	14,000	25
db.x1e.16xlarge	64	179	1,952	EBS 最適化のみ	7,000	10
db.x1e.8xlarge	32	91	976	EBS 最適化のみ	3,500	最大 10
db.x1e.4xlarge	16	47	488	EBS 最適化のみ	1,750	最大 10
db.x1e.2xlarge	8	23	244	EBS 最適化のみ	1,000	最大 10
db.x1e.xlarge	4	12	122	EBS 最適化のみ	500	最大 10

db.x1 – メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.x1.32xlarge	128	349	1,952	EBS 最適化のみ	14,000	25
db.x1.16xlarge	64	174.5	976	EBS 最適化のみ	7,000	10

db.m8gd – AWS Graviton4 プロセッサと SSD ストレージを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m8gd.48xlarge	192	—	768	6 x 1,900 NVMe SSD	40,000	50
db.m8gd.24xlarge	96	—	384	3 x 1,900 NVMe SSD	30,000	40
db.m8gd.16xlarge	64	—	256	2 x 1,900 NVMe SSD	20,000	30
db.m8gd.12xlarge	48	—	192	3 x 950 NVMe SSD	15,000	22.5
db.m8gd.8xlarge	32	—	128	1 x 1,900 NVMe SSD	10,000	15
db.m8gd.4xlarge	16	—	64	1 x 950 NVMe SSD	最大 10,000	最大 15
db.m8gd.2xlarge	8	—	32	1 x 474 NVMe SSD	最大 10,000	最大 15

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.m8gd.xlarge	4	—	16	1 x 237 NVMe SSD	最大 10,000	最大 12.5
db.m8gd.large	2	—	8	1 x 118 NVMe SSD	最大 10,000	最大 12.5

db.r8gd – AWS Graviton4 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8gd.48xlarge	192	—	1536	6 x 1,900 NVMe SSD	40,000	50
db.r8gd.24xlarge	96	—	768	3 x 1,900 NVMe SSD	30,000	40
db.r8gd.16xlarge	64	—	512	2 x 1,900 NVMe SSD	20,000	30
db.r8gd.12xlarge	48	—	384	3 x 950 NVMe SSD	15,000	22.5
db.r8gd.8xlarge	32	—	256	1 x 1,900 NVMe SSD	10,000	15
db.r8gd.4xlarge	16	—	128	1 x 950 NVMe SSD	最大 10,000	最大 15
db.r8gd.2xlarge	8	—	64	1 x 474 NVMe SSD	最大 10,000	最大 15

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8gd.xlarge	4	—	32	1 x 237 NVMe SSD	最大 10,000	最大 12.5
db.r8gd.large	2	—	16	1 x 118 NVMe SSD	最大 10,000	最大 12.5

db.r8g – AWS Graviton4 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8g.48xlarge	192	—	1536	EBS 最適化のみ	40,000	50
db.r8g.24xlarge	96	—	768	EBS 最適化のみ	30,000	40
db.r8g.16xlarge	64	—	512	EBS 最適化のみ	20,000	30
db.r8g.12xlarge	48	—	384	EBS 最適化のみ	15,000	22.5
db.r8g.8xlarge	32	—	256	EBS 最適化のみ	10,000	15
db.r8g.4xlarge*	16	—	128	EBS 最適化のみ	最大 10,000	最大 15
db.r8g.2xlarge*	8	—	64	EBS 最適化のみ	最大 10,000	最大 15

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8g.xlarge*	4	—	32	EBS 最適化のみ	最大 10,000	最大 12.5
db.r8g.large*	2	—	16	EBS 最適化のみ	最大 10,000	最大 12.5

db.r8i – インテル Xeon 6 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8i.metal-96xl	384	192	2	—	3,072	EBS のみ	80,000	100
db.r8i.metal-48xl	192	96	1	—	1,536	EBS のみ	60,000	75
db.r8i.96xlarge	384	—	—	—	3,072	EBS のみ	80,000	100
db.r8i.48xlarge	192	—	—	—	1,536	EBS のみ	60,000	75
db.r8i.32xlarge	128	—	—	—	1,024	EBS のみ	40,000	50
db.r8i.24xlarge	96	—	—	—	768	EBS のみ	30,000	40
db.r8i.16xlarge	64	—	—	—	512	EBS のみ	20,000	30
db.r8i.12xlarge	48	—	—	—	384	EBS のみ	15,000	22.5
db.r8i.8xlarge	32	—	—	—	256	EBS のみ	10,000	15
db.r8i.4xlarge	16	—	—	—	128	EBS のみ	最大 10,000	最大 15

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r8i.2xlarge	8	—	—	—	64	EBS のみ	最大 10,000	最大 15
db.r8i.xlarge	4	—	—	—	32	EBS のみ	最大 10,000	最大 12.5
db.r8i.large	2	—	—	—	16	EBS のみ	最大 10,000	最大 12.5

db.r7i – 第 4 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r7i.metal-48xl	192	96	2	—	1536	192	EBS 最適化のみ	40,000	50
db.r7i.metal-24xl	96	48	1	—	768	96	EBS 最適化のみ	30,000	37.5
db.r7i.48xlarge	192	—	—	—	1536	192	EBS 最適化のみ	40,000	50
db.r7i.24xlarge	96	—	—	—	768	96	EBS 最適化のみ	30,000	37.5

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r7i.16xlarge	64	—	—	—	512	64	EBS 最適化のみ	20,000	25
db.r7i.12xlarge	48	—	—	—	384	48	EBS 最適化のみ	15,000	18.75
db.r7i.8xlarge.tpc2.mem3x	32	—	—	—	768	96	EBS 最適化のみ	30,000	12.5
db.r7i.8xlarge.tpc2.mem2x	32	—	—	—	512	64	EBS 最適化のみ	20,000	12.5
db.r7i.8xlarge	32	—	—	—	256	32	EBS 最適化のみ	10,000	12.5
db.r7i.6xlarge.tpc2.mem4x	24	—	—	—	768	96	EBS 最適化のみ	30,000	最大 12.5
db.r7i.6xlarge.tpc2.mem2x	24	—	—	—	384	48	EBS 最適化のみ	15,000	最大 12.5
db.r7i.4xlarge.tpc2.mem4x	16	—	—	—	512	64	EBS 最適化のみ	20,000	最大 12.5
db.r7i.4xlarge.tpc2.mem3x	16	—	—	—	384	48	EBS 最適化のみ	15,000	最大 12.5
db.r7i.4xlarge.tpc2.mem2x	16	—	—	—	256	32	EBS 最適化のみ	10,000	最大 12.5
db.r7i.4xlarge	16	—	—	—	128	16	EBS 最適化のみ	最大 10,000	最大 12.5

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r7i.3xlarge.tpc2.mem4x	12	—	—	—	384	48	EBS 最適化のみ	15,000	最大 12.5
db.r7i.2xlarge.tpc2.mem8x	8	—	—	—	512	64	EBS 最適化のみ	20,000	最大 12.5
db.r7i.2xlarge.tpc2.mem4x	8	—	—	—	256	32	EBS 最適化のみ	10,000	最大 12.5
db.r7i.2xlarge	8	—	—	—	64	8	EBS 最適化のみ	最大 10,000	最大 12.5
db.r7i.xlarge.tpc2.mem4x	4	—	—	—	128	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.r7i.xlarge.tpc2.mem2x	4	—	—	—	64	8	EBS 最適化のみ	最大 10,000	最大 12.5
db.r7i.xlarge	4	—	—	—	32	4	EBS 最適化のみ	最大 10,000	最大 12.5
db.r7i.large	2	—	—	—	16	2	EBS 最適化のみ	最大 10,000	最大 12.5

db.r7g – AWS Graviton3 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r7g.16xlarge	64	—	512	EBS 最適化のみ	20,000	30
db.r7g.12xlarge	48	—	384	EBS 最適化のみ	15,000	22.5
db.r7g.8xlarge	32	—	256	EBS 最適化のみ	10,000	15
db.r7g.4xlarge	16	—	128	EBS 最適化のみ	最大 10,000	最大 15
db.r7g.2xlarge*	8	—	64	EBS 最適化のみ	最大 10,000	最大 15
db.r7g.xlarge*	4	—	32	EBS 最適化のみ	最大 10,000	最大 12.5
db.r7g.large*	2	—	16	EBS 最適化のみ	最大 10,000	最大 12.5

db.r6g – AWS Graviton2 プロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6g.16xlarge	64	—	512	EBS 最適化のみ	19,000	25
db.r6g.12xlarge	48	—	384	EBS 最適化のみ	14,250	20

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6g.8xlarge	32	—	256	EBS 最適化のみ	9,500	12
db.r6g.4xlarge	16	—	128	EBS 最適化のみ	4,750	最大 10
db.r6g.2xlarge*	8	—	64	EBS 最適化のみ	最大 4,750	最大 10
db.r6g.xlarge*	4	—	32	EBS 最適化のみ	最大 4,750	最大 10
db.r6g.large*	2	—	16	EBS 最適化のみ	最大 4,750	最大 10

db.r6gd — AWS Graviton2 プロセッサと SSD ストレージを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6gd.16xlarge	64	—	512	2 x 1,900 NVMe SSD	19,000	25
db.r6gd.12xlarge	48	—	384	2 x 1,425 NVMe SSD	14,250	20
db.r6gd.8xlarge	32	—	256	1 x 1,900 NVMe SSD	9,500	12
db.r6gd.4xlarge	16	—	128	1 x 950 NVMe SSD	4,750	最大 10

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6gd.2xlarge	8	—	64	1 x 474 NVMe SSD	最大 4,750	最大 10
db.r6gd.xlarge	4	—	32	1 x 237 NVMe SSD	最大 4,750	最大 10
db.r6gd.large	2	—	16	1 x 118 NVMe SSD	最大 4,750	最大 10

db.r6id – 第 3 世代インテル Xeon スケーラブルプロセッサおよび SSD ストレージを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6id.metal	128	64	2	—	1,024	4 x 1,900 NVMe SSD	40,000	50
db.r6id.32xlarge	128	—	—	—	1,024	4x1900 NVMe SSD	40,000	50
db.r6id.24xlarge	96	—	—	—	768	4x1425 NVMe SSD	30,000	37.5
db.r6id.16xlarge	64	—	—	—	512	2x1900 NVMe SSD	20,000	25
db.r6id.12xlarge	48	—	—	—	384	2x1425 NVMe SSD	15,000	18.75
db.r6id.8xlarge	32	—	—	—	256	1x1900 NVMe SSD	10,000	12.5

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6id.4xlarge*	16	—	—	—	128	1x950 NVMe SSD	最大 10,000	最大 12.5
db.r6id.2xlarge*	8	—	—	—	64	1x474 NVMe SSD	最大 10,000	最大 12.5
db.r6id.xlarge*	4	—	—	—	32	1x237 NVMe SSD	最大 10,000	最大 12.5
db.r6id.large*	2	—	—	—	16	1x118 NVMe SSD	最大 10,000	最大 12.5

db.r6idn – 第 3 世代インテル Xeon スケーラブルプロセッサ、SSD ストレージ、ネットワーク最適化を搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6idn.32xlarge	128	—	1,024	4x1900 NVMe SSD	100,000	200
db.r6idn.24xlarge	96	—	768	4x1425 NVMe SSD	75,000	150
db.r6idn.16xlarge	64	—	512	2x1900 NVMe SSD	50,000	100
db.r6idn.12xlarge	48	—	384	2x1425 NVMe SSD	37,500	75
db.r6idn.8xlarge	32	—	256	1x1900 NVMe SSD	25,000	50

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6idn.4xlarge*	16	—	128	1x950 NVMe SSD	最大25,000	最大 50
db.r6idn.2xlarge*	8	—	64	1x474 NVMe SSD	最大25,000	最大 40
db.r6idn.xlarge*	4	—	32	1x237 NVMe SSD	最大25,000	最大 30
db.r6idn.large*	2	—	16	1x118 NVMe SSD	最大25,000	最大 25

db.r6in – 第 3 世代インテル Xeon スケーラブルプロセッサとネットワーク最適化を搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6in.metal	128	64	2	—	1,024	EBS 最適化のみ	100,000	200
db.r6in.32xlarge	128	—	—	—	1,024	EBS 最適化のみ	100,000	200
db.r6in.24xlarge	96	—	—	—	768	EBS 最適化のみ	75,000	150
db.r6in.16xlarge	64	—	—	—	512	EBS 最適化のみ	50,000	100

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6in.12xlarge	48	—	—	—	384	EBS 最適化のみ	37,500	75
db.r6in.8xlarge	32	—	—	—	256	EBS 最適化のみ	25,000	50
db.r6in.4xlarge*	16	—	—	—	128	EBS 最適化のみ	最大25,000	最大 50
db.r6in.2xlarge*	8	—	—	—	64	EBS 最適化のみ	最大25,000	最大 40
db.r6in.xlarge*	4	—	—	—	32	EBS 最適化のみ	最大25,000	最大 30
db.r6in.large*	2	—	—	—	16	EBS 最適化のみ	最大25,000	最大 25

db.r6i – 高メモリ、ストレージ、I/O 用に事前設定された Oracle メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6i.8xlarge.tpc2.mem4x	32	—	1024	EBS 最適化のみ	40,000	50
db.r6i.8xlarge.tpc2.mem3x	32	—	768	EBS 最適化のみ	30,000	37.5

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6i.6xlarge.tpc2.mem4x	24	—	768	EBS 最適化のみ	30,000	37.5
db.r6i.4xlarge.tpc2.mem4x	16	—	512	EBS 最適化のみ	20,000	25
db.r6i.4xlarge.tpc2.mem3x	16	—	384	EBS 最適化のみ	15,000	18.75
db.r6i.4xlarge.tpc2.mem2x	16	—	256	EBS 最適化のみ	10,000	12.5
db.r6i.2xlarge.tpc2.mem8x	8	—	512	EBS 最適化のみ	20,000	25
db.r6i.2xlarge.tpc2.mem4x	8	—	256	EBS 最適化のみ	10,000	12.5
db.r6i.2xlarge.tpc1.mem2x	8	—	128	EBS 最適化のみ	最大 10,000	12.5
db.r6i.xlarge.tpc2.mem4x	4	—	128	EBS 最適化のみ	最大 10,000	12.5
db.r6i.xlarge.tpc2.mem2x	4	—	64	EBS 最適化のみ	最大 10,000	12.5
db.r6i.large.tpc1.mem2x	2	—	32	EBS 最適化のみ	最大 10,000	12.5

db.r6i — 第 3 世代インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCP	プロセッサコア	ソケット	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r6i.metal	128	64	2	—	1,024	256	EBS 最適化のみ	40,000	50
db.r6i.32xlarge	128	—	—	—	1,024	256	EBS 最適化のみ	40,000	50
db.r6i.24xlarge	96	—	—	—	768	192	EBS 最適化のみ	30,000	37.5
db.r6i.16xlarge	64	—	—	—	512	128	EBS 最適化のみ	20,000	25
db.r6i.12xlarge	48	—	—	—	384	96	EBS 最適化のみ	15,000	18.75
db.r6i.8xlarge	32	—	—	—	256	64	EBS 最適化のみ	10,000	12.5
db.r6i.4xlarge*	16	—	—	—	128	32	EBS 最適化のみ	最大 10,000	最大 12.5
db.r6i.2xlarge*	8	—	—	—	64	16	EBS 最適化のみ	最大 10,000	最大 12.5
db.r6i.xlarge*	4	—	—	—	32	8	EBS 最適化のみ	最大 10,000	最大 12.5
db.r6i.large*	2	—	—	—	16	4	EBS 最適化のみ	最大 10,000	最大 12.5

db.r5d — インテル Xeon Platinum プロセッサと SSD ストレージを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5d.24xlarge	96	347	768	4 x 900 NVMe SSD	19,000	25
db.r5d.16xlarge	64	264	512	4 x 600 NVMe SSD	13,600	20
db.r5d.12xlarge	48	173	384	2 x 900 NVMe SSD	9,500	12
db.r5d.8xlarge	32	132	256	2 x 600 NVMe SSD	6,800	10
db.r5d.4xlarge	16	71	128	2 x 300 NVMe SSD	4,750	最大 10
db.r5d.2xlarge*	8	38	64	1 x 300 NVMe SSD	最大 4,750	最大 10
db.r5d.xlarge*	4	19	32	1 x 150 NVMe SSD	最大 4,750	最大 10
db.r5d.large*	2	10	16	1 x 75 NVMe SSD	最大 4,750	最大 10

db.r5b – 高メモリ、ストレージ、I/O 用に事前設定された Oracle メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5b.8xlarge.tpc2.mem3x	32	—	768	EBS 最適化のみ	60,000	25
db.r5b.6xlarge.tpc2.mem4x	24	—	768	EBS 最適化のみ	60,000	25
db.r5b.4xlarge.tpc2.mem4x	16	—	512	EBS 最適化のみ	40,000	20
db.r5b.4xlarge.tpc2.mem3x	16	—	384	EBS 最適化のみ	30,000	12
db.r5b.4xlarge.tpc2.mem2x	16	—	256	EBS 最適化のみ	20,000	10
db.r5b.2xlarge.tpc2.mem8x	8	—	512	EBS 最適化のみ	40,000	20
db.r5b.2xlarge.tpc2.mem4x	8	—	256	EBS 最適化のみ	20,000	10
db.r5b.2xlarge.tpc1.mem2x	8	—	128	EBS 最適化のみ	10,000	最大 10
db.r5b.xlarge.tpc2.mem4x	4	—	128	EBS 最適化のみ	10,000	最大 10
db.r5b.xlarge.tpc2.mem2x	4	—	64	EBS 最適化のみ	最大 10,000	最大 10
db.r5b.large.tpc1.mem2x	2	—	32	EBS 最適化のみ	最大 10,000	最大 10

db.r5b — インテル Xeon Platinum プロセッサと EBS 最適化を搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5b.24xlarge	96	347	768	192	EBS 最適化のみ	60,000	25
db.r5b.16xlarge	64	264	512	128	EBS 最適化のみ	40,000	20
db.r5b.12xlarge	48	173	384	96	EBS 最適化のみ	30,000	12
db.r5b.8xlarge	32	132	256	64	EBS 最適化のみ	20,000	10
db.r5b.4xlarge	16	71	128	32	EBS 最適化のみ	10,000	最大 10
db.r5b.2xlarge*	8	38	64	16	EBS 最適化のみ	最大 10,000	最大 10
db.r5b.xlarge*	4	19	32	8	EBS 最適化のみ	最大 10,000	最大 10
db.r5b.large*	2	10	16	4	EBS 最適化のみ	最大 10,000	最大 10

db.r5 – 高メモリ、ストレージ、I/O 用に事前設定された Oracle メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5.12xlarge.tpc2.mem2x	48	—	768	EBS 最適化のみ	19,000	25

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5.8xlarge.tpc2.mem3x	32	—	768	EBS 最適化のみ	19,000	25
db.r5.6xlarge.tpc2.mem4x	24	—	768	EBS 最適化のみ	19,000	25
db.r5.4xlarge.tpc2.mem4x	16	—	512	EBS 最適化のみ	13,600	20
db.r5.4xlarge.tpc2.mem3x	16	—	384	EBS 最適化のみ	9,500	12
db.r5.4xlarge.tpc2.mem2x	16	—	256	EBS 最適化のみ	6,800	10
db.r5.2xlarge.tpc2.mem8x	8	—	512	EBS 最適化のみ	13,600	20
db.r5.2xlarge.tpc2.mem4x	8	—	256	EBS 最適化のみ	6,800	10
db.r5.2xlarge.tpc1.mem2x	8	—	128	EBS 最適化のみ	4,750	最大 10
db.r5.xlarge.tpc2.mem4x	4	—	128	EBS 最適化のみ	4,750	最大 10
db.r5.xlarge.tpc2.mem2x	4	—	64	EBS 最適化のみ	最大 4,750	最大 10
db.r5.large.tpc1.mem2x	2	—	32	EBS 最適化のみ	最大 4,750	最大 10

db.r5 – メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	正規化された単位	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r5.24xlarge	96	347	768	192	EBS 最適化のみ	19,000	25
db.r5.16xlarge	64	264	512	128	EBS 最適化のみ	13,600	20
db.r5.12xlarge	48	173	384	96	EBS 最適化のみ	9,500	10
db.r5.8xlarge	32	132	256	64	EBS 最適化のみ	6,800	10
db.r5.4xlarge	16	71	128	32	EBS 最適化のみ	4,750	最大 10
db.r5.2xlarge*	8	38	64	16	EBS 最適化のみ	最大 4,750	最大 10
db.r5.xlarge*	4	19	32	8	EBS 最適化のみ	最大 4,750	最大 10
db.r5.large*	2	10	16	4	EBS 最適化のみ	最大 4,750	最大 10

db.r4 — インテル Xeon スケーラブルプロセッサを搭載したメモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r4.16xlarge	64	195	488	EBS 最適化のみ	14,000	25

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r4.8xlarge	32	99	244	EBS 最適化のみ	7,000	10
db.r4.4xlarge	16	53	122	EBS 最適化のみ	3,500	最大 10
db.r4.2xlarge	8	27	61	EBS 最適化のみ	1,700	最大 10
db.r4.xlarge	4	13.5	30.5	EBS 最適化のみ	850	最大 10
db.r4.large	2	7	15.25	EBS 最適化のみ	425	最大 10

db.r3 – メモリ最適化インスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.r3.8xlarge**	32	104	244	EBS のみ	—	10
db.r3.4xlarge	16	52	122	EBS 最適化のみ	2,000	高
db.r3.2xlarge	8	26	61	EBS 最適化のみ	1,000	高
db.r3.xlarge	4	13	30.5	EBS 最適化のみ	500	中
db.r3.large	2	6.5	15.25	EBS 最適化のみ	—	中

* これらのインスタンスクラスは、24 時間に最低 1 回、30 分間にわたって、最大パフォーマンスをサポートできます。基礎となる EC2 インスタンスタイプのベースラインパフォーマンスの詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」を参照してください。

** r3.8xlarge DB インスタンスクラスには専用 EBS 帯域幅がないため、EBS 最適化は提供されません。これらのインスタンスクラスでは、ネットワークトラフィックと Amazon EBS トラフィックは同じ 10 ギガビットネットワークインターフェイスで共有されます。

コンピューティング最適化インスタンスクラスのハードウェア仕様

次の表は、コンピューティング最適化インスタンスクラスのコンピューティング、メモリ、ストレージ、帯域幅の仕様を示しています。

db.c6gd — コンピューティング最適化インスタンスクラス (マルチ AZ DB クラスター配置のみ)

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.c6gd.16xlarge	64	—	128	2 x 1,900 NVMe SSD	19,000	25
db.c6gd.12xlarge	48	—	96	2 x 1,425 NVMe SSD	14,250	20
db.c6gd.8xlarge	32	—	64	1 x 1,900 NVMe SSD	9,500	12
db.c6gd.4xlarge	16	—	32	1 x 950 NVMe SSD	4,750	最大 10
db.c6gd.2xlarge	8	—	16	1 x 474 NVMe SSD	最大 4,750	最大 10
db.c6gd.xlarge	4	—	8	1 x 237 NVMe SSD	最大 4,750	最大 10
db.c6gd.large	2	—	4	1 x 118 NVMe SSD	最大 4,750	最大 10

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.c6gd.medium	1	—	2	1 x 59 NVMe SSD	最大 4,750	最大 10

バーストパフォーマンスインスタンスクラスのハードウェア仕様

次の表は、バーストパフォーマンスインスタンスクラスのコンピューティング、メモリ、ストレージ、帯域幅の仕様を示しています。

db.t4g – AWS Graviton2 プロセッサを搭載したバーストパフォーマンスインスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.t4g.2xlarge*	8	—	32	EBS 最適化のみ	最大 2,780	最大 5
db.t4g.xlarge*	4	—	16	EBS 最適化のみ	最大 2,780	最大 5
db.t4g.large*	2	—	8	EBS 最適化のみ	最大 2,780	最大 5
db.t4g.medium*	2	—	4	EBS 最適化のみ	最大 2,085	最大 5
db.t4g.small*	2	—	2	EBS 最適化のみ	最大 2,085	最大 5
db.t4g.micro*	2	—	1	EBS 最適化のみ	最大 2,085	最大 5

db.t3 – バーストパフォーマンスインスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.t3.2xlarge*	8	可変	32	EBS 最適化のみ	最大 2,780	最大 5
db.t3.xlarge*	4	可変	16	EBS 最適化のみ	最大 2,780	最大 5
db.t3.large*	2	可変	8	EBS 最適化のみ	最大 2,780	最大 5
db.t3.medium*	2	可変	4	EBS 最適化のみ	最大 2,085	最大 5
db.t3.small*	2	可変	2	EBS 最適化のみ	最大 2,085	最大 5
db.t3.micro*	2	可変	1	EBS 最適化のみ	最大 2,085	最大 5

db.t2 – バーストパフォーマンスインスタンスクラス

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.t2.2xlarge	8	可変	32	EBS のみ	—	中
db.t2.xlarge	4	可変	16	EBS のみ	—	中
db.t2.large	2	可変	8	EBS のみ	—	中
db.t2.medium	2	可変	4	EBS のみ	—	中
db.t2.small	1	可変	2	EBS のみ	—	低

インスタンスクラス	vCPU	ECU	メモリ (GiB)	インスタンスストレージ (GiB)	最大 EBS 帯域幅 (Mbps)	ネットワーク帯域幅 (Gbps)
db.t2.micro	1	可変	1	EBS のみ	—	低

* これらのインスタンスクラスは、24 時間に最低 1 回、30 分間にわたって、最大パフォーマンスをサポートできます。基礎となる EC2 インスタンスタイプのベースラインパフォーマンスの詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」を参照してください。

Amazon RDS DB インスタンスストレージ

Amazon RDS for Db2、Amazon RDS for MariaDB、Amazon RDS for PostgreSQL、Amazon RDS for Oracle、および Amazon RDS for Microsoft SQL Server の DB インスタンスは、データベースおよびログのストレージに Amazon Elastic Block Store (Amazon EBS) ボリュームを使用します。

場合によっては、データベースワークロードは、プロビジョニングした IOPS を 100% 到達できません。詳細については、「データベースのパフォーマンスに影響する要因」を参照してください。

インスタンスストレージの料金の詳細については、「Amazon RDS の料金」を参照してください。

トピック

- Amazon RDS ストレージタイプ
- プロビジョンド IOPS SSD ストレージ
- 汎用 SSD ストレージ
- ソリッドステートドライブ (SSD) ストレージタイプのパフォーマンス特性
- マグネティックストレージ (レガシー、非推奨)
- 追加のストレージボリューム
- 専用ログボリューム (DLV)
- データベースのパフォーマンスのモニタリング
- データベースのパフォーマンスに影響する要因

Important

Amazon RDS は、マグネティックストレージを廃止しました。今後、マグネティックストレージを使用して新しい DB インスタンスを作成できなくなります。また、Amazon RDS は既存のマグネティックストレージボリュームを gp3 に移行しました。2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。マグネティックボリュームのスナップショットを復元する場合は、gp3 や io2 Block Express などの別のストレージタイプを選択する必要があります。詳細については、「DB インスタンスへの復元」を参照してください。

Amazon RDS ストレージタイプ

Amazon RDS は 2 種類のストレージタイプを提供しています。プロビジョンド IOPS SSD (別名 io1 および io2 Block Express)、汎用 SSD (別名 gp2 および gp3) です。これらはパフォーマンス特性と料金が異なるため、データベースのワークロードに応じてストレージのパフォーマンスとコストを調整できます。最大 64 テビバイト (TiB) のストレージを備えた Db2、MySQL、MariaDB、PostgreSQL の RDS DB インスタンスを作成できます。追加のストレージボリュームを使用して、最大 256 TiB のストレージとを備えた Oracle および SQL Server インスタンスを作成できます。詳細については、「追加のストレージボリューム」を参照してください。RDS for Db2 は、gp2 ストレージタイプをサポートしていません。マグネティックストレージ (標準とも呼ばれます) は廃止され、新しい DB インスタンスでは提供されなくなります。詳細については、「マグネティックストレージ (レガシー、非推奨)」を参照してください。

次のリストでは、ストレージのタイプについて簡単に説明しています。

- プロビジョンド IOPS SSD - プロビジョンド IOPS ストレージは、低 I/O レイテンシーおよび一貫した I/O スループットが必要となる I/O 負荷の高いワークロード (特にデータベースワークロード) のニーズを満たすように設計されています。プロビジョンド IOPS ストレージは本稼働環境に最適です。

  ストレージサイズの範囲を含むプロビジョンド IOPS ストレージの詳細については、「プロビジョンド IOPS SSD ストレージ」を参照してください。
- 汎用 SSD - 汎用 SSD ボリュームは、中規模の DB インスタンスで実行しているさまざまなワークロードに対応できるコスト効率の高いストレージとして使用できます。汎用ストレージは、開発およびテスト環境に適しています。

  ストレージサイズの範囲を含む汎用 SSD ストレージの詳細については、「汎用 SSD ストレージ」を参照してください。
- マグネティック (廃止) – マグネティックストレージは廃止され、新しい DB インスタンスでは提供されなくなります。2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。新しいストレージが必要な場合は、汎用 SSD またはプロビジョンド IOPS SSD を使用してください。詳細については、「マグネティックストレージ (レガシー、非推奨)」を参照してください。

プロビジョンド IOPS SSD ストレージ

高速で一貫した I/O 性能を必要とするすべての本稼働アプリケーションに、プロビジョンド IOPS ストレージをお勧めします。プロビジョンド IOPS ストレージは、予測可能なパフォーマンスと一貫して低いレイテンシーを実現するストレージタイプです。プロビジョンド IOPS ストレージは、一貫したパフォーマンスが求められるオンライントランザクション処理 (OLTP) ワークロード向けに最適化されています。プロビジョンド IOPS は、このようなワークロードのパフォーマンス調整に役立ちます。

Amazon RDS は、2 種類のプロビジョンド IOPS SSD ストレージ (io2 および io1) を提供します。DB インスタンスの作成時に、IOPS レートとボリュームのサイズを指定します。Amazon RDS では、変更するまでその IOPS レートが DB インスタンスに提供されます。

トピック

- io2 Block Express ストレージ (推奨)
- io1 ストレージ (旧世代)
- プロビジョンド IOPS ストレージとマルチ AZ 配置またはリードレプリカの組み合わせ
- プロビジョンド IOPS ストレージのコスト
- Amazon RDS プロビジョンド IOPS ストレージから最高のパフォーマンスを得る

io2 Block Express ストレージ (推奨)

I/O 集中型でレイテンシーの影響を受けやすいワークロードでは、1 秒あたり最大 256,000 回の I/O オペレーション (IOPS) を実現するため、プロビジョンド IOPS SSD io2 Block Express ストレージを使用することをお勧めします。io2 Block Express ボリュームのスループットは、ボリュームごとにプロビジョニングされる IOPS の量と、実行される I/O 操作のサイズによって異なります。

AWS Nitro System ベースの RDS io2 ボリュームは、すべて io2 Block Express ボリュームで、平均レイテンシーはサブミリ秒です。AWS Nitro システムベースでない DB インスタンスは io2 ボリュームです。

次の表は、各データベースエンジンおよびストレージサイズの範囲ごとに、ストレージボリュームあたりのプロビジョンド IOPS の範囲と、最大スループットを示しています。Amazon RDS for Oracle および SQL Server では、プライマリストレージボリュームに加えて最大 3 つの追加のストレージボリュームをアタッチして、合計で最大 256 TiB のストレージサイズをプロビジョニングできます。追加のストレージボリュームを使用すると、DB インスタンスに対して、より高いプロビジョンド IOPS と最大スループットをプロビジョニングできます。ただし、インスタンスクラスの制限がスト

レージボリュームにプロビジョニングした値よりも小さい場合、DB インスタンスはプロビジョンド IOPS と最大スループットを十分に活用できない場合があります。詳細については、「データベースのパフォーマンスに影響する要因」を参照してください。

データベースエンジン	ストレージサイズの範囲	プロビジョンド IOPS の範囲	最大スループット
Db2、MariaDB、MySQL、および PostgreSQL	100 ~ 65,536 GiB	1,000 ~ 256,000 IOPS	16,000 MiB/秒
Oracle	100 ~ 199 GiB	1,000 ~ 199,000 IOPS	4,000 MiB/秒
Oracle	200 ~ 65,536 GiB	1,000 ~ 256,000 IOPS	16,000 MiB/秒
SQL Server	20 ~ 65,536 GiB	1,000 ~ 256,000 IOPS	4,000 MiB/秒

IOPS とストレージサイズ範囲には、次の制約があります。

- 割り当て済みストレージに対する IOPS の比率 (GiB 単位) は、0.5 から 1000 の範囲でなければなりません。AWS Nitro System ベースでない DB インスタンスの場合、この比率は 0.5 から 500 の範囲にする必要があります。
- 最大 IOPS は、256 GiB 以上でプロビジョニングできます (1,000 IOPS × 256 GiB = 256,000 IOPS)。AWS Nitro システムベースでない DB インスタンスの場合、最大 IOPS は 512 GiB (500 IOPS x 512 GiB = 256,000 IOPS) でプロビジョニングできます。
- スループットはプロビジョンド IOPS ごとに最大 0.256 MiB/秒に比例して拡張できます。16 KiB の I/O サイズでは 256,000 IOPS、256 KiB の I/O サイズでは 16,000 IOPS 以上で 4,000 MiB/秒の最大スループットを達成できます。AWS Nitro システムベースでない DB インスタンスの場合、16 KiB の I/O サイズで 128,000 IOPS で 2,000 MiB/秒の最大スループットを達成できます。
- ストレージの自動スケーリングを使用している場合は、IOPS と最大ストレージしきい値 (GiB 単位) の間でも同じ比率が適用されます。ストレージのオートスケーリングの詳細については、「Amazon RDS ストレージの自動スケーリングによる容量の自動管理」を参照してください。

Amazon RDS io2 Block Express ボリュームは、すべての商用 AWS リージョンと AWS GovCloud (US) Regions で利用可能です。これらのボリュームは、中国リージョンでは使用できません。

io1 ストレージ (旧世代)

I/O 集中型のワークロードでは、プロビジョンド IOPS SSD io1 ストレージを使用して 1 秒あたり最大 256,000 回の I/O オペレーション (IOPS) を実現できます。io1 ボリュームのスループットは、ボリュームごとにプロビジョニングされる IOPS の量と、実行される I/O 操作のサイズによって異なります。利用可能な場合、io2 Block Express ストレージの使用をお勧めします。

次の表は、各データベースエンジンのプロビジョンド IOPS および最大スループットの範囲とストレージサイズの範囲を示しています。

データベースエンジン	ストレージサイズの範囲	プロビジョンド IOPS の範囲	最大スループット
Db2、MariaDB、MySQL、および PostgreSQL	100 ~ 399 GiB	1,000 ~ 19,950 IOPS	500 MiB/秒
Db2、MariaDB、MySQL、および PostgreSQL	400 ~ 65,536 GiB	1,000 ~ 256,000 IOPS	4,000 MiB/秒
Oracle	100 ~ 199 GiB	1,000 ~ 9,950 IOPS	500 MiB/秒
Oracle	200 ~ 65,536 GiB	1,000 ~ 256,000 IOPS[1]	4,000 MiB/秒
SQL Server	20 ~ 16,384 GiB	1,000 ~ 64,000 IOPS[2]	1,000 MiB/秒

> **Note**
>
> [1] Oracle の場合、最大 256,000 IOPS は r5b インスタンスタイプでのみプロビジョンできます。
>
> [2] SQL Server の場合、最大 IOPS の 64,000 は、m5*、m6i、r5*、r6i、および z1d インスタンスタイプにある Nitro-based のインスタンスでのみ保証されます。その他のインスタンスタイプは、最大 32,000 IOPS までのパフォーマンスを保証します。

IOPS とストレージサイズ範囲には、次の制約があります。

- 割り当て済みストレージに対する IOPS の比率 (GiB 単位) は、RDS for SQL Server では 1 ~ 50 で、他の RDS DB エンジンでは 0.5 ~ 50 である必要があります。
- ストレージの自動スケーリングを使用している場合は、IOPS と最大ストレージしきい値 (GiB 単位) の間でも同じ比率が適用されます。

  ストレージのオートスケーリングの詳細については、「Amazon RDS ストレージの自動スケーリングによる容量の自動管理」を参照してください。

プロビジョンド IOPS ストレージとマルチ AZ 配置またはリードレプリカの組み合わせ

本稼働 OLTP ユースケースの場合、耐障害性を強化できるマルチ AZ 配置を高速で予測可能なパフォーマンスを可能にするプロビジョンド IOPS ストレージで使用することをお勧めします。

また、MySQL、MariaDB または PostgreSQL 用リードレプリカでもプロビジョンド IOPS ストレージを使用できます。リードレプリカのストレージタイプは、プライマリ DB インスタンスのストレージタイプとは異なります。例えば、リードレプリカには汎用 SSD を使用し、プライマリ DB インスタンスにはプロビジョンド IOPS SSD ストレージを使用することで、コストを削減できます。ただし、この場合のリードレプリカのパフォーマンスは、プライマリ DB インスタンスとリードレプリカの両方がプロビジョンド IOPS ストレージを使用するように設定した場合のものと異なる可能性があります。

プロビジョンド IOPS ストレージのコスト

プロビジョンド IOPS ストレージでは、月別に実際に使用したかどうかに関係なく、プロビジョンドされたリソースに応じて課金されます。

料金の詳細については、「Amazon RDS の料金」を参照してください。

Amazon RDS プロビジョンド IOPS ストレージから最高のパフォーマンスを得る

ワークロードが I/O 限定の場合、プロビジョンド IOPS ストレージを使用すると、システムが同時に処理できる I/O リクエストの数を増大することができます。同時処理数が増加すると、I/O リクエストがキューにとどまる時間が縮小するため、レイテンシーが減少します。レイテンシーが減少すると、データベースのコミット処理が高速になり、その結果、応答時間が短縮され、データベースのスループットが向上します。

プロビジョンド IOPS ストレージでは、IOPS を指定することにより I/O 容量を予約する方法が提供されます。ただし、他のシステム容量属性と同様に、負荷時の最大スループットは初期に使用され

るリソースによって制限されます。そのリソースは、ネットワーク帯域幅、CPU、メモリ、またはデータベースの内部リソースのいずれかである場合があります。

汎用 SSD ストレージ

汎用ストレージは、レイテンシーまたはパフォーマンスの影響をあまり受けない、ほとんどのデータベースワークロードで使用できる、コスト効果に優れたストレージです。

> **Note**
>
> 汎用ストレージを使用する DB インスタンスは、プロビジョンド IOPS ストレージを使用するインスタンスよりもレイテンシーが大幅に大きくなります。これらの操作の後に、レイテンシーが最小の DB インスタンスを必要とする場合は、プロビジョンド IOPS SSD ストレージ を使用することをお勧めします。

Amazon RDS ストレージは、gp3 ストレージ (推奨) と gp2 ストレージ (旧世代) の 2 種類の汎用ストレージを提供します。

gp3 ストレージ (推奨)

汎用 gp3 ストレージボリュームを使用することで、ストレージ容量に関係なくストレージパフォーマンスをカスタマイズできます。ストレージパフォーマンスは、1 秒あたりの I/O オペレーションの数 (IOPS) と、ストレージボリュームが読み取りと書き込みを実行できる速度 (ストレージスループット) を組み合わせたものです。gp3 ストレージボリュームでは、Amazon RDS は 3000 IOPS と 125 MiB/秒のベースラインストレージパフォーマンスを提供します。

RDS for SQL Server を除くすべての RDS DB エンジンでは、gp3 ボリュームのストレージサイズが一定のしきい値に達すると、ベースラインストレージパフォーマンスは増加します。これは、ストレージが 1 つのボリュームではなく 4 つの論理ボリュームを使用するボリュームストライピングによるものです。RDS for SQL Server はボリュームストライピングをサポートしていないため、しきい値はありません。ストライピングボリュームでは、Amazon RDS は 12,000 IOPS および 500 MiB/秒のベースラインストレージパフォーマンスを提供します。

ストレージボリュームごとのしきい値を含む、Amazon RDS DB エンジン上の gp3 ボリュームのストレージパフォーマンスを次の表に示します。RDS for Oracle および SQL Server では、プライマリストレージボリュームに加えて、最大 3 つの追加のストレージボリュームをアタッチできます。RDS for Oracle では、3 つの gp3 追加ストレージボリュームを使用して、合計で最大 256 TiB

のストレージサイズをプロビジョニングできます。各 gp3 ストレージボリュームのサイズは最大 64 TiB にできるため、RDS for SQL Server の 3 つの gp3 追加ストレージボリュームを使用して、合計で最大 256 TiB のストレージサイズをプロビジョニングできます。追加のストレージボリュームを使用すると、DB インスタンスに対して、より高いプロビジョンド IOPS と最大スループットをプロビジョニングできます。ただし、インスタンスクラスの制限がストレージボリュームにプロビジョニングした値よりも小さい場合、DB インスタンスはプロビジョンド IOPS と最大スループットを十分に活用できない場合があります。詳細については、「データベースのパフォーマンスに影響する要因」を参照してください。

DB エンジン	ストレージサイズ	ベースラインストレージパフォーマンス	プロビジョンド IOPS の範囲	プロビジョニングされたストレージのスループットの範囲
Db2、MariaDB、MySQL、および PostgreSQL	20 ~ 399 GiB	3,000 IOPS/125 MiB/秒	該当なし	該当なし
Db2、MariaDB、MySQL、および PostgreSQL	400 ~ 65,536 GiB	12,000 IOPS/500 MiB/秒	12,000 ~ 64,000 IOPS	500 ~ 4,000 MiB/秒
Oracle	20 ~ 199 GiB	3,000 IOPS/125 MiB/秒	該当なし	該当なし
Oracle	200 ~ 65,536 GiB	12,000 IOPS/500 MiB/秒	12,000 ~ 64,000 IOPS	500 ~ 4,000 MiB/秒
SQL Server	20 ~ 65,536 GiB	3,000 IOPS/125 MiB/秒	3,000–80,000 IOPS	125 ~ 2,000 MiB/秒

RDS for SQL Server を除くすべての DB エンジンで、ストレージサイズがしきい値以上の場合、追加の IOPS とストレージスループットをプロビジョニングできます。RDS for SQL Server では、使用可能な任意のストレージサイズに対して追加の IOPS とストレージスループットをプロビジョニングできます。すべての DB エンジンについて、追加でプロビジョニングされたストレージのパフォー

マンスに対してのみお支払いいただきます。詳細については、「Amazon RDS の料金」を参照してください。

追加されるプロビジョンド IOPS とストレージスループットはストレージサイズには依存しませんが、相互に関連しています。MariaDB と MySQL の IOPS を 32,000 より高くすると、ストレージのスループットの値は 500 MiBps から自動的に増加します。例えば、RDS for MySQL で IOPS を 40,000 に設定した場合、ストレージのスループットは 625 MiBps 以上でなければなりません。Db2、Oracle、PostgreSQL、SQL Server の DB インスタンスでは、自動的な増加は行われません。

マルチ AZ DB クラスターの場合、Amazon RDS はプロビジョンした IOPS に基づいてスループット値を自動的に設定します。ユーザーは、このスループット値は変更できません。

RDS 上の gp3 ボリュームのストレージパフォーマンス値には、次の制約事項があります。

- IOPS に対するストレージスループットの最大比率は、サポートされているすべての DB エンジンで 0.25 です。
- 割り当て済みストレージに対する IOPS の最小比率 (GiB 単位) は、RDS for SQL Server では 0.5 です。サポートされている他の DB エンジンに最小比率はありません。
- 割り当て済みストレージに対する IOPS の最大比率は、サポートされているすべての DB エンジンで 500 です。
- ストレージの自動スケーリングを使用している場合は、IOPS と最大ストレージしきい値 (GiB 単位) の間でも同じ比率が適用されます。

  ストレージのオートスケーリングの詳細については、「Amazon RDS ストレージの自動スケーリングによる容量の自動管理」を参照してください。

gp2 ストレージ (旧世代)

アプリケーションで高いストレージパフォーマンスを必要としない場合は、汎用 SSD gp2 ストレージを使用できます。gp2 ストレージのベースライン I/O パフォーマンスは、1 GiB あたり 3 IOPS で、最低 100 IOPS です。この関係は、ボリュームが大きいほどパフォーマンスが向上することを意味します。例えば、1 つの 100 GiB のボリュームのベースラインパフォーマンスは 300 IOPS です。1 つの 1,000 GiB ボリュームのベースラインパフォーマンスは 3,000 IOPS です。

サイズが 1,000 GiB 未満の個別の gp2 ボリュームでも、長期間にわたって 3,000 IOPS をバーストする性能があります。ボリューム I/O クレジットバランスによって、バーストパフォーマンスが決まります。ベースラインパフォーマンスと I/O クレジットバランスがパフォーマンスにどのように影響

を与えるかについての詳細は、AWS データベースブログの投稿「Understanding Burst vs. Baseline Performance with Amazon RDS and GP2」(Amazon RDS と GP2 を使用したバーストとベースラインパフォーマンスを理解する) を参照してください。

多くのワークロードは、バーストバランスを消費することがありません。しかしながら、ワークロードによっては、3,000 IOPS のバーストストレージクレジットバランスが枯渇する場合があるため、ワークロードのニーズを満たすようにストレージ容量を計画します。

4,000 GiB を超える gp2 ボリュームの場合、ベースラインパフォーマンスはバーストパフォーマンスよりも大きくなります。このようなボリュームの場合、ベースラインパフォーマンスが 3,000 IOPS バーストパフォーマンスよりも優れているため、バーストは関係ありません。ただし、特定のエンジンとサイズの DB インスタンスでは、ストレージが 4 つのボリュームにストライピングされ、1 つのボリュームの 4 倍のベースラインスループットと 4 倍のバースト IOPS が提供されます。

Amazon RDS DB エンジンでのさまざまなストレージサイズの gp2 ボリュームのストレージパフォーマンスを次の表に示します。

DB エンジン	RDS ストレージサイズ	ベースライン IOPS の範囲	ベースラインスループットの範囲	バースト IOPS
MariaDB、MySQL、PostgreSQL	5 ~ 399 GiB	100 ~ 1197 IOPS	128 ~ 250 MiB/秒	3,000
MariaDB、MySQL、PostgreSQL	400 ~ 1,335 GiB	1,200 ~ 4,005 IOPS	512 ~ 1,000 MiB/秒	12,000
MariaDB、MySQL、PostgreSQL	1,336 ~ 3,999 GiB	4,008 ~ 11,997 IOPS	1,000 MiB/秒	12,000
MariaDB、MySQL、PostgreSQL	4,000 ~ 65,536 GiB	12,000 ~ 64,000 IOPS	1,000 MiB/秒	該当なし[1]
Oracle	20 ~ 199 GiB	100 ~ 597 IOPS	128 ~ 250 MiB/秒	3,000

DB エンジン	RDS ストレージサイズ	ベースライン IOPS の範囲	ベースラインスループットの範囲	バースト IOPS
Oracle	200 ~ 1,335 GiB	600 ~ 4,005 IOPS	512 ~ 1,000 MiB/秒	12,000
Oracle	1,336 ~ 3,999 GiB	4,008 ~ 11,997 IOPS	1,000 MiB/秒	12,000
Oracle	4,000 ~ 65,536 GiB	12,000 ~ 64,000 IOPS	1,000 MiB/秒	該当なし[1]
SQL Server	20 ~ 333 GiB	100 ~ 999 IOPS	128 ~ 250 MiB/秒	3,000
SQL Server	334 ~ 999 GiB	1,002 ~ 2,997 IOPS	250 MiB/秒	3,000
SQL Server	1,000 ~ 16,384 GiB	3,000 ~ 16,000 IOPS	250 MiB/秒	該当なし[1]

[1] ボリュームのベースラインパフォーマンスが最大バーストパフォーマンスを超えた場合。

ソリッドステートドライブ (SSD) ストレージタイプのパフォーマンス特性

次の表に Amazon RDS が使用する SSD ストレージボリュームのユースケースとボリュームあたりのパフォーマンス特性を示します。

特性	プロビジョンド IOPS (io2 Block Express)	プロビジョンド IOPS (io1)	汎用 (gp3)	汎用 (gp2)
説明	RDS ストレージポートフォリオ内で最高のパフォーマンス (IOPS、スループット、レイテンシー)	一貫したストレージパフォーマンス (IOPS、スループット、レイテンシー)	ストレージ、IOPS、スループットを個別にプロビジョニングできる柔軟性	バースト可能な IOPS を提供します。 さまざまなトランザクション

特性	プロビジョンド IOPS (io2 Block Express)	プロビジョンド IOPS (io1)	汎用 (gp3)	汎用 (gp2)
	レイテンシーの影響を受けやすいトランザクションワークロード向けに設計されています	レイテンシーの影響を受けやすいトランザクションワークロード向けに設計されています	さまざまなトランザクションワークロードに対して、料金パフォーマンスのバランスをとります。	ワークロードに対して、料金パフォーマンスのバランスをとります。
ユースケース	サブミリ秒のレイテンシーと最大 256,000 IOPS の持続的な IOPS パフォーマンスを必要とするビジネスクリティカルなトランザクションワークロード。	最大 256,000 IOPS の持続的な IOPS パフォーマンスを必要とするトランザクションワークロード。	開発/テスト環境の中規模リレーショナルデータベースで実行される幅広いワークロード	開発/テスト環境の中規模リレーショナルデータベースで実行される幅広いワークロード
レイテンシー	サブミリ秒 (99.9% の期間一貫して提供)	1 桁ミリ秒 (99.9% の期間一貫して提供)	1 桁ミリ秒 (99% の期間一貫して提供)	1 桁ミリ秒 (99% の期間一貫して提供)
ボリュームサイズ	100 ~ 65,536 GiB	100 ~ 65,536 GiB (RDS for SQL Server では 20–16,384 GiB)	20 ~ 65,536 GiB (RDS for SQL Server では 16,384 GiB)	20 ~ 65,536 GiB (RDS for SQL Server では 16,384 GiB)

特性	プロビジョンド IOPS (io2 Block Express)	プロビジョンド IOPS (io1)	汎用 (gp3)	汎用 (gp2)
最大 IOPS	256,000	256,000 (RDS for SQL Server では 64,000)	64,000 (RDS for SQL Server では 16,000)	64,000 (RDS for SQL Server では 16,000) **Note** gp2 ストレージに IOPS を直接プロビジョニングすることはできません。IOPS は割り当てられたストレージサイズによって異なります。

特性	プロビジョンド IOPS (io2 Block Express)	プロビジョンド IOPS (io1)	汎用 (gp3)	汎用 (gp2)
最大スループット	プロビジョンド IOPS に基づいて最大 4,000 MB/秒まで拡張 スループットはプロビジョンド IOPS ごとに最大 0.256 MiB/秒に比例して拡張できます。16 KiB の I/O サイズでは 256,000 IOPS、256 KiB の I/O サイズでは 16,000 IOPS 以上で 4,000 MiB/秒の最大スループットを達成できます。 AWS Nitro システムベースでないインスタンスの場合、16 KiB の I/O サイズで 128,000 IOPS で最大 2,000 MiB/秒のスループットを達成できます。	プロビジョンド IOPS に基づいて最大 4,000 MB/秒まで拡張	最大 4,000 MB/秒の追加のスループットをプロビジョニング (RDS for SQL Server では 1,000 MB/秒)	1,000 MB/秒 (RDS for SQL Server では 250 MB/秒)
AWS CLI および RDS API 名	io2	io1	gp3	gp2

SSD ボリューム間の自動ストライピング

汎用 SSD またはプロビジョンド IOPS SSD を選択すると、選択したエンジンと要求されるストレージの量に応じて、Amazon RDS は次の表に示すように複数のボリュームに自動的にストライピングしてパフォーマンスを向上させます。

データベースエンジン	Amazon RDS ストレージサイズ	プロビジョニングされたボリューム数
Db2	400 GiB 未満	1
Db2	400 ~ 65,536 GiB	4
MariaDB、MySQL、PostgreSQL	400 GiB 未満	1
MariaDB、MySQL、PostgreSQL	400 ~ 65,536 GiB	4
Oracle	200 GiB 未満	1
Oracle	200 ~ 65,536 GiB	4
SQL Server	いずれか	1

SSD ボリュームを変更した場合のパフォーマンスへの影響

汎用 SSD ボリュームまたはプロビジョンド IOPS SSD ボリュームを変更すると、次の順に状態が変更されます。ボリュームが `optimizing` 状態である場合、ボリュームのパフォーマンスはソースとターゲットの設定仕様の中間にあります。過渡的ボリュームのパフォーマンスは、2 つの仕様のうち最低のものを下回ることはありません。

インスタンスのストレージを 1 つのボリュームから 4 つのボリュームに変更したり、磁気ストレージを使用してインスタンスを変更したりする場合、Amazon RDS は Elastic Volumes 機能を使用しません。代わりに、Amazon RDS は新しいボリュームをプロビジョニングし、古いボリュームから新しいボリュームにデータを透過的に移動します。このオペレーションは、古いボリュームと新しいボリュームの両方で大量の IOPS とスループットを消費します。ボリュームのサイズと変更中に発生するデータベースワークロードの量によっては、RDS インスタンスが `Modifying` 状態のままであっ

ても、この操作は大量の IOPS を消費し、I/O レイテンシーを大幅に増加させ、完了するまでに数時間かかる場合があります。

EBS 最適化インスタンスのベースライン IOPS レートと最大 IOPS レート

EBS 最適化インスタンスには、ベースライン IOPS レートと最大 IOPS レートがあります。最大 IOPS レートは、DB インスタンスレベルで適用されます。IOPS レートが最大値を超えるように組み合わされた EBS ボリュームのセットは、インスタンスレベルのしきい値を超えることはできません。例えば、特定の DB インスタンスクラスの最大 IOPS が 40,000 であり、64,000 IOPS EBS ボリュームを 4 つアタッチした場合、最大 IOPS は 256,000 ではなく 40,000 になります。各 EC2 インスタンスタイプに固有の IOPS の最大値については、「Amazon EC2 Linux インスタンス用ユーザーガイドの「インスタンスタイプ」を参照してください。

マグネティックストレージ (レガシー、非推奨)

> **Warning**
>
> Amazon RDS は、マグネティックストレージを廃止しました。今後、マグネティックストレージを使用して新しい DB インスタンスを作成できなくなります。また、Amazon RDS は既存のマグネティックストレージボリュームを gp3 に移行しました。2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。マグネティックボリュームのスナップショットを復元する場合は、gp3 や io2 Block Express などの別のストレージタイプを選択する必要があります。

マグネティックストレージは廃止され、新しい DB インスタンスでは提供されなくなります。新しいストレージが必要な場合は、汎用 SSD またはプロビジョンド IOPS SSD を使用してください。マグネティックストレージには、次の制限が適用されます。

- SQL Server データベースエンジン使用時には、ストレージをスケーリングできません。
- SQL Server データベースエンジンの使用時には、ストレージを別のストレージタイプに変換できません。
- ストレージのオートスケーリングをサポートしていません。
- Amazon Redshift とのゼロ ETL 統合はサポートされていません。
- Elastic ボリュームをサポートしていません。
- 最大サイズが 3 TiB に制限されます。

- 最大で 1000 IOPS に制限されます。

追加のストレージボリューム

RDS for Oracle および RDS for SQL Server では、DB インスタンスに最大 3 つの追加のストレージボリュームをアタッチできます。ワークロードの要件に応じて、ボリュームごとに gp3 ストレージと io2 ストレージのいずれかを選択します。

追加のストレージボリュームには、次の利点があります。

- 柔軟なストレージ設定とパフォーマンスの最適化 – さまざまなストレージタイプ (gp3 と io2) を混在させて、データアクセスパターンに基づいてコストとパフォーマンスの両方を最適化します。高性能 io2 ストレージのアクセス頻度の高いデータを、費用対効果の高い gp3 ストレージのアーカイブデータから分離します。
- 拡張容量 – プライマリストレージボリュームと追加のストレージボリュームを組み合わせて、DB インスタンスごとに合計ストレージを最大 256 TiB までスケールできます。
- 必要に応じてストレージ容量を拡張および削減する – データ移行中など、追加のストレージが必要な場合はボリュームを作成し、後でボリュームを削除します。このようにして、DB インスタンスストレージの合計を拡張および削減できます。
- オンラインデータ移動 – Oracle Database の組み込み機能を使用して、ダウンタイムなしでボリューム間でデータを移動します。

> **Note**
>
> 追加のストレージボリュームを削除することはできますが、プライマリボリュームを削除することはできません。

追加のボリュームでは、次の表に示すボリューム名を使用する必要があります。

RDS for Oracle ボリューム名	RDS for SQL Server ボリューム名
`rdsdbdata2`	`H:`
`rdsdbdata3`	`I:`
`rdsdbdata4`	`J:`

ボリュームあたりのストレージオペレーションのモニタリング

インスタンスに追加のストレージボリュームがある場合、初期化 (スナップショット復元、ポイントインタイムリカバリ、リードレプリカの作成、シングル AZ からマルチ AZ への変換後) や最適化 (ストレージのスケーリング後) などのストレージオペレーションがボリュームごとに追跡されます。このため、プライマリボリュームと追加の各ストレージボリュームは、同時に異なる状態になる可能性があります。例えば、ストレージのスケールアップ後に追加のストレージボリュームを最適化している間に、プライマリボリュームがまだ初期化中であるなどです。

ボリュームごとの状態を観察するには、DescribeDBInstances オペレーションを使用します。レスポンスには、最上位レベルの `StorageOperationStatus` と `StorageOperationPercentProgress` (プライマリボリュームの状態を報告する)、および `AdditionalStorageVolumes` 配列内の各エントリ (その特定のボリュームの状態を報告する) が含まれます。オペレーション中、`StorageOperationStatus` は `Initializing` または `Optimizing` に設定されます。オペレーションが完了すると、両方のフィールドは存在しなくなります。

DB インスタンスのステータスとストレージオペレーションフィールドの詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

追加のストレージボリュームの使用の詳細については、以下のセクションを参照してください。

- Amazon RDS DB インスタンスのストレージを使用する
- RDS for Oracle でストレージを使用する
- RDS for SQL Server のストレージの使用

専用ログボリューム (DLV)

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、プロビジョンド IOPS (PIOPS) ストレージを使用する DB インスタンスの専用ログボリューム (DLV) を使用できます。DLV は、PostgreSQL データベーストランザクションログと MySQL/MariaDB REDO ログおよびバイナリログを、データベーステーブルを含んでいるボリュームとは別のストレージボリュームに移動します。DLV を使用すると、トランザクション書き込みロギングの効率と一貫性が向上します。DLV は、割り当てられたストレージの容量が大きいデータベース、1 秒あたりの I/O (IOPS) 要件が高い、または遅延の影響を受けやすいワークロードがあるデータベースに最適です。

DLV は PIOPS ストレージ (io1 および io2 Block Express) でサポートされており、1,024 GiB の固定サイズと 3,000 のプロビジョンド IOPS で作成されます。

> **Note**
>
> DLV は汎用ストレージ (gp2 および gp3) ではサポートされていません。

Amazon RDS は、次のバージョンについて、すべての AWS リージョン の DLV でサポートします。

- MariaDB 10.6.7 以上の 10 バージョン
- MySQL 8.0.28 以上の 8.0 バージョン、MySQL 8.4.3 以上の 8.4 バージョン
- PostgreSQL 13.10 以降のバージョン 13、14.7 以降のバージョン 14、15.2 以降のバージョン 15、および 16.1 以降のバージョン 16

RDS は、マルチ AZ 配置で DLV をサポートします。マルチ AZ インスタンスを変更または作成すると、プライマリとセカンダリの両方に DLV が作成されます。

RDS はリードレプリカによる DLV をサポートします。プライマリ DB インスタンスで DLV が有効になっている場合、DLV を有効にした後に作成されたすべてのリードレプリカにも DLV が設定されます。DLV への切り替え前に作成されたリードレプリカは、明示的に変更されない限り、有効になりません。DLV を有効にする前にプライマリインスタンスにアタッチされていたすべてのリードレプリカも、DLV を使用するように手動で変更することが推奨されます。

DB インスタンスの DLV 設定を変更したら、DB インスタンスを再起動する必要があります。

DLV の有効化の詳細については、「専用ログボリューム (DLV) を使用する」を参照してください。

データベースのパフォーマンスのモニタリング

Amazon RDS では、DB インスタンスの動作を特定するためにいくつかのメトリクスを利用できます。Amazon RDS マネジメントコンソールのインスタンスの概要ページでメトリクスを表示できます。また、Amazon CloudWatch を使用して、これらのメトリクスをモニタリングすることもできます。詳細については、「Amazon RDS コンソールでのメトリクスの表示」を参照してください。拡張モニタリングは、より詳細な I/O メトリクスを提供します。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

以下のメトリクスは、DB インスタンスのパフォーマンスをモニタリングするために便利です。

- `DiskQueueDepth` - サービスされるのを待つキュー内の I/O リクエスト数。アプリケーションによって送信されても、デバイスが他の I/O リクエストの処理でビジー状態のため、デバイスに送信されていない I/O リクエストです。キューでの待ち時間は、レイテンシーとサービス時間 (メ

トリクスとしては使用できない) のコンポーネントです。このメトリックは、特定の時間間隔の平均キュー深度として報告されます。Amazon RDS は、キューの深さを 1 分間隔で報告します。キューの深度の典型的な値は、0 から数百までの範囲です。

- `EBSByteBalance%` - RDS データベースのバーストバケットに残っているスループットクレジットの割合。このメトリクスは基本モニタリング専用です。メトリクス値は、データベースファイルを含むボリュームのみではなく、ルートボリュームを含むすべてのボリュームのスループットに基づいています。

  このメトリクスがゼロに近づくと、DB インスタンスのコンピューティングキャパシティーが不足していることを意味します。このような状況が定期的に発生する場合は、db.r6g.large から db.r6g.xlarge など、より大きなインスタンスクラスサイズにアップグレードすることを検討してください。詳細については、「DB インスタンスクラス」を参照してください。

- `ReadIOPS` および `WriteIOPS` - 1 秒ごとの I/O オペレーションの数。このメトリクスは、指定された時間間隔の平均 IOPS として報告されます。Amazon RDS は、IOPS の読み取りと書き込みを 1 分間隔で個別に報告します。`TotalIOPS` は、IOPS の読み取りおよび書き込み IOPS の合計です。IOPS の典型的な値は、1 秒あたり 0 から数万の範囲内です。

  `TotalIOPS` 値が DB インスタンスに設定したプロビジョンド IOPS 値に定期的に近づく場合は、プロビジョンド IOPS (io1、io2 Block Express、gp3 ストレージタイプ) を増やすことを検討してください。

  測定された IOPS 値は、個別の I/O オペレーションのサイズには依存しません。これは、I/O パフォーマンスの測定時に、I/O オペレーションの数のみではなく、インスタンスのスループットを見る必要があることを意味しています。

- `ReadLatency` および `WriteLatency` - I/O リクエスト送信から完了までの経過時間。このメトリクスは、指定された時間間隔の平均レイテンシーとして報告されます。Amazon RDS は 1 分間レイテンシーで読み取りおよび書き込み IOPS を個別に報告します。レイテンシーの一般的な値はミリ秒 (ms) です。

- `ReadThroughput` および `WriteThroughput` - ディスクへまたはディスクから転送される 1 秒ごとのバイト数。このメトリックは、特定の時間間隔の平均スループットとして報告されます。Amazon RDS は、1 分間隔の読み取りおよび書き込みスループットを個別に 1 秒あたりのバイト (B/秒) 単位で報告します。スループットの典型的な値は、0 から I/O チャネルの最大帯域幅までの範囲内です。

  スループット値が DB インスタンスの最大スループットに定期的に近づいている場合は、gp3 ストレージタイプを使用している場合は、より多くのストレージスループットのプロビジョニングを検討してください。

データベースのパフォーマンスに影響する要因

システムの活用状況、データベースワークロード、DB インスタンスは、データベースのパフォーマンスに影響する可能性があります。

トピック

- システムの活用状況
- データベースのワークロード
- DB インスタンスクラス

システムの活用状況

次のシステム関連アクティビティは I/O 容量を消費するため、進行中に DB インスタンスのパフォーマンスを低下させる可能性があります。

- マルチ AZ スタンバイの作成
- リードレプリカの作成
- ストレージタイプを変更する

データベースのワークロード

データベースまたはアプリケーションの設計によっては、平行処理問題、ロック、その他データベースの競合が発生します。このような場合、プロビジョニングされた帯域幅のすべてを直接使用できない場合があります。また、以下のワークロードに関連する状況が発生する場合があります。

- 基本的なインスタンスタイプのスループット制限に到達すること。
- アプリケーションが十分な I/O オペレーションを進行していないため、キューの深度が 1 を大幅に下回ること。
- 一部の I/O 容量が未使用にもかかわらず、データベースでクエリの競合が発生すること。

場合によっては、制限に達している、またはほぼ制限に達しているシステムリソースがないにもかかわらず、スレッドを追加してもデータベースのトランザクションレートが増加しないことがあります。このような場合、ボトルネックはデータベース内の競合である可能性が最も高いです。最も一般的なのは行ロックとインデックスページロックの競合ですが、それ以外にも多くの可能性があります。このような状況では、データベースパフォーマンスチューニングの エキスパートに助言を求める必要があります。

DB インスタンスクラス

Amazon RDS DB インスタンスが最大限のパフォーマンスを発揮するように、現行世代のインスタンスタイプを十分な帯域幅と IOPS で選択して、ストレージタイプをサポートします。例えば、Amazon EBS 最適化インスタンス,および 10 ギガビットのネットワーク接続のインスタンスを選択できます。

> **Important**
>
> 使用しているインスタンスクラスによっては、RDS がプロビジョニングできる最大値よりも帯域幅、スループット、IOPS パフォーマンスが低下する場合があります。DB インスタンスクラスの帯域幅、スループット、IOPS パフォーマンスに関する詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンス」を参照してください。DB インスタンス内のストレージボリュームのプロビジョンド IOPS とスループット値を設定する前に、インスタンスクラスの最大帯域幅、スループット、IOPS を決定することをお勧めします。

最適なパフォーマンスを得るには、最新世代のインスタンスを使用してください。旧世代の DB インスタンスを使用すると、ストレージの上限が低くなります。

一部の古い 32 ビットファイルシステムでは、ストレージ容量が少ない場合があります。DB インスタンスのストレージ容量を確認するには、describe-valid-db-instance-modificationsAWS CLIコマンドを使用できます。

次のリストは、大部分の DB インスタンスクラスが各データベースエンジンに対して拡張できる最大ストレージを示しています。

- Db2 – 64 TiB
- MariaDB – 64 TiB
- Microsoft SQL Server – 64 TiB
- MySQL – 64 TiB
- Oracle – 64 TiB
- PostgreSQL – 64 TiB

次の表にいくつかの最大ストレージの例外を示します (TiB 単位)。io2 Block Express ストレージとは別のすべての RDS for Microsoft SQL Server DB インスタンスの最大ストレージは 16 TiB であるため、SQL Server のエントリはありません。

インスタンスクラス	Db2	MariaDB	MySQL	Oracle	PostgreSQL
db.m3 - スタンダードインスタンスクラス					
db.t4g – バーストパフォーマンスインスタンスクラス					
db.t4g.medium	該当なし	16	16	該当なし	32
db.t4g.small	該当なし	16	16	該当なし	16
db.t4g.micro	該当なし	6	6	該当なし	6
db.t3 – バーストパフォーマンスインスタンスクラス					
db.t3.medium	32	16	16	32	32
db.t3.small	32	16	16	32	16
db.t3.micro	該当なし	6	6	32	6
db.t2 – バーストパフォーマンスインスタンスクラス					

サポートされているすべてのインスタンスクラスの詳細については、旧世代の DB インスタンスを参照してください。

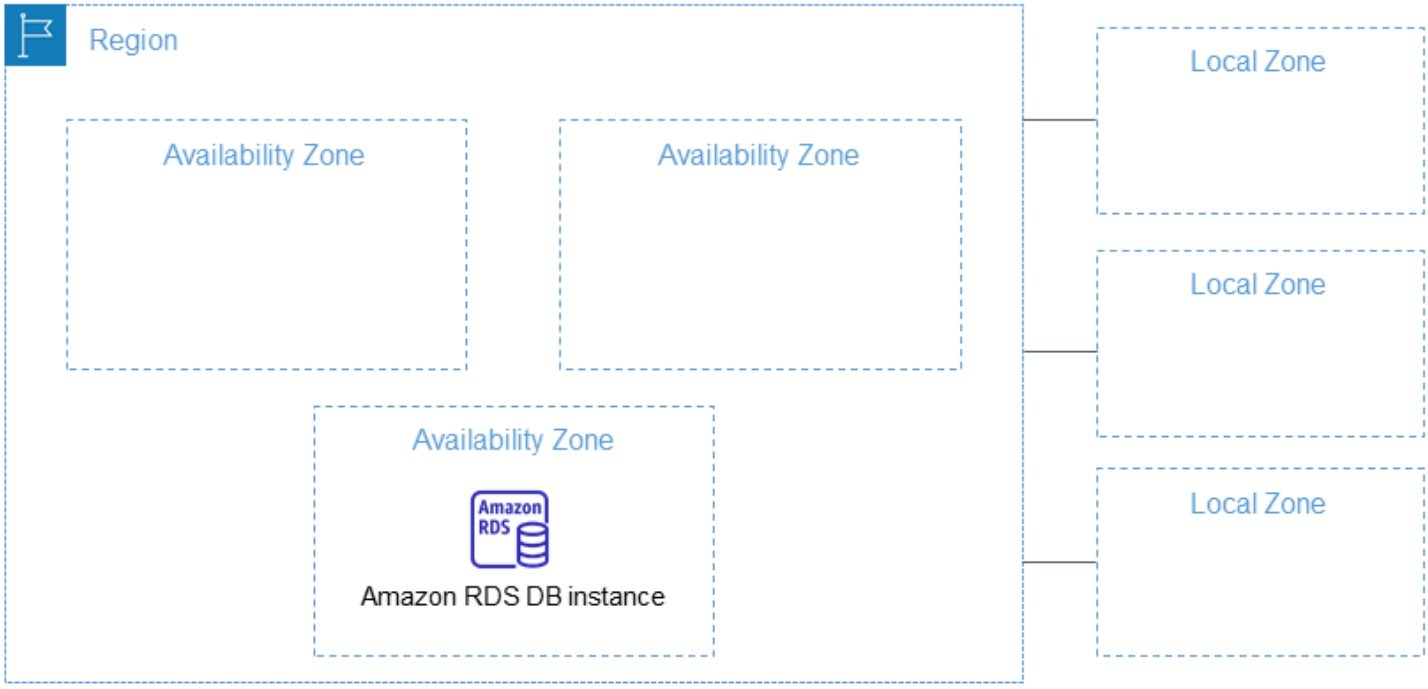
リージョン、アベイラビリティーゾーン、および Local Zones

Amazon クラウドコンピューティングリソースは、世界各地の多くの場所でホストされています。これらの場所は、AWS リージョン、アベイラビリティーゾーン、および Local Zonesで構成されています。AWS リージョンはそれぞれ、地理的に離れた領域です。AWS リージョンごとにアベイラビリティーゾーンと呼ばれる複数の独立した場所があります。

> **Note**
>
> AWS リージョンでのアベイラビリティーゾーンの検索については、Amazon EC2 ドキュメントの「Describe Your Availability Zones」(アベイラビリティーゾーンの詳細表示) を参照してください。

Local Zones を使用すると、コンピューティングやストレージなどのリソースをユーザーに近い複数の場所に配置できます。Amazon RDS を使用すると、DB インスタンスなどのリソースやデータを複数の場所に配置できます。リソースは、お客様が指定しない限り、複数の AWS リージョン間でレプリケートされることはありません。

Amazon は、アベイラビリティーの高い最新のデータセンターを運用しています。ただし、非常にまれですが、同じ場所にある DB インスタンスすべての可用性に影響する障害が発生することもあります。すべての DB インスタンスを 1 か所でホストしている場合、そのような障害が起きた際に DB インスタンスがすべて利用できなくなります。



各 AWS リージョンは完全に独立していることを覚えておくことが重要です。スタートした Amazon RDS アクティビティ (例えば、データベースインスタンスの作成や使用可能なデータベースインスタンスの一覧表示など) は、現在のデフォルト AWSリージョンでのみ実行されます。デフォルトの AWS リージョンは、コンソールで変更するか、環境変数 `AWS_DEFAULT_REGION` を設定することにより変更できます。または、AWS Command Line Interface (AWS CLI) で `--region` パラメータを使用すると、リージョンを上書きできます。詳細については、「AWS Command Line Interface の設定」を参照し、特に環境可変とコマンドラインオプションのセクションに注目してください。

Amazon RDS では、AWS と呼ばれる特別な AWS GovCloud (US) リージョンをサポートしています。これらのリージョンは、米国政府機関および関係者が、より機密性の高いワークロードをクラウドに移行するができるように設計されたものです。AWS GovCloud (US) リージョンは、米国政府の特定の規制とコンプライアンスの要件に対応しています。詳細については、「AWS GovCloud (US) とは」を参照してください。

特定の AWS リージョンの Amazon RDS DB インスタンスを作成または操作するには、対応するリージョンのサービスエンドポイントを使用します。

AWS リージョン

各 AWS リージョンは、他の AWS リージョンと完全に分離されるように設計されています。この設計により、最大限の耐障害性と安定性が達成されます。

リソースを表示すると、指定した AWS リージョンに結び付けられているリソースのみが表示されます。これは、AWS リージョンが相互に分離されており、AWS リージョン間ではリソースが自動的にレプリケートされないためです。

利用可能なリージョン

次の表は、現時点で Amazon RDS を利用できる AWS リージョンとリージョン別のエンドポイントの一覧です。

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	
米国東部 (バージニア北部)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
		rds-fips.us-east-1.api.aws	HTTPS	
		rds-fips.us-east-1.amazonaws.com	HTTPS	
		rds.us-east-1.api.aws	HTTPS	
米国西部 (北カリフォルニア)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
		rds.us-west-1.api.aws	HTTPS	
		rds-fips.us-west-1.amazonaws.com	HTTPS	
		rds-fips.us-west-1.api.aws	HTTPS	
米国西部 (オレゴン)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
		rds-fips.us-west-2.amazonaws.com	HTTPS	
		rds.us-west-2.api.aws	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル
		rds-fips.us-west-2.api.aws	HTTPS
アフリカ (ケープタウン)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
アジアパシフィック (香港)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS
アジアパシフィック (ジャカルタ)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS
		rds.ap-southeast-3.api.aws	HTTPS
アジアパシフィック (マレーシア)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
アジアパシフィック (メルボルン)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS
		rds.ap-southeast-4.api.aws	HTTPS
アジアパシフィック (ムンバイ)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
		rds.ap-south-1.api.aws	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (ニュージーランド)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS	
アジアパシフィック (ソウル)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com rds.ap-southeast-2.api.aws	HTTPS HTTPS	
アジアパシフィック (台北)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
アジアパシフィック (タイ)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (東京)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
カナダ (中部)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
カナダ西部 (カルガリー)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS
欧州 (アイルランド)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
		rds.eu-west-1.api.aws	HTTPS
欧州 (ロンドン)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
		rds.eu-west-2.api.aws	HTTPS
ヨーロッパ (ミラノ)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
		rds.eu-south-1.api.aws	HTTPS
欧州 (パリ)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
		rds.eu-west-3.api.aws	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (スペイン)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
		rds.eu-south-2.api.aws	HTTPS	
欧州 (ストックホルム)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
		rds.eu-north-1.api.aws	HTTPS	
欧州 (チューリッヒ)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	
		rds.eu-central-2.api.aws	HTTPS	
イスラエル (テルアビブ)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS	
		rds.il-central-1.api.aws	HTTPS	
メキシコ (中部)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
中東 (バーレーン)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS	
		rds.me-south-1.api.aws	HTTPS	
中東 (アラブ首長国連邦)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	
		rds.me-central-1.api.aws	HTTPS	
南米 (サンパウロ)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
		rds.sa-east-1.api.aws	HTTPS	
AWS GovCloud (米国東部)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
		rds.us-gov-east-1.api.aws	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
AWS GovCloud (米国西部)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS	

エンドポイントを明示的に指定しない場合、米国西部 (オレゴン) エンドポイントがデフォルトになります。

AWS CLI または API オペレーションを使用して DB インスタンスを操作する場合は、そのリージョンエンドポイントを指定する必要があります。

アベイラビリティーゾーン

DB インスタンスを作成するときに、アベイラビリティーゾーンを自分で選択するか、Amazon RDS にランダムに選択させることができます。アベイラビリティーゾーンは、AWS リージョンコードとそれに続く文字識別子によって表されます (us-east-1a など)。

次のように describe-availability-zones Amazon EC2 コマンドを使用して、アカウントで有効な、指定されたリージョン内のアベイラビリティーゾーンを記述します。

```
aws ec2 describe-availability-zones --region region-name
```

例えば、アカウントで有効になっている米国東部 (バージニア北部) リージョン (us-east-1) 内のアベイラビリティーゾーンを記述するには、次のコマンドを実行します。

```
aws ec2 describe-availability-zones --region us-east-1
```

マルチ AZ DB デプロイで、プライマリ DB インスタンスとセカンダリ DB インスタンスのアベイラビリティーゾーンを選択することはできません。Amazon RDS によってランダムに選択されます。マルチ AZ 配置については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

> **Note**
>
> RDS でアベイラビリティーゾーンをランダムに選択しても、1 つのアカウントまたは DB サブネットグループ内のアベイラビリティーゾーン間で DB インスタンスが均等に分散されるとは限りません。シングル AZ インスタンスを作成または変更するときに特定の AZ をリクエストできます。マルチ AZ インスタンスにはより固有の DB サブネットグループを使用できます。詳細については、「Amazon RDS DB インスタンスの作成」および「Amazon RDS DB インスタンスを変更する」を参照してください。

ローカルゾーン

ローカルゾーンは、ユーザーに地理的に近い、AWS リージョンの拡張です。親の AWS のリージョンからローカルゾーンに VPC を拡張できます。そのためには、新しいサブネットを作成し、これを AWS ローカルゾーンに割り当てます。ローカルゾーンにサブネットを作成すると、VPC はそのローカルゾーンに拡張されます。ローカルゾーンのサブネットは、VPC 内の他のサブネットと同じように動作します。

DB インスタンスを作成するときに、ローカルゾーンのサブネットを選択できます。Local Zones は、インターネットへの独自の接続を持ち、Direct Connect をサポートします。したがって、ローカルゾーンで作成したリソースは、非常に低いレイテンシーの通信をローカルユーザーに提供できます。詳細については、「AWS Local Zones」を参照してください。

ローカルゾーンを表すには、AWS リージョンコードに続けて場所を示す識別子を使用します (例: `us-west-2-lax-1a`)。

> **Note**
>
> ローカルゾーンをマルチ AZ 配置に含めることはできません。

ローカルゾーンを使用するには

1. Amazon EC2 コンソールでローカルゾーンを有効にします。

   詳細については、Amazon EC2 ユーザーガイドの「Local Zones の有効化」を参照してください。
2. ローカルゾーン内にサブネットを作成します。

詳細については、Amazon VPC ユーザーガイドの「VPC でサブネットを作成する」を参照してください。

3. ローカルゾーン内に DB サブネットグループを作成します。

   DB サブネットグループを作成するときに、ローカルゾーンのアベイラビリティーゾーングループを選択します。

   詳細については、「VPC に DB インスタンスを作成する」を参照してください。

4. DB サブネットグループを使用する DB インスタンスをローカルゾーン内に作成します。

   詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

> **Important**
>
> 現在、Amazon RDS が利用可能な AWS ローカルゾーンは、米国西部 (オレゴン) リージョン内のロサンゼルスのみです。

Amazon RDS リソースのリージョンオプトアウトとオプトインの動作

AWS では、アカウントの特定の AWS リージョンを有効または無効にできます。リージョンを無効化 (オプトアウト) すると、そのリージョンのリソースにアクセスできなくなります。リージョンを有効化 (オプトイン) すると、リージョンを再度使用できます。このトピックでは、リージョンのオプトアウトとオプトインが Amazon RDS リソースに与える影響について説明します。

AWS リージョンに関する一般的な情報、およびリージョンを有効化または無効化する方法については、「AWS アカウント管理リファレンスガイド」の「AWS リージョンの管理」を参照してください。Amazon RDS で使用できるリージョンについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

リージョンからオプトアウトするとどうなるか

AWS リージョンからオプトアウトすると、そのリージョンの Amazon RDS リソースに次の変更が適用されます。

- そのリージョンの DB インスタンスと DB クラスターのスナップショットが作成され、その後、DB インスタンスと DB クラスターが削除されます。
- オプトアウトしたリージョンに新しい DB インスタンスまたは DB クラスターを作成することはできません。

リージョンがオプトアウトされている間も、リージョン内のスナップショットに対して課金されます。

リージョンからのオプトアウトは、元に戻すことができるアクションです。リソースは、スナップショットの作成後にのみ削除されます。リージョンに再オプトインすると、スナップショットを使用してリソースを復元できます。

リージョンに再オプトインするとどうなるか

以前にオプトアウトした AWS リージョンに再オプトインすると、次の変更が適用されます。

- スナップショットは再びアクセス可能になり、それらを使用してリソースを復元できます。

AWS リージョン と DB エンジンにより Amazon RDS でサポートされている機能

Amazon RDS の機能とオプションのサポートは、AWS リージョン と各 DB エンジンの固有のバージョンによって異なります。特定の AWS リージョン での RDS DB エンジンのバージョンサポートと可用性を特定するには、次のセクションを使用できます。

Amazon RDS 機能は、エンジンネイティブの機能やオプションとは異なります。エンジンネイティブの機能とオプションの詳細については、「エンジンネイティブ機能」を参照してください。

サポートされているリージョンと DB エンジン

- テーブルの規則
- 機能クイックリファレンス
- Amazon RDS のブルー/グリーンデプロイでサポートされているリージョンと DB エンジン
- Amazon RDS でのクロスリージョン自動バックアップでサポートされているリージョンと DB エンジン
- Amazon RDS でのクロスリージョンリードレプリカでサポートされているリージョンと DB エンジン
- Amazon RDS のデータベースアクティビティストリームでサポートされているリージョンと DB エンジン
- Amazon RDS のデュアルスタックモードでサポートされているリージョンと DB エンジン
- Amazon RDS の S3 へのスナップショットデータエクスポートでサポートされているリージョンと DB エンジン
- Amazon RDS での IAM データベース認証でサポートされているリージョンと DB エンジン
- Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン
- Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン
- Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン
- RDS Custom でサポートされているリージョンと DB エンジン
- Amazon RDS Proxy でサポートされているリージョンと DB エンジン
- Secrets Manager と Amazon RDS の統合でサポートされているリージョンと DB エンジン
- Amazon RDS ゼロ ETL 統合でサポートされているリージョンと DB エンジン
- Amazon RDS のエンジンネイティブ機能

テーブルの規則

機能セクションのテーブルでは、これらのパターンを使ってバージョン番号と可用性レベルを指定しています。

- バージョン x.y - 使用できるのは明記されたバージョンのみです。
- バージョン x.y 以降 - 指定されているバージョン、およびそのメジャーバージョンのすべてのマイナーバージョンです。例えば「バージョン 10.11 以降」であれば、バージョン 10.11、10.11.1、10.12 が使用可能なことを意味します。

機能クイックリファレンス

次のクイックリファレンス表は、各機能と使用可能な RDS DB エンジンの一覧です。リージョンと特定のバージョンの可用性については、後の機能セクションで説明します。

機能	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
ブルー/グリーンデプロイ	利用不可	[使用可能]	[使用可能]	利用不可	[使用可能]	利用不可
クロスリージョン自動バックアップ	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]

機能	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
クロスリージョンリードレプリカ	利用不可	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]
データベースアクティビティストリーミング	利用不可	利用不可	利用不可	[使用可能]	利用不可	[使用可能]
デュアルスタックモード	利用不可	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]

機能	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
Amazo S3 にスナップショットをエクスポートする	利用不可	[使用可能]	[使用可能]	利用不可	[使用可能]	利用不可
AWS Identity and Access Manag t (IAM) データベース認証	利用不可	[使用可能]	[使用可能]	利用不可	[使用可能]	利用不可
Kerbe 認証	[使用可能]	利用不可	[使用可能]	[使用可能]	[使用可能]	[使用可能]

機能	RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
マルチ AZ DB クラスター	利用不可	利用不可	[使用可能]	利用不可	[使用可能]	利用不可
Perfor ce Insight	利用不可	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]
RDS Custor	利用不可	利用不可	利用不可	[使用可能]	利用不可	[使用可能]
RDS Proxy	利用不可	[使用可能]	[使用可能]	利用不可	[使用可能]	[使用可能]
Secret Manag の統合	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]	[使用可能]

Amazon RDS のブルー/グリーンデプロイでサポートされているリージョンと DB エンジン

ブルー/グリーンデプロイでは、本稼働データベース環境を別の同期されたステージング環境にコピーします。Amazon RDS ブルー/グリーンデプロイを使用すると、本稼働環境に影響を与えずにステージング環境のデータベースに変更を加えることができます。例えば、DB エンジンのメジャーまたはマイナーバージョンのアップグレード、データベースパラメータの変更、スキーマの変更をステージング環境で行うことができます。準備ができたら、ステージング環境を新しい本稼働データベース環境に昇格できます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

ブルー/グリーンデプロイ機能はすべての AWS リージョン でサポートされています。

ブルー/グリーンデプロイ機能は次のエンジンではサポートされていません。

- RDS for Db2
- RDS for SQL Server
- RDS for Oracle

トピック

- RDS for MariaDB を使用したブルー/グリーンデプロイ
- RDS for MySQL によるブルー/グリーンデプロイ
- RDS for PostgreSQL によるブルー/グリーンデプロイ

RDS for MariaDB を使用したブルー/グリーンデプロイ

RDS for MariaDB では、ブルー/グリーンデプロイは次のバージョンでサポートされています。

- RDS for MariaDB 12.3 (使用できるすべてのバージョン)
- RDS for MariaDB 11.8 (使用可能なすべてのバージョン)
- RDS for MariaDB 11.4 (使用可能なすべてのバージョン)
- RDS for MariaDB バージョン 10.2 以降のバージョン 10

RDS for MySQL によるブルー/グリーンデプロイ

RDS for MySQL では、ブルー/グリーンデプロイは次のバージョンでサポートされています。

- RDS for MySQL 8.4 (使用できるすべてのバージョン)
- RDS for MySQL 8.0 (使用できるすべてのバージョン)
- RDS for MySQL 5.7 (使用できるすべてのバージョン)

RDS for PostgreSQL によるブルー/グリーンデプロイ

RDS for PostgreSQL では、ブルー/グリーンデプロイはバージョン 11.1 以降のすべてのメジャーおよびマイナーバージョンでサポートされています。

> **Note**
>
> 特定の条件下では、RDS for PostgreSQL は物理レプリケーションの代わりに論理レプリケーションを使用して、グリーン環境とブルー環境の同期を維持します。詳細については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

Amazon RDS でのクロスリージョン自動バックアップでサポートされているリージョンと DB エンジン

Amazon RDS のバックアップレプリケーションを利用して、RDS DB インスタンスがスナップショットやトランザクションログを送信先のリージョンにレプリケーションするよう設定することができます。DB インスタンスにバックアップレプリケーションを設定した場合、RDS は準備ができると、すべてのスナップショットとトランザクションログのクロスリージョンコピーをスタートします。詳細については、「別の AWS リージョン への自動バックアップのレプリケーション」を参照してください。

バックアップレプリケーションが使用可能な AWS リージョンについては、「別の AWS リージョンへの自動バックアップのレプリケーション」を参照してください。

トピック

- RDS for Db2 によるバックアップレプリケーション
- RDS for MariaDB によるバックアップレプリケーション
- RDS for MySQL によるバックアップレプリケーション
- RDS for Oracle によるバックアップレプリケーション
- RDS for PostgreSQL によるバックアップレプリケーション
- RDS for SQL Server によるバックアップレプリケーション

RDS for Db2 によるバックアップレプリケーション

Amazon RDS は、RDS for Db2 の現在利用可能なすべてのバージョンについて、バックアップレプリケーションをサポートしています。

RDS for MariaDB によるバックアップレプリケーション

Amazon RDS は、RDS for MariaDB の現在利用可能なすべてのバージョンについて、バックアップレプリケーションをサポートしています。

RDS for MySQL によるバックアップレプリケーション

Amazon RDS は、RDS for MySQL の現在利用可能なすべてのバージョンについて、バックアップレプリケーションをサポートしています。

RDS for Oracle によるバックアップレプリケーション

Amazon RDS は、RDS for Oracle の現在利用可能なすべてのバージョンのバックアップレプリケーションをサポートしています。

RDS for PostgreSQL によるバックアップレプリケーション

Amazon RDS は、RDS for PostgreSQL の現在利用可能なすべてのバージョンのバックアップレプリケーションをサポートしています。

RDS for SQL Server によるバックアップレプリケーション

Amazon RDS は、RDS for SQL Server の現在利用可能なすべてのバージョンのバックアップレプリケーションをサポートしています。

Amazon RDS でのクロスリージョンリードレプリカでサポートされているリージョンと DB エンジン

Amazon RDS のリージョン間リードレプリカを使用すると、ソース DB インスタンスとは異なるリージョンに Db2、MariaDB、MySQL、Oracle、PostgreSQL、または SQL Server のリードレプリカを作成することができます。コピー元とコピー先リージョンの考慮事項を含む、クロスリージョンリードレプリカの詳細については、「別の でのリードレプリカの作成AWS リージョン」を参照してください。

トピック

- RDS for Db2 を使用したクロスリージョンリードレプリカ
- RDS for MariaDB を使用したクロスリージョンリードレプリカ
- RDS for MySQL を使用したクロスリージョンリードレプリカ
- RDS for Oracle を使用したクロスリージョンリードレプリカ
- RDS for PostgreSQL を使用したクロスリージョンリードレプリカ
- RDS for SQL Server を使用したクロスリージョンリードレプリカ

RDS for Db2 を使用したクロスリージョンリードレプリカ

RDS for Db2 を使用したクロスリージョンリードレプリカは、以下のバージョンのすべてのリージョンで使用できます。

- RDS for Db2 11.5.9 (使用できるすべてのバージョン)
- RDS for Db2 12.1.4 (使用できるすべてのバージョン)

RDS for MariaDB を使用したクロスリージョンリードレプリカ

RDS for MariaDB を使用したクロスリージョンリードレプリカは、以下のバージョンのすべてのリージョンで使用できます。

- RDS for MariaDB 12.3 (使用できるすべてのバージョン)
- RDS for MariaDB 11.8 (使用可能なすべてのバージョン)
- RDS for MariaDB 11.4 (使用可能なすべてのバージョン)
- RDS for MariaDB 10.11 (使用できるすべてのバージョン)
- RDS for MariaDB 10.6 (使用できるすべてのバージョン)
- RDS for MariaDB 10.5 (使用できるすべてのバージョン)
- RDS for MariaDB 10.4 (使用できるすべてのバージョン)

RDS for MySQL を使用したクロスリージョンリードレプリカ

RDS for MySQL を使用したクロスリージョンリードレプリカは、以下のバージョンのすべてのリージョンで使用できます。

- RDS for MySQL 8.4 (使用できるすべてのバージョン)
- RDS for MySQL 8.0 (使用できるすべてのバージョン)
- RDS for MySQL 5.7 (使用できるすべてのバージョン)

RDS for Oracle を使用したクロスリージョンリードレプリカ

RDS for Oracle のクロスリージョンリードレプリカは、Enterprise Edition を使用するサポートされているすべてのデータベースバージョンのすべての AWS リージョン で使用できます。レプリカは、非 CDB および CDB アーキテクチャのシングルテナント構成でサポートされています。クロス

リージョンリードレプリカは、CDB アーキテクチャのマルチテナント構成ではサポートされていません。

RDS for Oracle でのクロスリージョンリードレプリカの追加要件の詳細については、「RDS for Oracle レプリカの要件と考慮事項」を参照してください。

RDS for PostgreSQL を使用したクロスリージョンリードレプリカ

RDS for PostgreSQL を使用したクロスリージョンリードレプリカは、以下のバージョンのすべてのリージョンで使用できます。

- RDS for PostgreSQL 18 (使用できるすべてのバージョン)
- RDS for PostgreSQL 17 (使用できるすべてのバージョン)
- RDS for PostgreSQL 16 (使用できるすべてのバージョン)
- RDS for PostgreSQL 15 (使用できるすべてのバージョン)
- RDS for PostgreSQL 14 (使用できるすべてのバージョン)
- RDS for PostgreSQL 13 (使用できるすべてのバージョン)
- RDS for PostgreSQL 12 (使用できるすべてのバージョン)
- RDS for PostgreSQL 11 (使用できるすべてのバージョン)
- RDS for PostgreSQL 10 (使用できるすべてのバージョン)

RDS for SQL Server を使用したクロスリージョンリードレプリカ

RDS for SQL Server のクロスリージョンリードレプリカは、Microsoft SQL Server Enterprise Edition を使用する以下のバージョンですべてのリージョンで使用できます。

- RDS for SQL Server 2022
- RDS for SQL Server 2019 (バージョン 15.00.4073.23 以降)
- RDS for SQL Server 2017 (バージョン 14.00.3281.6 以降)
- RDS for SQL Server 2016 (バージョン 13.00.6300.2 以降)

Amazon RDS のデータベースアクティビティストリームでサポートされているリージョンと DB エンジン

Amazon RDS のデータベースアクティビティストリームを使用すると、Oracle データベースと SQL Server データベース内の監査活動を監視し、アラームを設定することができます。詳細については、「データベースアクティビティストリーミングの概要」を参照してください。

データベースアクティビティストリームは、次のエンジンでは使用できません。

- RDS for Db2
- RDS for MariaDB
- RDS for MySQL
- RDS for PostgreSQL

トピック

- RDS for Oracle を使用したデータベースアクティビティストリーミング
- RDS for SQL Server でのデータベースアクティビティストリーム

RDS for Oracle を使用したデータベースアクティビティストリーミング

RDS for Oracle によるデータベースアクティビティストリームでサポートされているエンジンと利用可能なリージョンは以下のとおりです。

> **Note**
>
> Oracle Database 21c 以降では、データベースアクティビティストリームは使用できません。

RDS for Oracle でのデータベースアクティビティストリーミングに関する追加要件の詳細については、「データベースアクティビティストリーミングの概要」を参照してください。

リージョン	RDS for Oracle 19c
米国東部 (バージニア北部)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
米国東部 (オハイオ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
米国西部 (北カリフォルニア)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
米国西部 (オレゴン)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アフリカ (ケープタウン)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (香港)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (ハイデラバード)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (ジャカルタ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (マレーシア)	利用不可
アジアパシフィック (メルボルン)	利用不可

リージョン	RDS for Oracle 19c
アジアパシフィック (ムンバイ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (ニュージーランド)	利用不可
アジアパシフィック (大阪)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (ソウル)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (シンガポール)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (シドニー)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
アジアパシフィック (台北)	利用不可
アジアパシフィック (タイ)	利用不可
アジアパシフィック (東京)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
カナダ (中部)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上

リージョン	RDS for Oracle 19c
カナダ西部 (カルガリー)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
中国 (北京)	利用不可
中国 (寧夏)	利用不可
欧州 (フランクフルト)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (アイルランド)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (ロンドン)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (ミラノ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (パリ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (スペイン)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (ストックホルム)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
欧州 (チューリッヒ)	利用不可

リージョン	RDS for Oracle 19c
イスラエル (テルアビブ)	利用不可
メキシコ (中部)	利用不可
中東 (バーレーン)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
中東 (アラブ首長国連邦)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
南米 (サンパウロ)	Enterprise Edition (EE) または Standard Edition 2 (SE2) を使用する Oracle Database 19.0.0.0.ru-2019-07.rur-2019-07.r1 以上
AWS GovCloud (米国東部)	利用不可
AWS GovCloud (米国西部)	利用不可

RDS for SQL Server でのデータベースアクティビティストリーム

RDS for SQL Server によるデータベースアクティビティストリームでサポートされているエンジンと利用可能なリージョンは以下のとおりです。

RDS for SQL Server でのデータベースアクティビティストリームの追加要件の詳細については、「データベースアクティビティストリーミングの概要」を参照してください。

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	利用不可	利用不可	利用不可
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	利用不可	利用不可	利用不可
イスラエル (テルアビブ)	利用不可	利用不可	利用不可
メキシコ (中部)	利用不可	利用不可	利用不可
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	利用不可	利用不可	利用不可
AWS GovCloud (米国西部)	利用不可	利用不可	利用不可

Amazon RDS のデュアルスタックモードでサポートされているリージョンと DB エンジン

RDS でデュアルスタックモードを使用することで、リソースはインターネットプロトコルバージョン 4 (IPv4)、インターネットプロトコルバージョン 6 (IPv6)、またはその両方を使用して DB インスタンスと通信できます。詳細については、「デュアルスタックモード」を参照してください。

トピック

- RDS for Db2 を使用したデュアルスタックモード
- RDS for MariaDB を使用したデュアルスタックモード
- RDS for MySQL を使用したデュアルスタックモード
- RDS for Oracle を使用したデュアルスタックモード
- RDS for PostgreSQL を使用したデュアルスタックモード
- RDS for SQL Server を使用したデュアルスタックモード

RDS for Db2 を使用したデュアルスタックモード

RDS for Db2 を使用したデュアルスタックモードが利用できるリージョンとエンジンバージョンは、次のとおりです。

リージョン	RDS for Db2 11.5				
米国東部 (バージニア北部)	使用可能なすべてのバージョン				
米国東部 (オハイオ)	使用可能なすべてのバージョン				
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン				

リージョン	RDS for Db2 11.5				
米国西部 (オレゴン)	使用可能なすべてのバージョン				
アフリカ (ケープタウン)	使用可能なすべてのバージョン				
アジアパシフィック (香港)	使用可能なすべてのバージョン				
アジアパシフィック (ハイデラバード)	利用不可				
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン				
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン				
アジアパシフィック (メルボルン)	利用不可				
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン				

リージョン	RDS for Db2 11.5				
アジアパシフィック (ニュージーランド)	利用不可				
アジアパシフィック (大阪)	使用可能なすべてのバージョン				
アジアパシフィック (ソウル)	使用可能なすべてのバージョン				
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン				
アジアパシフィック (シドニー)	使用可能なすべてのバージョン				
アジアパシフィック (台北)	使用可能なすべてのバージョン				
アジアパシフィック (タイ)	使用可能なすべてのバージョン				
アジアパシフィック (東京)	使用可能なすべてのバージョン				
カナダ (中部)	使用可能なすべてのバージョン				

リージョン	RDS for Db2 11.5				
カナダ西部 (カルガリー)	使用可能なすべてのバージョン				
中国 (北京)	利用不可				
中国 (寧夏)	利用不可				
欧州 (フランクフルト)	使用可能なすべてのバージョン				
欧州 (アイルランド)	使用可能なすべてのバージョン				
欧州 (ロンドン)	使用可能なすべてのバージョン				
欧州 (ミラノ)	使用可能なすべてのバージョン				
欧州 (パリ)	使用可能なすべてのバージョン				
欧州 (スペイン)	利用不可				
欧州 (ストックホルム)	使用可能なすべてのバージョン				

リージョン	RDS for Db2 11.5				
欧州 (チューリッヒ)	利用不可				
イスラエル (テルアビブ)	使用可能なすべてのバージョン				
メキシコ (中部)	使用可能なすべてのバージョン				
中東 (バーレーン)	使用可能なすべてのバージョン				
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン				
南米 (サンパウロ)	使用可能なすべてのバージョン				
AWS GovCloud (米国東部)	使用可能なすべてのバージョン				
AWS GovCloud (米国西部)	使用可能なすべてのバージョン				

RDS for MariaDB を使用したデュアルスタックモード

RDS for MariaDB を使用したデュアルスタックモードで使用可能なリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイ	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
デラバード)							
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for MySQL を使用したデュアルスタックモード

RDS for MySQL を使用したデュアルスタックモードで使用可能なリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	利用不可	利用不可	利用不可
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	利用不可	利用不可	利用不可
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	利用不可	利用不可	利用不可
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	利用不可	利用不可	利用不可
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	利用不可	利用不可	利用不可
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for Oracle を使用したデュアルスタックモード

RDS for Oracle を使用したデュアルスタックモードで使用可能なリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for PostgreSQL を使用したデュアルスタックモード

RDS for PostgreSQL を使用したデュアルスタックモードで使用可能なリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
カリフォルニア)	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for SQL Server を使用したデュアルスタックモード

RDS for SQL Server を使用したデュアルスタックモードで使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

Amazon RDS の S3 へのスナップショットデータエクスポートでサポートされているリージョンと DB エンジン

RDS DB スナップショットデータを Amazon S3 バケットにエクスポートできます。すべてのタイプの DB スナップショット (手動スナップショット、自動システムスナップショット、AWS Backup で作成されたスナップショットなど) をエクスポートできます。データをエクスポートすると、Amazon Athena や Amazon Redshift Spectrum などのツールを使用して、エクスポートしたデータを直接分析できます。詳細については、「Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。

S3 へのスナップショットのエクスポートは、次のエンジンでは使用できません。

- RDS for Db2
- RDS for Oracle
- RDS for SQL Server

トピック

- RDS for MariaDB でスナップショットを S3 にエクスポートする
- RDS for MySQL でスナップショットを S3 にエクスポートする
- RDS for PostgreSQL でスナップショットを S3 にエクスポートする

RDS for MariaDB でスナップショットを S3 にエクスポートする

RDS for MariaDB を使用した S3 へのスナップショットエクスポートをサポートしているエンジンバージョンとリージョンは以下のとおりです。

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for MySQL でスナップショットを S3 にエクスポートする

RDS for MySQL を使用した S3 へのスナップショットエクスポートをサポートしているエンジンとリージョンは以下のとおりです。

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for PostgreSQL でスナップショットを S3 にエクスポートする

RDS for PostgreSQL を使用した S3 へのスナップショットのエクスポートを使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オ	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
レゴン)	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
中国(寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州(フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州(アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州(ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州(ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

Amazon RDS での IAM データベース認証でサポートされているリージョンと DB エンジン

Amazon RDS で IAM データベース認証を使用することで、DB インスタンスに接続するときにパスワードなしで認証できます。代わりに、認証トークンを使用します。(詳しくは、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。)

IAM データベース認証は、次のエンジンでは利用できません。

- RDS for Db2
- RDS for Oracle
- RDS for SQL Server

トピック

- RDS for MariaDB による IAM データベース認証
- RDS for MySQL による IAM データベース認証
- RDS for PostgreSQL による IAM データベース認証

RDS for MariaDB による IAM データベース認証

RDS for MariaDB を使用した IAM データベース認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (ハイ	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
デラバード)							
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (ニュージーランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
アジアパシフィック (台北)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (スペイン)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
欧州 (チューリッヒ)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
イスラエル (テルアビブ)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可	利用不可
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
中東 (アラブ首長国連邦)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5	RDS for MariaDB 10.4
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可	利用不可

RDS for MySQL による IAM データベース認証

RDS for MySQL による IAM データベース認証は、すべてのバージョンのすべてのリージョンで利用できます。

- RDS for MySQL 8.4 – 使用できるすべてのバージョン
- RDS for MySQL 8.0 – 使用できるすべてのバージョン
- RDS for MySQL 5.7 – 使用できるすべてのバージョン

RDS for PostgreSQL による IAM データベース認証

RDS for PostgreSQL による IAM データベース認証は、すべてのバージョンのすべてのリージョンで利用できます。

- RDS for PostgreSQL 18 – 使用可能なすべてのバージョン
- RDS for PostgreSQL 17 – 使用可能なすべてのバージョン
- RDS for PostgreSQL 16 – 使用可能なすべてのバージョン

- RDS for PostgreSQL 15 – 使用できるすべてのバージョン
- RDS for PostgreSQL 14 – 使用できるすべてのバージョン
- RDS for PostgreSQL 13 – 使用できるすべてのバージョン
- RDS for PostgreSQL 12 – 使用できるすべてのバージョン
- RDS for PostgreSQL 11 – 使用できるすべてのバージョン
- RDS for PostgreSQL 10 – 使用できるすべてのバージョン

Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン

Amazon RDS の Kerberos 認証を使用して、Kerberos および Microsoft Active Directory を使用して、データベースユーザーの外部認証に対応できます。Kerberos と Active Directory を使用することで、シングルサインオンとデータベースユーザーの一元化認証という利点が得られます。

Kerberos 認証は、次のエンジンでは使用できません。

- RDS for MariaDB

ほとんどの AWS リージョンは AWS アカウントでデフォルトでアクティブになりますが、特定のリージョンは手動で選択した場合にのみアクティブになります。これらのリージョンは、オプトインリージョンと呼ばれます。これに対して、AWS アカウントの作成後すぐにデフォルトでアクティブになるリージョンは、商用リージョンまたは単にリージョンと呼ばれます。オプトインリージョンの場合は、`directoryservice.rds.region_name.amazonaws.com` という形式のリージョン化されたサービスプリンシパルを使用する必要があります。例えば、アフリカ (ケープタウン) の場合は、信頼ポリシーにサービスプリンシパル `directoryservice.rds.af-south-1.amazonaws.com` を追加する必要があります。イスラエル (テルアビブ) より後に開設された AWS リージョン (カナダ西部 (カルガリー) 以降) では、エンジン固有のサービスプリンシパル `directoryservice-engine.rds.region_name.amazonaws.com` を使用する必要があります。例えば、カナダ西部 (カルガリー) の RDS PostgreSQL の場合、信頼ポリシーにサービスプリンシパル `directoryservice-postgresql.rds.ca-west-1.amazonaws.com` を追加する必要があります。詳細については、「Kerberos 認証」を参照してください。

トピック

- RDS for Db2 を使用した Kerberos 認証
- RDS for MySQL を使用した Kerberos 認証

- RDS for Oracle を使用した Kerberos 認証
- RDS for PostgreSQL を使用した Kerberos 認証
- RDS for SQL Server を使用したKerberos 認証

RDS for Db2 を使用した Kerberos 認証

RDS for Db2 による Kerberos 認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for Db2 11.5
米国東部 (バージニア北部)	すべてのバージョン
米国東部 (オハイオ)	すべてのバージョン
米国西部 (北カリフォルニア)	すべてのバージョン
米国西部 (オレゴン)	すべてのバージョン
アフリカ (ケープタウン)	すべてのバージョン
アジアパシフィック (香港)	利用不可
アジアパシフィック (ハイデラバード)	すべてのバージョン
アジアパシフィック (ジャカルタ)	すべてのバージョン
アジアパシフィック (マレーシア)	利用不可
アジアパシフィック (メルボルン)	すべてのバージョン
アジアパシフィック (ムンバイ)	すべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可
アジアパシフィック (大阪)	利用不可
アジアパシフィック (ソウル)	すべてのバージョン
アジアパシフィック (シンガポール)	すべてのバージョン

リージョン	RDS for Db2 11.5
アジアパシフィック (シドニー)	すべてのバージョン
アジアパシフィック (台北)	利用不可
アジアパシフィック (タイ)	利用不可
アジアパシフィック (東京)	すべてのバージョン
カナダ (中部)	すべてのバージョン
カナダ西部 (カルガリー)	利用不可
中国 (北京)	利用不可
中国 (寧夏)	利用不可
欧州 (フランクフルト)	すべてのバージョン
欧州 (アイルランド)	すべてのバージョン
欧州 (ロンドン)	すべてのバージョン
欧州 (ミラノ)	すべてのバージョン
欧州 (パリ)	利用不可
欧州 (スペイン)	すべてのバージョン
欧州 (ストックホルム)	すべてのバージョン
欧州 (チューリッヒ)	すべてのバージョン
イスラエル (テルアビブ)	すべてのバージョン
メキシコ (中部)	利用不可
中東 (バーレーン)	すべてのバージョン
中東 (アラブ首長国連邦)	すべてのバージョン

リージョン	RDS for Db2 11.5
南米 (サンパウロ)	すべてのバージョン
AWS GovCloud (米国東部)	すべてのバージョン
AWS GovCloud (米国西部)	すべてのバージョン

RDS for MySQL を使用した Kerberos 認証

RDS for MySQL による Kerberos 認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
米国東部 (バージニア北部)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国東部 (オハイオ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (北カリフォルニア)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (オレゴン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アフリカ (ケープタウン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (香港)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ハイデラバード)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ジャカルタ)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
アジアパシフィック (マレーシア)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (メルボルン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ムンバイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ニュージーランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (大阪)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ソウル)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シンガポール)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シドニー)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (台北)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (タイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (東京)	すべてのバージョン	すべてのバージョン	すべてのバージョン
カナダ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
カナダ西部 (カルガリー)	すべてのバージョン	すべてのバージョン	すべてのバージョン
中国 (北京)	すべてのバージョン	すべてのバージョン	すべてのバージョン
中国 (寧夏)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (フランクフルト)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (アイルランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ロンドン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ミラノ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (パリ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (スペイン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ストックホルム)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (チューリッヒ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
イスラエル (テルアビブ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
メキシコ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン
中東 (バーレーン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
中東 (アラブ首長国連邦)	すべてのバージョン	すべてのバージョン	すべてのバージョン
南米 (サンパウロ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国東部)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7 (RDS 延長サポートが必要)
AWS GovCloud (米国西部)	すべてのバージョン	すべてのバージョン	すべてのバージョン

RDS for Oracle を使用した Kerberos 認証

RDS for Oracle による Kerberos 認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
米国東部 (バージニア北部)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国東部 (オハイオ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (北カリフォルニア)	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (オレゴン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アフリカ (ケープタウン) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (香港) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ハイデラバード) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
アジアパシフィック (ジャカルタ) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ムンバイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	利用不可	利用不可	利用不可
アジアパシフィック (ソウル)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シンガポール)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シドニー)	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可
アジアパシフィック (東京)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
カナダ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン
カナダ西部 (カルガリー)	利用不可	利用不可	利用不可
中国 (北京)	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可
欧州 (フランクフルト)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (アイルランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ロンドン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ミラノ) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (パリ)	利用不可	利用不可	利用不可
欧州 (スペイン) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ストックホルム)	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (チューリッヒ) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
イスラエル (テルアビブ) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
メキシコ (中部)	利用不可	利用不可	利用不可
中東 (バーレーン) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for Oracle 26ai	RDS for Oracle 21c	RDS for Oracle 19c
中東 (アラブ首長国連邦) (オプトインリージョン)	すべてのバージョン	すべてのバージョン	すべてのバージョン
南米 (サンパウロ)	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国東部)	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国西部)	すべてのバージョン	すべてのバージョン	すべてのバージョン

RDS for PostgreSQL を使用した Kerberos 認証

RDS for PostgreSQL による Kerberos 認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
米国東部 (バージニア北部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国東部 (オハイオ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (北カリ	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
フォルニア)									
米国西部 (オレゴン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アフリカ (ケープタウン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (香港)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ハイデラバード)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ジャカルタ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (マレーシア)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (メルボルン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ムンバイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ニュージーランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (大阪)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ソウル)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シンガポール)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (シドニー)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (台北)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (タイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (東京)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
カナダ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
カナダ西部 (カルガリー)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
中国 (北京)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
中国 (寧夏)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (フランクフルト)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (アイルランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ロンドン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ミラノ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (パリ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (スペイン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ストックホルム)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (チューリッヒ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
イスラエル (テルアビブ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
メキシコ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
中東 (バーレーン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
中東 (アラブ首長国連邦)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
南米 (サンパウロ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国東部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国西部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

RDS for SQL Server を使用したKerberos 認証

RDS for SQL Server による Kerberos 認証で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国東部 (バージニア北部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国東部 (オハイオ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (北カリフォルニア)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
米国西部 (オレゴン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アフリカ (ケープタウン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (香港)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ハイデラバード)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ムンバイ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可

リージョン	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
アジアパシフィック (大阪)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (ソウル)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シンガポール)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (シドニー)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (東京)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
カナダ (中部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
カナダ西部 (カルガリー)	利用不可	利用不可	利用不可	利用不可
中国 (北京)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
中国 (寧夏)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
欧州 (フランクフルト)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (アイルランド)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ロンドン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ミラノ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (パリ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (スペイン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (ストックホルム)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
欧州 (チューリッヒ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
イスラエル (テルアビブ)	利用不可	利用不可	利用不可	利用不可
メキシコ (中部)	利用不可	利用不可	利用不可	利用不可
中東 (バーレーン)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
中東 (アラブ首長国連邦)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
南米 (サンパウロ)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

リージョン	RDS for SQL Server 2022	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
AWS GovCloud (米国東部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン
AWS GovCloud (米国西部)	すべてのバージョン	すべてのバージョン	すべてのバージョン	すべてのバージョン

Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン

Amazon RDS のマルチ AZ DB クラスターデプロイとは、2 つの読み取り可能なスタンバイ DB インスタンスを備えた Amazon RDS の高可用性デプロイモードです。マルチ AZ DB クラスターには、同じのリージョンに 3 つの別々のアベイラビリティーゾーンに 1 つのライター DB インスタンスと 2 つのリーダー DB インスタンスがあります。マルチ AZ DB クラスターは、マルチ AZ DB インスタンスの配置と比較して、高可用性、読み取りワークロードの容量の増加、および書き込みレイテンシーの低減を提供します。(詳しくは、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。)

マルチ AZ DB クラスターは、次のエンジンでは使用できません。

- RDS for Db2
- RDS for MariaDB
- RDS for Oracle
- RDS for SQL Server

トピック

- RDS for MySQL を使用したマルチ AZ DB クラスター
- RDS for PostgreSQL を使用したマルチ AZ DB クラスター

RDS for MySQL を使用したマルチ AZ DB クラスター

RDS for MySQL を使用したマルチ AZ DB クラスターでサポートされているリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	利用不可	利用不可
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	利用不可	利用不可

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0
AWS GovCloud (米国西部)	利用不可	利用不可

AWS CLI を使用して、特定の DB インスタンスクラスにおいてリージョンで使用できるバージョンを一覧表示できます。DB インスタンスクラスを変更して、使用可能なエンジンバージョンを表示します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options \
--engine mysql \
--db-instance-class db.r5d.large \
--query '*[]|[?SupportsClusters == `true`].[EngineVersion]'  \
--output text
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options ^
--engine mysql ^
--db-instance-class db.r5d.large ^
--query "*[]|[?SupportsClusters == `true`].[EngineVersion]"  ^
--output text
```

RDS for PostgreSQL を使用したマルチ AZ DB クラスター

RDS for PostgreSQL を使用したマルチ AZ DB クラスターで使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
米国東部 (バージニア北部)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
米国東部 (オハイオ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
米国西部 (北カリフォルニア)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
米国西部 (オレゴン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アフリカ (ケープタウン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (香港)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (ハイデラバード)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (ジャカルタ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
アジアパシフィック (マレーシア)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (メルボルン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (ムンバイ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (ソウル)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
アジアパシフィック (シンガポール)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (シドニー)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
アジアパシフィック (台北)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (東京)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
カナダ (中部)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
カナダ西部 (カルガリー)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
中国 (北京)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
中国 (寧夏)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (フランクフルト)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (アイルランド)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (ロンドン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (ミラノ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (パリ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
欧州 (スペイン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (ストックホルム)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
欧州 (チューリッヒ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
イスラエル (テルアビブ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
メキシコ (中部)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
中東 (バーレーン)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
中東 (アラブ首長国連邦)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13
南米 (サンパウロ)	すべての PostgreSQL 18 バージョン	すべての PostgreSQL 17 バージョン	すべての PostgreSQL 16 バージョン	すべての PostgreSQL 15 バージョン	バージョン 14.5 以降	バージョン 13.4、バージョン 13.7 以降
AWS GovCloud (米国東部)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
AWS GovCloud (米国西部)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可

AWS CLI を使用して、特定の DB インスタンスクラスにおいてリージョンで使用できるバージョンを一覧表示できます。DB インスタンスクラスを変更して、使用可能なエンジンバージョンを表示します。

Linux、macOS、Unix の場合:

```
aws rds describe-orderable-db-instance-options \
--engine postgres \
--db-instance-class db.r5d.large \
--query '*[]|[?SupportsClusters == `true`].[EngineVersion]'  \
--output text
```

Windows の場合:

```
aws rds describe-orderable-db-instance-options ^
--engine postgres ^
--db-instance-class db.r5d.large ^
--query "*[]|[?SupportsClusters == `true`].[EngineVersion]"  ^
--output text
```

Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
> Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までにニーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
> アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
> CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Amazon RDS の Performance Insights は、既存の Amazon RDS モニタリング機能を拡張して、データベースのパフォーマンスを明確にし、分析しやすくします。Performance Insights ダッシュボードを使用して Amazon RDS DB インスタンスのデータベースロードを視覚化できます。ロードを待機、SQL ステートメント、ホスト、ユーザー別にフィルタリングすることもできます。詳細につい

ては、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。

Performance Insights は、RDS for Db2 以外のすべての RDS DB エンジンで使用できます。

利用可能な DB エンジンでは、使用可能なすべてのエンジンバージョンとすべての AWS リージョンで Performance Insights を使用できます。

Performance Insights 機能のリージョン、DB エンジン、およびインスタンスクラスのサポート情報については、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

RDS Custom でサポートされているリージョンと DB エンジン

Amazon RDS Custom でデータベース管理タスクとオペレーションが自動化されます。RDS Custom を使用して、データベース管理者としてデータベース環境とオペレーティングシステムにアクセスしてカスタマイズできます。RDS Custom を使用すると、レガシー、カスタム、パッケージのアプリケーション要件を満たすカスタマイズが可能です。詳細については、「Amazon RDS Custom」を参照してください。

RDS Custom は次の DB エンジンでのみサポートされています。

トピック

- RDS Custom for Oracle でサポートされているリージョンと DB エンジン
- RDS Custom for SQL Server でサポートされているリージョンと DB エンジン

RDS Custom for Oracle でサポートされているリージョンと DB エンジン

RDS for Oracle で使用可能なリージョンとエンジンバージョンは以下のとおりです。

リージョン	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
米国東部 (バージニア北部)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
米国東部 (オハイオ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2

リージョン	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
米国西部 (北カリフォルニア)	利用不可	利用不可	利用不可
米国西部 (オレゴン)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アフリカ (ケープタウン)	利用不可	利用不可	利用不可
アジアパシフィック (香港)	利用不可	利用不可	利用不可
アジアパシフィック (ジャカルタ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	利用不可	利用不可	利用不可
アジアパシフィック (ムンバイ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2

リージョン	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
アジアパシフィック (ソウル)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アジアパシフィック (シンガポール)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アジアパシフィック (シドニー)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可
アジアパシフィック (東京)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
カナダ (中部)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
カナダ西部 (カルガリー)	利用不可	利用不可	利用不可
中国 (北京)	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可
欧州 (フランクフルト)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2

リージョン	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
欧州 (アイルランド)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
欧州 (ロンドン)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
欧州 (ミラノ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
欧州 (パリ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
欧州 (ストックホルム)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
イスラエル (テルアビブ)	利用不可	利用不可	利用不可
メキシコ (中部)	利用不可	利用不可	利用不可
中東 (バーレーン)	利用不可	利用不可	利用不可
中東 (アラブ首長国連邦)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
南米 (サンパウロ)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2

リージョン	Oracle Database 19c	Oracle Database 18c	Oracle Database 12c
AWS GovCloud (米国東部)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2
AWS GovCloud (米国西部)	2021 年 1 月以降の RU/RUR を使用した 19c	2021 年 1 月以降の RU/RUR を使用した 18c	2021 年 1 月以降の RU/RUR を使用した 12.1 および 12.2

RDS Custom for SQL Server でサポートされているリージョンと DB エンジン

RDS Custom for SQL Server は、RDS が提供するエンジンバージョン (RPEV) またはカスタムエンジンバージョン (CEV) のいずれかを使用してデプロイできます。

- RPEV を使用する場合は、デフォルトの Amazon マシンイメージ (AMI) と SQL Server のインストールが含まれます。RDS Custom for SQL Server は、RDS が提供するエンジンバージョン (RPEV) のメンテナンスアクションによるオペレーティングシステムの更新もサポートしています。オペレーティングシステム (OS) をカスタマイズまたは変更した場合、パッチ適用、スナップショット復元、または自動リカバリの実行中に変更が保持されない場合があります。
- CEV を使用する場合は、Microsoft SQL Server がプリインストールされた独自の AMI を選択するか、独自のメディアを使用してインストールする SQL Server のいずれかを選択します。AWS が提供されている CEV を使用する場合は、RDS Custom for SQL Server がサポートする累積更新 (CU) を含む、AWS で利用可能な最新の Amazon EC2 イメージ (AMI) を選択します。CEV では、OS と SQL Server の設定は、企業のニーズに合わせてカスタマイズできます。

現在、Windows 2019 オペレーティングシステムのみを使用できます。

RDS Custom for SQL Server で使用可能な AWS リージョン と DB エンジンバージョンは以下のとおりです。エンジンバージョンのサポートは、RPEV で RDS Custom for SQL Server、AWS が提供する CEV、またはカスタマーが提供する CEV を使用しているかによって異なります。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
米国東部 (バージニア北部)	SQL Server 2022 Enterprise、Standar	SQL Server 2022 Enterprise、Standar	SQL Server 2022 Enterprise、Standar

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
	d、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	d、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	d、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
米国東部 (オハイオ)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
米国西部 (北カリフォルニア)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
米国西部 (オレゴン)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アフリカ (ケープタウン)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
アジアパシフィック (香港)	利用不可	利用不可	利用不可
アジアパシフィック (ハイデラバード)	利用不可	利用不可	利用不可
アジアパシフィック (ジャカルタ)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	利用不可	利用不可	利用不可
アジアパシフィック (ムンバイ)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (大阪)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (ソウル)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (シンガポール)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (シドニー)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
アジアパシフィック (東京)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
カナダ (中部)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
カナダ西部 (カルガリー)	利用不可	利用不可	利用不可
中国 (北京)	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
欧州 (フランクフルト)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
欧州 (アイルランド)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
欧州 (ロンドン)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
欧州 (ミラノ)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
欧州 (パリ)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
欧州 (スペイン)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
欧州 (ストックホルム)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
欧州 (チューリッヒ)	利用不可	利用不可	利用不可
イスラエル (テルアビブ)	利用不可	利用不可	利用不可
メキシコ (中部)	利用不可	利用不可	利用不可
中東 (バーレーン)	利用不可	利用不可	利用不可

リージョン	RPEV	AWS が提供する CEV	カスタマーが提供する CEV
中東 (アラブ首長国連邦)	利用不可	利用不可	利用不可
南米 (サンパウロ)	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。	SQL Server 2022 Enterprise、Standard、Developer、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。SQL Server 2019 Enterprise、Standard、Developer、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
AWS GovCloud (米国東部)	利用不可	利用不可	利用不可
AWS GovCloud (米国西部)	利用不可	利用不可	利用不可

Amazon RDS Proxy でサポートされているリージョンと DB エンジン

Amazon RDS Proxy は、確立済みのデータベース接続をプーリングし共有することでアプリケーションのスケーラビリティを高める、フルマネージドの高可用性データベースプロキシです。詳細については、「Amazon RDS Proxy」を参照してください。

RDS プロキシは次のエンジンでは使用できません。

- RDS for Db2
- RDS for Oracle

トピック

- RDS for MariaDB を使用した RDS Proxy
- RDS for MySQL を使用した RDS Proxy
- RDS for PostgreSQL を使用した RDS Proxy
- RDS for SQL Server を使用した RDS Proxy

RDS for MariaDB を使用した RDS Proxy

RDS for MariaDB を使用した RDS Proxy で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリ	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
フォルニア)						
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MariaDB 12.3	RDS for MariaDB 11.8	RDS for MariaDB 11.4	RDS for MariaDB 10.11	RDS for MariaDB 10.6	RDS for MariaDB 10.5
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for MySQL を使用した RDS Proxy

RDS for MySQL を使用した RDS Proxy で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4	RDS for MySQL 8.0	RDS for MySQL 5.7
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for PostgreSQL を使用した RDS Proxy

RDS for PostgreSQL を使用した RDS Proxy で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オ	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす	使用可能なす

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
ハイオ)	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン	べてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for PostgreSQL 18	RDS for PostgreSQL 17	RDS for PostgreSQL 16	RDS for PostgreSQL 15	RDS for PostgreSQL 14	RDS for PostgreSQL 13	RDS for PostgreSQL 12	RDS for PostgreSQL 11	RDS for PostgreSQL 10
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

RDS for SQL Server を使用した RDS Proxy

RDS for SQL Server を使用した RDS Proxy で使用できるリージョンとエンジンバージョンは以下のとおりです。

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (台北)	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (北京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中国 (寧夏)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
メキシコ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for SQL Server 2019	RDS for SQL Server 2017	RDS for SQL Server 2016
AWS GovCloud (米国東部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovCloud (米国西部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

Secrets Manager と Amazon RDS の統合でサポートされているリージョンと DB エンジン

AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (データベースパスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。Secrets Manager の詳細については、AWS Secrets Manager ユーザーガイドを参照してください。

Amazon RDS DB インスタンスまたはマルチ AZ DB クラスターのシークレットマネージャーで、Amazon RDS がマスターユーザーパスワードを管理するように指定できます。RDS はパスワードを生成してシークレットマネージャーに保存し、定期的にローテーションします。詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。

Secrets Manager の統合は、すべての AWS リージョン で利用可能です。

Amazon RDS ゼロ ETL 統合でサポートされているリージョンと DB エンジン

RDS ゼロ ETL 統合は、トランザクションデータを Amazon RDS DB インスタンスに書き込んだ後に Amazon Redshift または Amazon SageMaker で利用できるようにするためのフルマネージドソリューションです。詳細については、「ゼロ ETL 統合」を参照してください。

トピック

- RDS for MySQL とのゼロ ETL 統合
- RDS for PostgreSQL とのゼロ ETL 統合
- RDS for Oracle とのゼロ ETL 統合

RDS for MySQL とのゼロ ETL 統合

Amazon Redshift および Amazon SageMaker との RDS for MySQL ゼロ ETL 統合は、以下のリージョンとエンジンバージョンで利用できます。

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
米国東部 (バージニア北部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
米国西部 (オレゴン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
アフリカ (ケープタウン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
アジアパシフィック (香港)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
アジアパシフィック (ニュージーランド)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (大阪)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
アジアパシフィック (ソウル)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
アジアパシフィック (台北)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (タイ)	利用不可	利用不可	利用不可	利用不可
アジアパシフィック (東京)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
カナダ西部 (カルガリー)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
中国 (北京)	利用不可	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可	利用不可
欧州 (フランクフルト)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
欧州 (ミラノ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
欧州 (パリ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
欧州 (スペイン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
欧州 (ストックホルム)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
イスラエル (テルアビブ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
メキシコ (中部)	利用不可	利用不可	利用不可	利用不可
メキシコ (中部)	利用不可	利用不可	利用不可	利用不可
中東 (バーレーン)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	利用不可	利用不可
南米 (サンパウロ)	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン	使用可能なすべてのバージョン
AWS GovClo (米国東部)	利用不可	利用不可	利用不可	利用不可

リージョン	RDS for MySQL 8.4 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon Redshift とのゼロ ETL 統合	RDS for MySQL 8.4 の Amazon SageMaker とのゼロ ETL 統合	RDS for MySQL 8.0 の Amazon SageMaker とのゼロ ETL 統合
AWS GovClo (米国西部)	利用不可	利用不可	利用不可	利用不可

RDS for PostgreSQL とのゼロ ETL 統合

RDS for PostgreSQL と Amazon Redshift のゼロ ETL 統合では、以下のリージョンとエンジンバージョンを利用できます。

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
米国東部 (バージニア北部)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
米国東部 (オハイオ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
米国西部 (北カリフォルニア)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
米国西部 (オレゴン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アフリカ (ケープタウン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (香港)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (ハイデラバード)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (ジャカルタ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
アジアパシフィック (マレーシア)	利用不可	利用不可	利用不可
アジアパシフィック (メルボルン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (ムンバイ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (大阪)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (ソウル)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
アジアパシフィック (シンガポール)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (シドニー)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (タイ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
アジアパシフィック (東京)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
カナダ (中部)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
カナダ西部 (カルガリー)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
中国 (北京)	利用不可	利用不可	利用不可
中国 (寧夏)	利用不可	利用不可	利用不可
欧州 (フランクフルト)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (アイルランド)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (ロンドン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (ミラノ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (パリ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (スペイン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (ストックホルム)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
欧州 (チューリッヒ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降

リージョン	RDS for PostgreSQL 15 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 16 の Amazon Redshift とのゼロ ETL 統合	RDS for PostgreSQL 17 の Amazon Redshift とのゼロ ETL 統合
イスラエル (テルアビブ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
中東 (バーレーン)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
中東 (アラブ首長国連邦)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
南米 (サンパウロ)	バージョン 15.11 以降	バージョン 16.7 以降	バージョン 17.3 以降
AWS GovCloud (米国東部)	利用不可	利用不可	利用不可
AWS GovCloud (米国西部)	利用不可	利用不可	利用不可

RDS for Oracle とのゼロ ETL 統合

Amazon Redshift との RDS for Oracle ゼロ ETL 統合は、以下のリージョンとエンジンバージョンで利用できます。

リージョン	RDS for Oracle 19c の Amazon Redshift とのゼロ ETL 統合
米国東部 (バージニア北部)	使用可能なすべてのバージョン
米国東部 (オハイオ)	使用可能なすべてのバージョン
米国西部 (北カリフォルニア)	使用可能なすべてのバージョン
米国西部 (オレゴン)	使用可能なすべてのバージョン
アフリカ (ケープタウン)	使用可能なすべてのバージョン
アジアパシフィック (香港)	使用可能なすべてのバージョン
アジアパシフィック (ハイデラバード)	使用可能なすべてのバージョン
アジアパシフィック (ジャカルタ)	使用可能なすべてのバージョン
アジアパシフィック (マレーシア)	利用不可
アジアパシフィック (メルボルン)	使用可能なすべてのバージョン
アジアパシフィック (ムンバイ)	使用可能なすべてのバージョン
アジアパシフィック (大阪)	使用可能なすべてのバージョン
アジアパシフィック (ソウル)	使用可能なすべてのバージョン
アジアパシフィック (シンガポール)	使用可能なすべてのバージョン
アジアパシフィック (シドニー)	使用可能なすべてのバージョン
アジアパシフィック (タイ)	使用可能なすべてのバージョン

リージョン	RDS for Oracle 19c の Amazon Redshift とのゼロ ETL 統合
アジアパシフィック (東京)	使用可能なすべてのバージョン
カナダ (中部)	使用可能なすべてのバージョン
カナダ西部 (カルガリー)	使用可能なすべてのバージョン
中国 (北京)	利用不可
中国 (寧夏)	利用不可
欧州 (フランクフルト)	使用可能なすべてのバージョン
欧州 (アイルランド)	使用可能なすべてのバージョン
欧州 (ロンドン)	使用可能なすべてのバージョン
欧州 (ミラノ)	使用可能なすべてのバージョン
欧州 (パリ)	使用可能なすべてのバージョン
欧州 (スペイン)	使用可能なすべてのバージョン
欧州 (ストックホルム)	使用可能なすべてのバージョン
欧州 (チューリッヒ)	使用可能なすべてのバージョン
イスラエル (テルアビブ)	使用可能なすべてのバージョン
中東 (バーレーン)	使用可能なすべてのバージョン
中東 (アラブ首長国連邦)	使用可能なすべてのバージョン
南米 (サンパウロ)	使用可能なすべてのバージョン
AWS GovCloud (米国東部)	利用不可
AWS GovCloud (米国西部)	利用不可

Amazon RDS のエンジンネイティブ機能

Amazon RDS データベース エンジンは、多くの最も一般的なエンジン ネイティブの機能もサポートしています。これらの機能は、このページに記載されている Amazon RDS ネイティブの機能とは異なります。一部のエンジンネイティブの機能には、制限付きのサポートまたは制限された特権があります。

エンジンネイティブの機能の詳細については、以下を参照してください。

- Amazon RDS for Db2 の機能
- Amazon RDS での MariaDB 機能のサポート
- Amazon RDS での MySQL 機能のサポート
- RDS for Oracle の機能
- Amazon RDS for PostgreSQL でサポートされている PostgreSQL の機能を使用する
- Amazon RDS での Microsoft SQL Server の機能

Amazon RDS DB インスタンスの請求

Amazon RDS インスタンスは、以下のコンポーネントに基づいて請求されます。

- DB インスタンス時間 (1 時間あたり) - DB インスタンスの DB インスタンスクラス (db.t2.small や db.m4.large など) に基づきます。料金は 1 時間単位で表示されますが、請求の計算方法には秒単位が適用され、時間は 10 進数の形式で表示されます。RDS の使用料は 1 秒ごとに課金され、10 分未満の場合は 10 分の料金が発生します。詳細については、「DB インスタンスクラス」を参照してください。
- ストレージ (1 か月あたりの GiB) - DB インスタンスにプロビジョニングしたストレージ容量。準備したストレージ容量を当月以内に拡張した場合、請求は比例配分されます。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。
- 入出力 (I/O) リクエスト (1 か月あたり 100 万リクエスト) - 請求期間内に行ったストレージ I/O リクエストの合計数。Amazon RDS マグネティックストレージ に対するものに限ります。
- プロビジョンド IOPS (1 か月あたりの IOPS) - プロビジョンド IOPS レートで、Amazon RDS プロビジョンド IOPS (SSD) および汎用 (SSD) gp3 ストレージの場合、消費された IOPS には関係ありません。EBS ボリュームのプロビジョンドストレージは、1 秒ごとに課金され、10 分未満の場合は 10 分の料金が発生します。
- バックアップストレージ (1 か月あたりの GiB) - バックアップストレージは、自動データベースバックアップおよび作成したアクティブなデータベースのスナップショットに関連付けられているストレージです。バックアップ保持期間を延長するか、追加のデータベーススナップショットを撮ると、データベースが消費するバックアップストレージが増加します。1 秒単位の請求はバックアップストレージには適用されません (1 か月あたり GB 単位で請求されます)。

  詳細については、「データのバックアップ、復元、エクスポート」を参照してください。
- データ転送 (GB あたり) - DB インスタンスと、インターネットおよび AWS リージョンの間で送受信されるデータ転送。便利な例については、AWS ブログ記事「AWS マネージドデータベースのデータ転送コストの調査」を参照してください。

Amazon RDS には、ニーズに基づいてコストを最適化するための以下の購入オプションがあります。

- オンデマンドインスタンス - 使用した DB インスタンス時間に対して時間単位でお支払いいただきます。料金は 1 時間単位で表示されますが、請求の計算方法には秒単位が適用され、時間は 10 進形式で表示されます。現在、RDS の使用料は 1 秒ごとに課金され、10 分未満の場合は 10 分の料金が発生します。

- リザーブドインスタンス - DB インスタンスを 1 年間または 3 年間予約することで、オンデマンド DB インスタンスの料金と比べて大幅な割引が得られます。リザーブドインスタンスの使用状況では 1 時間以内に複数のインスタンスを起動、削除、スタート、終了することができ、すべてのインスタンスにおいてリザーブドインスタンスのメリットが得られます。

Amazon RDS の料金情報については、「Amazon RDS の料金」ページを参照してください。

トピック

- Amazon RDS 用のオンデマンド DB インスタンス
- Amazon RDS 向けリザーブド DB インスタンス

Amazon RDS 用のオンデマンド DB インスタンス

Amazon RDS オンデマンド DB インスタンスは、DB インスタンスのクラス (db.t3.small や db.m5.large など) に基づいて請求されます。Amazon RDS の料金情報については、Amazon RDS の製品ページを参照してください。

DB インスタンスの課金は DB インスタンスが利用可能になった時点からスタートされます。料金は 1 時間単位で表示されますが、請求の計算方法には秒単位が適用され、時間は 10 進数の形式で表示されます。Amazon RDS の使用料は 1 秒ごとに課金され、10 分未満の場合は 10 分の料金が発生します。請求可能な設定の変更 (例: コンピューティング容量またはストレージ容量のスケーリング) の場合は、10 分の料金が請求されます。課金は DB インスタンスが終了するまで継続します。終了とは、DB インスタンスが削除された場合、または DB インスタンスに障害が発生した場合です。

DB インスタンスに対する課金が不要になった場合は、これ以上 DB インスタンス時間に請求が行われないようにインスタンスを停止するか、削除する必要があります。課金される DB インスタンスの状態に関する詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

停止した DB インスタンス

DB インスタンスが停止していても、プロビジョンド IOPS を含むプロビジョニング済みストレージに対して課金されます。また、指定された保持ウィンドウ内の手動スナップショットや自動バックアップのストレージを含むバックアップストレージに対しても課金されます。DB インスタンス時間に対しては請求されません。

マルチ AZ DB インスタンス

マルチ AZ 設定は、異なるアベイラビリティーゾーンで同期スタンバイレプリカを自動的にプロビジョニングして維持することで、データの耐久性と可用性を向上させます。マルチ AZ 配置は、追加のリソースと可用性の向上により、シングル AZ 配置よりも料金が高くなり、追加のスタンバイインスタンスと関連リソースにより、コストが約 2 倍になる可能性があります。

マルチ AZ 料金に関する以下の重要な詳細を考慮してください。

- コンピューティングコスト: プライマリインスタンスとスタンバイインスタンスの両方について、DB インスタンス時間ごとに請求されます。
- ストレージコスト: プライマリインスタンスとスタンバイインスタンスの両方で、プロビジョニングしたストレージに GB/月単位で課金されます。

- データ転送コスト: プライマリインスタンスとスタンバイインスタンス間のレプリケーションはコストに含まれますが、他のデータ転送は使用量に応じて課金される場合があります。

特定のユースケースと AWS リージョン に基づいて月額コストを正確に見積もるには、AWS 料金見積りツール を使用できます。このツールを使用すると、設定の詳細を入力し、包括的なコスト内訳を提供できます。

> **Note**
>
> 料金は変更される可能性があります。最新情報については、Amazon RDS の料金ページを参照してください。

Amazon RDS 向けリザーブド DB インスタンス

リザーブド DB インスタンスを使用することで、DB インスタンスを 1 年間または 3 年間予約できます。オンデマンド DB インスタンスの料金と比べて、リザーブド DB インスタンスには大幅な割引が適用されます。リザーブド DB インスタンスは物理インスタンスと言うよりも、アカウントで特定のオンデマンド DB インスタンスを使用した場合に適用される請求の割引と言えます。リザーブド DB インスタンスの割引は、インスタンスタイプと AWS リージョン に関連付けられています。

リザーブド DB インスタンスの一般的な使用プロセスとしては、まず使用可能なリザーブド DB インスタンスのタイプに関する情報を取得します。次に、該当するタイプのリザーブド DB インスタンスを購入します。最後に、既存のリザーブド DB インスタンスに関する情報を取得します。

リザーブド DB インスタンスの購入とリザーブド DB インスタンスの請求の表示については、以下のセクションを参照してください。

- Amazon RDS のリザーブド DB インスタンスの購入
- Amazon RDS のリザーブド DB インスタンスの請求の表示

リザーブド DB インスタンスの概要

Amazon RDS のリザーブド DB インスタンスを購入すると、このリザーブド DB インスタンスの該当期間中、特定の DB インスタンスタイプに対して割引料金が適用されます。Amazon RDS のリザーブド DB インスタンスを使用するには、オンデマンドインスタンスの場合と同様に、新しい DB インスタンスを作成します。

新しく作成する DB インスタンスの仕様は、次のリザーブド DB インスタンスの仕様と同じである必要があります。

- AWS リージョン
- DB エンジン (DB エンジンのバージョン番号は一致する必要はありません)。
- DB インスタンスのタイプ
- DB インスタンスサイズ (RDS for Db2、RDS for SQL Server、および RDS for Oracle ライセンス込み)
- エディション (RDS for Db2、RDS for SQL Server、および RDS for Oracle)
- ライセンスタイプ (license-included または bring-your-own-license)

新しい DB インスタンスの仕様がアカウント内の既存のリザーブド DB インスタンスと一致する場合は、リザーブド DB インスタンスに適用される割引料金で請求されます。一致しない場合、DB インスタンスはオンデマンド料金で請求されます。

リザーブド DB インスタンスとして使用している DB インスタンスを変更できます。変更がリザーブド DB インスタンスの仕様の範囲内である場合、割引の一部またはすべてが、変更された DB インスタンスに適用されます。インスタンスクラスの変更など、変更が仕様の範囲外である場合、割引は適用されません。詳細については、「サイズ柔軟なリザーブド DB インスタンス」を参照してください。

トピック

- 提供タイプ
- サイズ柔軟なリザーブド DB インスタンス
- リザーブド DB インスタンスの請求例
- マルチ AZ DB クラスターのリザーブド DB インスタンス
- リザーブド DB インスタンスの削除

リザーブド DB インスタンスの料金などの詳細については、Amazon RDS リザーブドインスタンスを参照してください。

提供タイプ

リザーブド DB インスタンスには、予想される使用量に基づいて Amazon RDS のコストを最適化するための 3 種類のオプション — 前払いなし、一部前払い、全前払い — があります。

> **Note**
>
> すべての RDS インスタンスクラスがすべてのリザーブドインスタンスオファリングタイプをサポートしているわけではありません。例えば、一部のインスタンスクラスには [前払いなし] オプションがない場合があります。可用性を確認するには、AWS マネジメントコンソールでリザーブドインスタンスのオファリングを確認するか、`describe-reserved-db-instances-offerings` AWS CLI コマンドを使用します。

前払いなし

このオプションは前払い料金なしでリザーブド DB インスタンスへのアクセスを提供します。前払いなしのリザーブド DB インスタンスでは、使用量にかかわらず、期間内の時間はすべて、割

引された時間料金で請求されます。前払い料金は必要ありません。このオプションは、1 年間の予約でのみ利用できます。

一部前払い

このオプションでは、リザーブド DB インスタンスの一部を前払いする必要があります。期間内の残りの時間は、使用量にかかわらず、割引された時間料金で請求されます。このオプションは、以前の "重度使用" オプションに代わるオプションです。

全前払い

期間のスタート時に全額を支払います。使用時間数に関係なく、残りの期間にそれ以外のコストは生じません。

一括請求を使用している場合、組織内のすべてのアカウントが 1 つのアカウントとして扱われます。これは、組織内のすべてのアカウントが、他のアカウントで購入したリザーブド DB インスタンスの時間単位のコスト利点を受けることができるということを意味しています。一括請求 (コンソリデーティッドビリング) の詳細については、AWS 請求情報とコスト管理ユーザーガイドの「Amazon RDS リザーブド DB インスタンス」を参照してください。

サイズ柔軟なリザーブド DB インスタンス

リザーブド DB インスタンスを購入する際、指定する項目の 1 つはインスタンスクラス (db.r5.large など) です。DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。

既存の DB インスタンスがあり、これをスケールして容量を増やす必要がある場合、リザーブド DB インスタンスはスケールした DB インスタンスに自動的に適用されます。つまり、リザーブド DB インスタンスは DB インスタンスクラスのすべてのサイズに自動的に適用されます。サイズに柔軟性のあるリザーブド DB インスタンスは、同じ AWS リージョン およびデータベースエンジンの DB インスタンスで利用できます。サイズ柔軟なリザーブド DB インスタンスは、そのインスタンスクラスタイプでしかスケールできません。例えば、db.r6i.large のリザーブド DB インスタンスは db.r6i.xlarge には適用できますが、db.r6id.large または db.r7g.large には適用できません。db.r6id.large と db.r7g.large は異なるインスタンスクラスタイプであるためです。

リザーブド DB インスタンスの利点はマルチ AZ およびシングル AZ の両設定に適用されます。これは、同じ DB インスタンスクラスタイプ内の設定間を自由に移動できることを意味します。例えば、1 つのラージ DB インスタンス (1 時間あたりの正規化された単位 4) で実行されているシングル AZ 配置から、2 つのミディアム DB インスタンス (1 時間あたりの正規化された単位 2+2=4) で実行されているマルチ AZ 配置に移行できます。

サイズ柔軟なリザーブド DB インスタンスは、以下の Amazon RDS データベースエンジンで使用できます。

- RDS for Db2
- RDS for MariaDB
- RDS for MySQL
- RDS for Oracle (Bring-Your-Own-License)
- RDS for PostgreSQL

サイズの柔軟性は、RDS for SQL Server および RDS for Oracle ライセンス込みには適用されません。

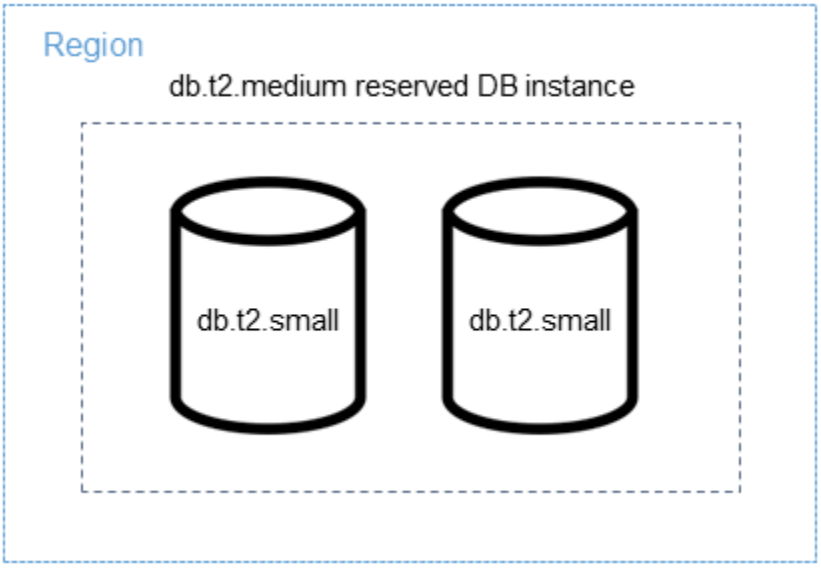
Aurora でサイズ柔軟なリザーブドインスタンスを使用する方法の詳細については、「Aurora のリザーブド DB インスタンス」を参照してください。

リザーブド DB インスタンスのサイズ別の使用は、1 時間あたりの正規化された単位を使用して比較できます。例えば、2 つの db.r3.large DB インスタンスでの 1 単位の使用は、1 つの db.r3.small での 1 時間あたりの正規化された単位 8 の使用に相当します。次の表は、DB インスタンスのサイズ別の 1 時間あたりの正規化された単位の数を示しています。

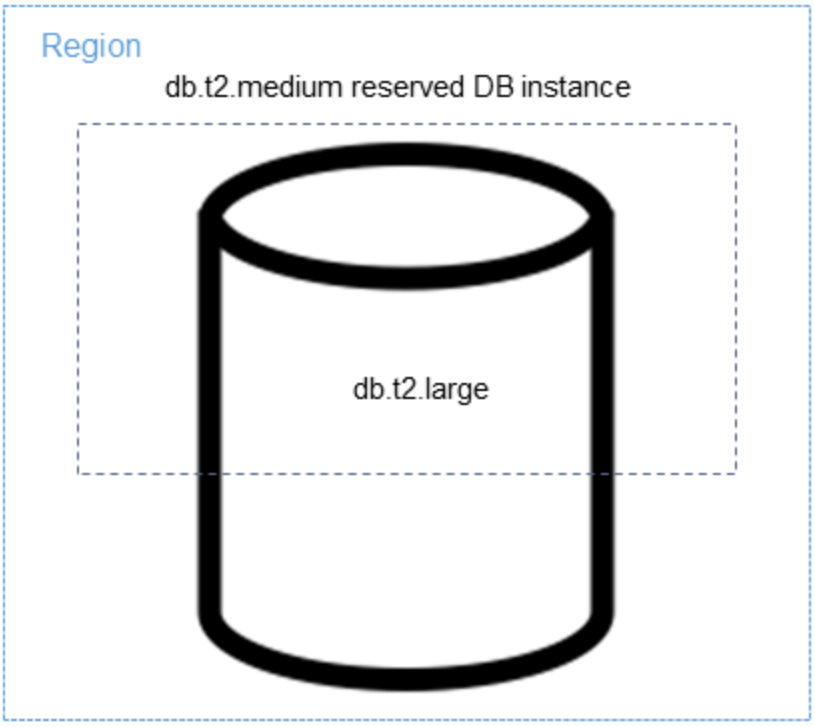
インスタンスサイズ	シングル AZ 1 時間あたりの正規化された単位 (1 つの DB インスタンスでのデプロイ)	マルチ AZ DB インスタンス 1 時間あたりの正規化された単位 (1 つの DB インスタンスと 1 つのスタンバイによるデプロイ)	マルチ AZ DB クラスター 1 時間あたりの正規化された単位 (1 つの DB インスタンスと 2 つのスタンバイによるデプロイ)
micro	0.5	1	1.5
small	1	2	3
medium	2	4	6
large	4	8	12
xlarge	8	16	24
2xlarge	16	32	48

インスタンスサイズ	シングル AZ 1 時間あたりの正規化された単位 (1 つの DB インスタンスでのデプロイ)	マルチ AZ DB インスタンス 1 時間あたりの正規化された単位 (1 つの DB インスタンスと 1 つのスタンバイによるデプロイ)	マルチ AZ DB クラスター 1 時間あたりの正規化された単位 (1 つの DB インスタンスと 2 つのスタンバイによるデプロイ)
4xlarge	32	64	96
6xlarge	48	96	144
8xlarge	64	128	192
10xlarge	80	160	240
12xlarge	96	192	288
16xlarge	128	256	384
24xlarge	192	384	576
32xlarge	256	512	768

例えば、db.t2.medium リザーブド DB インスタンスを購入し、同じ AWS リージョン のアカウントで 2 つの db.t2.small DB インスタンスを実行しているとします。この場合、料金上の利点は両方のインスタンスに全面的に適用されます。



また、同じ AWS リージョン のアカウントで 1 つの `db.t2.large` インスタンスを実行している場合、この DB インスタンスの使用の 50 パーセント に支払い特典が適用されます。



リザーブド DB インスタンスの請求例

リザーブド DB インスタンスの価格には、ストレージ、バックアップ、I/O に関連するコストの割引はありません。 時間単位のオンデマンド インスタンスの使用に対してのみ割引が提供されます。次の例では、リザーブド DB インスタンスの 1 か月当たりの総コストを示します。

- RDS for MySQL リザーブドシングル AZ db.r5.large DB インスタンスクラスのコストは、米国東部 (バージニア北部) の場合、前払いなしでインスタンスに 0.12 USD、または 1 か月あたり 90 USD
- 汎用 SSD (gp2) ストレージの 400 GiB のコストは、1 か月 1 GiB あたり 0.115 USD、または 1 か月あたり 45.60 USD
- バックアップストレージの 600 GiB のコストは、0.095 USD、または 1 か月あたり 19 USD (400 GiB は無料)

リザーブド DB インスタンスにこれらすべての料金 (90 USD + 45.60 USD + 19 USD) を加えると、1 か月あたりの総コストは 154.60 USD です。

リザーブド DB インスタンスの代わりにオンデマンド DB インスタンスを使用する場合、RDS for MySQL シングル AZ db.r5.large DB インスタンスクラスのコストは米国東部 (バージニア北部) では、1 時間あたり 0.1386 USD、または 1 か月あたり 101.18 USD です。つまり、オンデマンド DB インスタンスの場合、これらすべてのオプション (101.18 USD + 45.60 USD + 19 USD) が加わり、1 か月あたりの総コストは 165.78 USD となります。リザーブド DB インスタンスを使用すると、月々 11 ドル強の節約になります。

> **Note**
>
> この例で説明しているのはサンプルの価格であり、実際の価格とは一致しない場合があります。Amazon RDS の料金情報については、「Amazon RDS の料金」を参照してください。

マルチ AZ DB クラスターのリザーブド DB インスタンス

マルチ AZ DB クラスターと同等のリザーブド DB インスタンスを購入するには、次のいずれかを実行します。

- クラスター内のインスタンスと同じサイズのシングル AZ DB インスタンスを 3 つ予約します。
- クラスター内の DB インスタンスと同じサイズのマルチ AZ DB インスタンス 1 つとシングル AZ DB インスタンスを 1 つ予約します。

例えば、1 個のクラスターが 3 つの db.m6gd.large DB インスタンスで構成されるとします。この場合、db.m6gd.large シングル AZ リザーブド DB インスタンスを 3 つ購入するか、db.m6gd.large マルチ AZ リザーブド DB インスタンスを 1 つと db.m6gd.large シングル AZ リザーブド DB インスタンスを 1 つ購入することができます。どちらのオプションでも、マルチ AZ DB クラスターのリザーブドインスタンスの最大割引を得られます。

あるいは、サイズに関して柔軟な DB インスタンスを使用し、大きな DB インスタンスを購入して、1 つ以上のクラスター内で小さな DB インスタンスをカバーすることもできます。合計 6 つの db.m6gd.large DB インスタンスを持つ 2 つのクラスターがある場合は、db.m6gd.xl シングル AZ リザーブド DB インスタンスを 3 つ購入できます。これにより、2 つのクラスター内の 6 つの DB インスタンスすべてが予約されます。詳細については、「サイズ柔軟なリザーブド DB インスタンス」を参照してください。

クラスター内の DB インスタンスと同じサイズの DB インスタンスを予約する場合、クラスター内の DB インスタンスの合計数よりも少ない数の DB インスタンスが予約されます。ただし、その場合はクラスターが部分的にしか予約されません。例えば、1 つのクラスターに 3 つの db.m6gd.large DB インスタンスがあり、db.m6gd.large マルチ AZ リザーブド DB インスタンスを 1 つ購入したとします。この場合、クラスター内の 3 つのインスタンスのうち 2 つのみがリザーブド DB インスタンスの対象となっているため、クラスターは部分的にしか予約されません。残りの DB インスタンスは、オンデマンドの db.m6gd.large 時間料金で請求されます。

マルチ AZ DB クラスターの詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。

リザーブド DB インスタンスの削除

リザーブド DB インスタンスには 1 年契約と 3 年契約があります。リザーブド DB インスタンスをキャンセルすることはできません。ただし、リザーブド DB インスタンスの割引対象である DB インスタンスは削除できます。リザーブド DB インスタンスの割引対象である DB インスタンスの削除プロセスは、他の DB インスタンスの削除プロセスと同じです。

リソースを使用するかどうかにかかわらず、前払いコストが請求されます。

リザーブド DB インスタンスの割引対象である DB インスタンスを削除した場合、互換性がある仕様の別の DB インスタンスを起動できます。この場合、予約期間 (1 年または 3 年) 中、割引料金を利用できます。

Amazon RDS のリザーブド DB インスタンスの購入

AWS マネジメントコンソール、AWS CLI、および RDS API を使用して、リザーブド DB インスタンスを使用できます。

コンソール

リザーブド DB インスタンスを AWS マネジメントコンソール で使用するには、次の手順に従います。

リザーブド DB インスタンス提供タイプの料金表と情報を取得するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[リザーブドインスタンス] を選択します。
3. [Purchase Reserved DB Instance] を選択します。
4. [製品の説明] で、DB エンジンとライセンスタイプを選択します。
5. [DB インスタンスクラス] で、DB インスタンスのクラスを選択します。
6. [デプロイオプション] で、シングル AZ または マルチ AZ DB インスタンスのデプロイが必要かどうかを選択します。

   > **Note**
   >
   > マルチ AZ DB クラスターのデプロイ用に同等のリザーブド DB インスタンスを購入するには、シングル AZ リザーブド DB インスタンスを 3 つ購入するか、マルチ AZ リザーブド DB インスタンスを 1 つと シングル AZ リザーブド DB インスタンスを 1 つ購入します。詳細については、「マルチ AZ DB クラスターのリザーブド DB インスタンス」を参照してください。

7. [期間] で、DB インスタンスを予約する期間を選択します。
8. [提供タイプ] で、提供タイプを選択します。

   提供タイプを選択すると、料金情報が表示されます。

   > **Important**
   >
   > リザーブド DB インスタンスの購入と料金の発生を防ぐには、[キャンセル] を選択します。

リザーブド DB インスタンス提供タイプに関する情報を取得したら、次の手順に従い、この情報を使用して提供タイプを購入できます。

リザーブド DB インスタンスを購入するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[リザーブドインスタンス] を選択します。
3.
> **Important**
>
> 先に進む前に、正しい AWS リージョンを使用していることを確認してください。リザーブド DB インスタンスはリージョン固有であり、リージョン間で転送することはできません。コンソールにあるリージョンセレクターを使用して、目的のリージョンでリザーブドインスタンスを購入していることを確認してください。

4. [Purchase Reserved DB Instance] (リザーブド DB インスタンスの購入) を選択します。
5. [製品の説明] で、DB エンジンとライセンスタイプを選択します。
6. [DB インスタンスクラス] で、DB インスタンスのクラスを選択します。
7. [マルチ AZ 配置] で、シングル AZ またはマルチ AZ DB インスタンス配置が必要かどうかを選択します。

> **Note**
>
> マルチ AZ DB クラスターのデプロイ用に同等のリザーブド DB インスタンスを購入するには、シングル AZ リザーブド DB インスタンスを 3 つ購入するか、マルチ AZ リザーブド DB インスタンスを 1 つと シングル AZ リザーブド DB インスタンスを 1 つ購入します。詳細については、「マルチ AZ DB クラスターのリザーブド DB インスタンス」を参照してください。

8. [Term] で、DB インスタンスを予約する期間を選択します。
9. [提供タイプ] で、提供タイプを選択します。

   提供タイプを選択すると、料金情報が表示されます。
   [リザーブド DB インスタンスを購入] コンソール。
10. (オプション) 購入したリザーブド DB インスタンスに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。[Reserved Id] に、リザーブド DB インスタンスの識別子を入力します。
11. [Submit] を選択してください。

リザーブド DB インスタンスを購入すると、リザーブドインスタンス リストに表示されます。

リザーブド DB インスタンスを購入したら、次の手順に従ってリザーブド DB インスタンスに関する情報を取得できます。

AWS アカウントのリザーブド DB インスタンスの情報を入手するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [ナビゲーション] ペインで [リザーブドインスタンス] を選択します。

   アカウントのリザーブド DB インスタンスが表示されます。特定のリザーブド DB インスタンスに関する詳細な情報を確認するには、リストにあるそのインスタンスを選択します。これにより、詳細ペインで、そのインスタンスに関する詳細情報を表示できます。

AWS CLI

リザーブド DB インスタンスを AWS CLI で使用するには、以下の例に従います。

Example使用可能なリザーブド DB インスタンスの提供タイプに関する情報を入手する

使用可能なリザーブド DB インスタンス提供タイプに関する情報を取得するには、AWS CLI コマンド `describe-reserved-db-instances-offerings` を呼び出します。

```
aws rds describe-reserved-db-instances-offerings
```

この呼び出しにより、以下のような出力が返されます。

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
```

```
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

リザーブド DB インスタンス提供タイプに関する情報を取得したら、この情報を使用して提供タイプを購入できます。

リザーブド DB インスタンスを購入するには、以下のパラメータを指定して AWS CLI コマンド [purchase-reserved-db-instances-offering] を呼び出します。

- `--reserved-db-instances-offering-id` - 購入する提供タイプの ID。提供タイプの ID を取得するには、前の例を参照してください。
- `--reserved-db-instance-id` - 購入したリザーブド DB インスタンスに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。

Exampleリザーブド DB インスタンスを購入する

次の例では、ID が *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f* のリザーブド DB インスタンスを購入し、識別子として *MyReservation* を割り当てます。

Linux、macOS、Unix の場合:

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Windows の場合:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

このコマンドにより、以下のような出力が返されます。

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
```

```
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

リザーブド DB インスタンスを購入したら、リザーブド DB インスタンスに関する情報を取得できます。

AWS アカウントのリザーブド DB インスタンスに関する情報を取得するには、以下の例に従って、AWS CLI コマンド describe-reserved-db-instances を呼び出します。

Exampleリザーブド DB インスタンスを取得する

```
aws rds describe-reserved-db-instances
```

このコマンドにより、以下のような出力が返されます。

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

RDS API

RDS API を使用して、リザーブド DB インスタンスを操作できます。

- 使用可能なリザーブド DB インスタンス提供タイプに関する情報を取得するには、Amazon RDS API オペレーション DescribeReservedDBInstancesOfferings を呼び出します。
- リザーブド DB インスタンス提供タイプに関する情報を取得したら、この情報を使用して提供タイプを購入できます。次のパラメータを指定して、PurchaseReservedDBInstancesOffering RDS API オペレーションを実行します。
  - `--reserved-db-instances-offering-id` - 購入する提供タイプの ID。
  - `--reserved-db-instance-id` - 購入したリザーブド DB インスタンスに独自の識別子を割り当てると、インスタンスを追跡しやすくなります。
- リザーブド DB インスタンスを購入したら、リザーブド DB インスタンスに関する情報を取得できます。DescribeReservedDBInstances RDS API オペレーション を呼び出します。

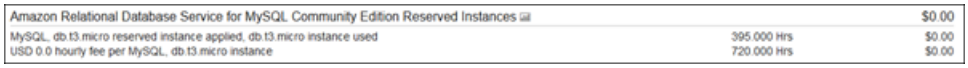
Amazon RDS のリザーブド DB インスタンスの請求の表示

リザーブド DB インスタンスの請求は、AWS マネジメントコンソールの「請求ダッシュボード」 で表示できます。

リザーブド DB インスタンスの請求を表示する

1. AWS マネジメントコンソール にサインインします。
2. [アカウント] メニューで、[請求ダッシュボード] を選択します。
3. [料金明細] を選択します。
4. 「AWSサービス料金」で、リレーショナルデータベースサービスを展開します。
5. リザーブド DB インスタンスが置かれている AWS リージョン、例えば米国西部 (オレゴン) を展開します。

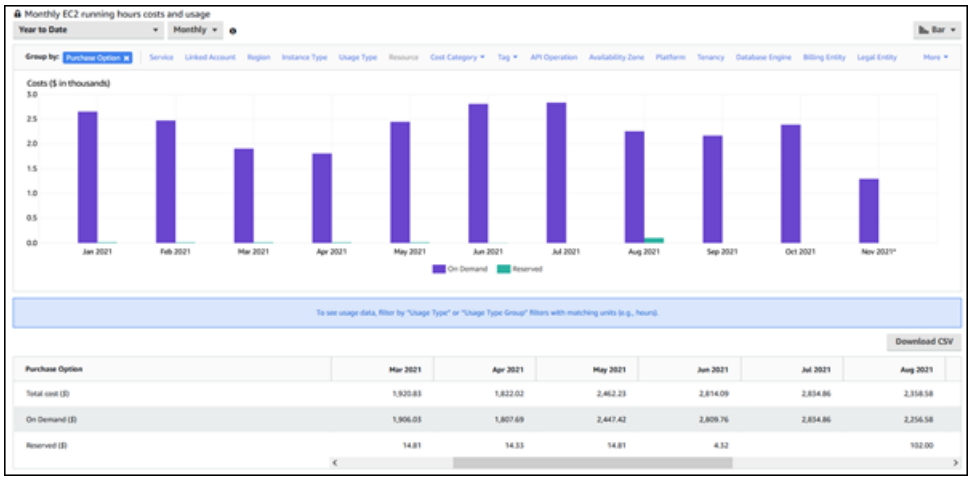
   リザーブド DB インスタンスと当月の時間単位の料金は、Amazon Relational Database Service ##########リザーブドインスタンスに表示されます。

   この例では、リザーブド DB インスタンスは全額前払いで購入したため、時間単位の料金は発生しません。
6. リザーブドインスタンスの見出し横にある「Cost Explorer」(棒グラフ) アイコンを選択します。

   Cost Explorer には、毎月の EC2 稼働時間のコストと使用状況グラフが表示されます。
7. グラフ右横の使用タイプグループフィルターをクリアします。
8. 使用コストを調べる期間と時間単位を選択します。

   次の例は、オンデマンドおよびリザーブド DB インスタンスの年間使用コストを月単位で表示しています。

   2021 年 1 月から 6 月までのリザーブド DB インスタンスの料金は、一部前払いのインスタンスの月額料金で、2021 年 8 月の料金は全額前払いのインスタンスの 1 回限りの料金です。

一部前払いのインスタンスのリザーブドインスタンス割引は 2021 年 6 月に有効期限が切れましたが、DB インスタンスは削除されませんでした。有効期限を過ぎてからは、オンデマンド料金で請求されました。

Amazon RDS 環境のセットアップ

このページでは、アカウント設定、セキュリティ、リソース管理など、Amazon Relational Database Service をセットアップするための包括的なガイドを提供します。データベース環境を効率的に作成、管理、保護するための重要なステップについて説明します。Amazon RDS を初めて使用する場合も、特定の要件に合わせて設定する場合も、これらのセクションを参照して、セットアップが最適化されており、ベストプラクティスに準拠していることを確認できます。

トピック

- AWS アカウント へのサインアップ
- 要件の確認
- セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する

既に AWS アカウント があり、Amazon RDS の要件を理解していて、IAM と VPC セキュリティグループでデフォルト設定を使用する場合は、Amazon RDS のスタート方法 にスキップできます。

AWS アカウント へのサインアップ

AWS を使い始めるには、AWS アカウントが必要です。AWS アカウントの作成の詳細については、「AWS アカウント管理 リファレンスガイド」の「AWS アカウントの開始方法」を参照してください。

要件の確認

Amazon RDS の基本的な構成要素は DB インスタンスです。DB インスタンスでは、データベースを作成します。DB インスタンスは、エンドポイントというネットワークアドレスを提供します。アプリケーションでは、このエンドポイントを使用して DB インスタンスに接続します。DB インスタンスの作成時に、ストレージ、メモリ、データベースエンジンとバージョン、ネットワーク構成、セキュリティ、メンテナンス期間などの詳細を指定します。DB インスタンスへのネットワークアクセスは、セキュリティグループを通じて制御します。

DB インスタンスとセキュリティグループを作成する前に、DB インスタンスとネットワークに関する要件を理解しておく必要があります。重要な留意事項を以下に示します。

- リソース要件 – アプリケーションまたはサービスに関するメモリとプロセッサの要件。これらの設定を使用すると、使用する DB インスタンスクラスを判断するのに役立ちます。DB インスタンスクラスの仕様については、「DB インスタンスクラス」を参照してください。
- VPC、サブネット、セキュリティグループ – DB インスタンスは Virtual Private Cloud (VPC) である可能性が最も高くなります。DB インスタンスに接続するには、セキュリティグループルールを設定する必要があります。これらのルールは、使用する VPC の種類と使用方法に応じて設定が異なります。例えば、デフォルト VPC またはユーザー定義の VPC を使用できます。

  各 VPC オプションに関するルールを次に説明します。

  - デフォルト VPC – AWS アカウントのデフォルト VPC が現在の AWS リージョンにある場合、その VPC は DB インスタンスをサポートするように設定されます。DB インスタンスの作成時にデフォルト VPC を指定する場合は、次の操作を行います。
    - アプリケーションやサービスから Amazon RDS DB インスタンスへの接続を許可する VPC セキュリティグループを必ず作成します。VPC セキュリティグループを作成するには、VPC コンソールの [Security Group] (セキュリティグループ) オプションまたは AWS CLI を使用します。詳細については、「ステップ 3: VPC セキュリティグループを作成する」を参照してください。
    - デフォルトの DB サブネットグループを指定します。この AWS リージョンで DB インスタンスを初めて作成した場合、DB インスタンスの作成時にデフォルトの DB サブネットグループが、Amazon RDS により作成されます。
  - ユーザー定義の VPC – DB インスタンスの作成時にユーザー定義の VPC を指定する場合は、以下の点に注意します。
    - アプリケーションやサービスから Amazon RDS DB インスタンスへの接続を許可する VPC セキュリティグループを必ず作成します。VPC セキュリティグループを作成するには、VPC コンソールの [Security Group] (セキュリティグループ) オプションまたは AWS CLI を使用します。詳細については、「ステップ 3: VPC セキュリティグループを作成する」を参照してください。
    - DB インスタンスをホストするには、VPC は特定の要件 (2 つ以上のサブネットを保持しており、各サブネットは個別のアベイラビリティーゾーン内にあることなど) を満たす必要があります。詳細については、「Amazon VPC と Amazon RDS」を参照してください。
    - DB インスタンスで使用できる VPC 内のサブネットを定義する DB サブネットグループを必ず指定します。詳細については、「VPC 内の DB インスタンスの使用」の「DB サブネットグループの使用」セクションを参照してください。
- 高可用性 – フェイルオーバーサポートが必要かどうか。Amazon RDS では、マルチ AZ 配置により、フェイルオーバーのサポート用に、プライマリ DB インスタンスとセカンダリスタンバイ DB

インスタンスが別個のアベイラビリティーゾーンに作成されます。本番稼働用のワークロードには、高可用性を維持するためにマルチ AZ 配置をお勧めします。開発およびテストの目的では、マルチ AZ 配置以外のデプロイを使用できます。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

- IAM ポリシー – DAmazon RDS オペレーションの実行に必要なアクセス許可を付与するためのポリシーが、自分の AWS アカウントにあるかどうか。IAM 認証情報を使用して AWS に接続している場合、IAM アカウントには、Amazon RDS オペレーションの実行するためのアクセス許可を付与する IAM ポリシーが必要です。詳細については、「Amazon RDS での Identity and Access Management」を参照してください。
- 開いているポート – データベースでリッスンする TCP/IP ポート。一部の企業のファイアウォールでは、データベースエンジン用のデフォルトポートへの接続がブロックされる場合があります。企業のファイアウォールがデフォルトのポートをブロックしている場合は、新しい DB インスタンス用に他のポートを選択します。指定したポートでリッスンする DB インスタンスを作成した場合、DB インスタンスを変更することでポートを変更できます。
- AWS リージョン – データベースが必要となる AWS リージョン。アプリケーションやウェブサービスの近くにデータベースを配置すると、ネットワークレイテンシーを低減できます。詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。
- DB ディスクサブシステム – ストレージ要件について検討します。Amazon RDS には、次の 2 つのストレージタイプがあります。
  - 汎用 (SSD)
  - プロビジョンド IOPS (PIOPS)

  マグネティックストレージ (標準ストレージとも呼ばれます) は廃止され、新しい DB インスタンスでは提供されなくなります。Amazon RDS ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

セキュリティグループと DB インスタンスの作成に必要な情報を把握したら、次のステップに進みます。

セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する

VPC セキュリティグループは、VPC 内の DB インスタンスへのアクセスを提供します。セキュリティグループは、関連付けられた DB インスタンスのファイアウォールとして動作し、インバウンドトラフィックとアウトバウンドトラフィックの両方を DB インスタンスレベルで制御します。DB イ

ンスタンスはデフォルトでファイアウォールによって作成され、DB インスタンスを保護するデフォルトのセキュリティグループとなります。

DB インスタンスに接続する前に、接続を可能にするルールをセキュリティグループに追加する必要があります。ネットワークと設定に関する情報を使用して、DB インスタンスへのアクセスを許可するルールを作成します。

例えば、アプリケーションから VPC 内にある DB インスタンスのデータベースにアクセスするとします。この場合、カスタム TCP ルールを追加し、アプリケーションからデータベースにアクセスするためのポート範囲と IP アドレスを指定する必要があります。アプリケーションが Amazon EC2 インスタンスにある場合は、Amazon EC2 インスタンスに設定したセキュリティグループを使用できます。

DB インスタンスを作成するときに、Amazon EC2 インスタンスと DB インスタンス間の接続を設定できます。(詳しくは、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。)

> **Tip**
>
> DB インスタンスを作成するときに、Amazon EC2 インスタンスと DB インスタンス間のネットワーク接続を自動的に設定できます。詳細については、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。

Amazon Lightsail のリソースを DB インスタンスに接続する方法については、「 Connect Lightsail resources to AWS のサービス using VPC peering」を参照してください。

DB インスタンスにアクセスするための一般的なシナリオについては、VPC の DB インスタンスにアクセスするシナリオ を参照してください。

VPC セキュリティグループを作成するには

1. AWS マネジメントコンソール にサインインして、Amazon VPC コンソール (https://console.aws.amazon.com/vpc) を開きます。

   > **Note**
   >
   > RDS コンソールではなく VPC コンソールにアクセスしていることを確認します。

2. AWS マネジメントコンソールで、VPC セキュリティグループと DB インスタンスを作成する AWS リージョンを選択します。この AWS リージョンにある Amazon VPC リソースのリストには、少なくとも 1 の VPC と複数のサブネットが表示されます。表示されない場合、この AWS リージョンにはデフォルト VPC がありません。
3. ナビゲーションペインで、[Security Groups] を選択します。
4. [セキュリティグループの作成] を選択します。

   [Create security group] (セキュリティグループの作成) ページが表示されます。
5. [Basic details] (基本的な詳細) で、[Security group name] (セキュリティグループ名) と [Description] (説明) を入力します。[VPC] で、DB インスタンスを作成する先の VPC を選択します。
6. [Inbound rules] (インバウンドルール) で、[Add rule] (ルールを追加) を選択します。

   a. [タイプ] で [カスタム TCP] を選択します。
   b. [Port range] (ポート範囲) で、DB インスタンスに使用するポート値を入力します。
   c. [Source] (ソース) で、セキュリティグループ名を選択するか、DB インスタンスにアクセスする IP アドレスの範囲 (CIDR 値) を入力します。[マイ IP] を選択すると、ブラウザで検出された IP アドレスから DB インスタンスにアクセスできます。
7. IP アドレスや異なるポート範囲を追加する必要がある場合は、[Add rule] (ルールを追加) を選択し、ルールの情報を入力します。
8. (オプション) [Outbound rules] (アウトバウンドルール) で、アウトバウンドトラフィックのルールを追加します。デフォルトではすべてのアウトバウンドトラフィックが許可されます。
9. [セキュリティグループの作成] を選択します。

ここで作成した VPC セキュリティグループは、DB インスタンスの作成時にセキュリティグループとして使用できます。

> **Note**
>
> デフォルトの VPC を使用する場合、すべての VPC のサブネットにまたがるデフォルトのサブネットグループが作成されます。DB インスタンスを作成する場合は、デフォルト VPC を選択し、[DB サブネットグループ] の [デフォルト] を使用できます。

セットアップに必要なステップが完了したら、ユーザーの要件とセキュリティグループを利用して、DB インスタンスを作成できます。これを行うには、「Amazon RDS DB インスタンスの作成」

の手順に従います。特定の DB エンジンを使用する DB インスタンスを作成して使用を開始する方法については、次の表にある関連ドキュメントを参照してください。

データベースエンジン	ドキュメント
MariaDB	MariaDB DB インスタンスの作成と接続
Microsoft SQL Server	Microsoft SQL Server DB インスタンスを作成して接続する
MySQL	MySQL DB インスタンスの作成と接続
Oracle	Oracle DB インスタンスを作成して接続する
PostgreSQL	PostgreSQL DB インスタンスを作成して接続する

> **Note**
>
> 作成後に DB インスタンスに接続できない場合は、「Amazon RDS DB インスタンスに接続できない」のトラブルシューティング情報を参照してください。

Amazon RDS のスタート方法

以下の例では、Amazon Relational Database Service (Amazon RDS) を使用して DB インスタンスを作成および接続する方法について説明します。Db2、MariaDB、MySQL、Microsoft SQL Server、Oracle、または PostgreSQL を使用する DB インスタンスを作成できます。

> **Important**
>
> DB インスタンスを作成したり、DB インスタンスに接続したりする前に、必ず Amazon RDS 環境のセットアップ のタスクを完了してください。

DB インスタンスの作成と DB インスタンスでのデータベースへの接続は、DB エンジンごとに若干異なります。DB インスタンスの作成と接続に関する詳細を確認するために使用する DB エンジンを次のいずれかから 1 つ選択します。DB インスタンスを作成して接続した後、DB インスタンスを削除する手順も示されています。

トピック

- MariaDB DB インスタンスの作成と接続
- Microsoft SQL Server DB インスタンスを作成して接続する
- MySQL DB インスタンスの作成と接続
- Oracle DB インスタンスを作成して接続する
- PostgreSQL DB インスタンスを作成して接続する
- チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する
- チュートリアル: Lambda 関数を使用して Amazon RDS にアクセスする

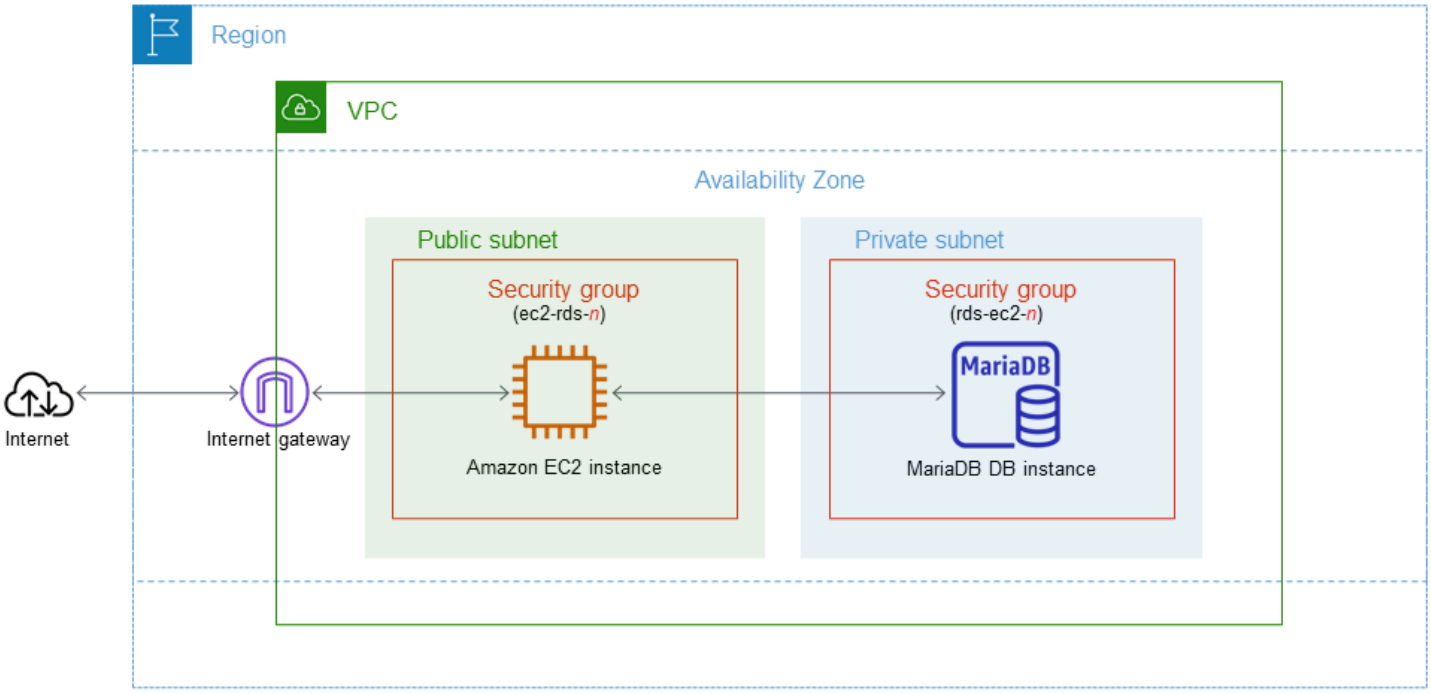
MariaDB DB インスタンスの作成と接続

このチュートリアルでは、EC2 インスタンスと RDS for MariaDB DB インスタンスを作成します。このチュートリアルでは、標準の MySQL クライアントを使用して EC2 インスタンスから DB インスタンスにアクセスする方法を示します。ベストプラクティスとして、このチュートリアルでは、プライベート DB インスタンスを仮想プライベートクラウド (VPC) に作成します。ほとんどの場合、EC2 インスタンスなど、同じ VPC 内の他のリソースは DB インスタンスにアクセスできますが、VPC 外部のリソースはアクセスできません。

チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。1 つのアベイラビリティーゾーンで、EC2 インスタンスはパブリックサブネットにあり、DB インスタンスはプライベートサブネットにあります。

> **Important**
>
> AWS アカウント を作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

次の図は、チュートリアルが完了した時点の設定を示しています。



このチュートリアルでは、次のいずれかの方法を使用してリソースを作成できます。

1. AWS マネジメントコンソール を使用する -「EC2 インスタンスの作成」と「MariaDB DB インスタンスを作成する」
2. CloudFormation を使用してデータベースインスタンスと EC2 インスタンスを作成する - (オプション) CloudFormation を使用して VPC、EC2 インスタンス、MariaDB インスタンスを作成する

最初の方法では、[簡単作成] を使用して、AWS マネジメントコンソール でプライベート MariaDB DB インスタンスを作成します。ここでは、DB エンジンタイプ、DB インスタンスサイズ、DB インスタンス識別子のみを指定します。[Easy create (簡易作成)] では、他の設定オプションのデフォルト設定を使用します。

代わりに [標準作成] を使用する場合は、DB インスタンスの作成時にさらに多くの設定オプションを指定できます。このようなオプションには、可用性、セキュリティ、バックアップ、メンテナンスの設定があります。パブリック DB インスタンスを作成するには、[標準作成] を使用する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

トピック

- 前提条件
- EC2 インスタンスの作成
- MariaDB DB インスタンスを作成する
- (オプション) CloudFormation を使用して VPC、EC2 インスタンス、MariaDB インスタンスを作成する
- MariaDB DB インスタンスに接続する
- EC2 インスタンスと DB インスタンスを削除する
- (オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する
- (オプション) DB インスタンスを Lambda 関数に接続する

前提条件

開始する前に、以下のセクションのステップを完了してください。

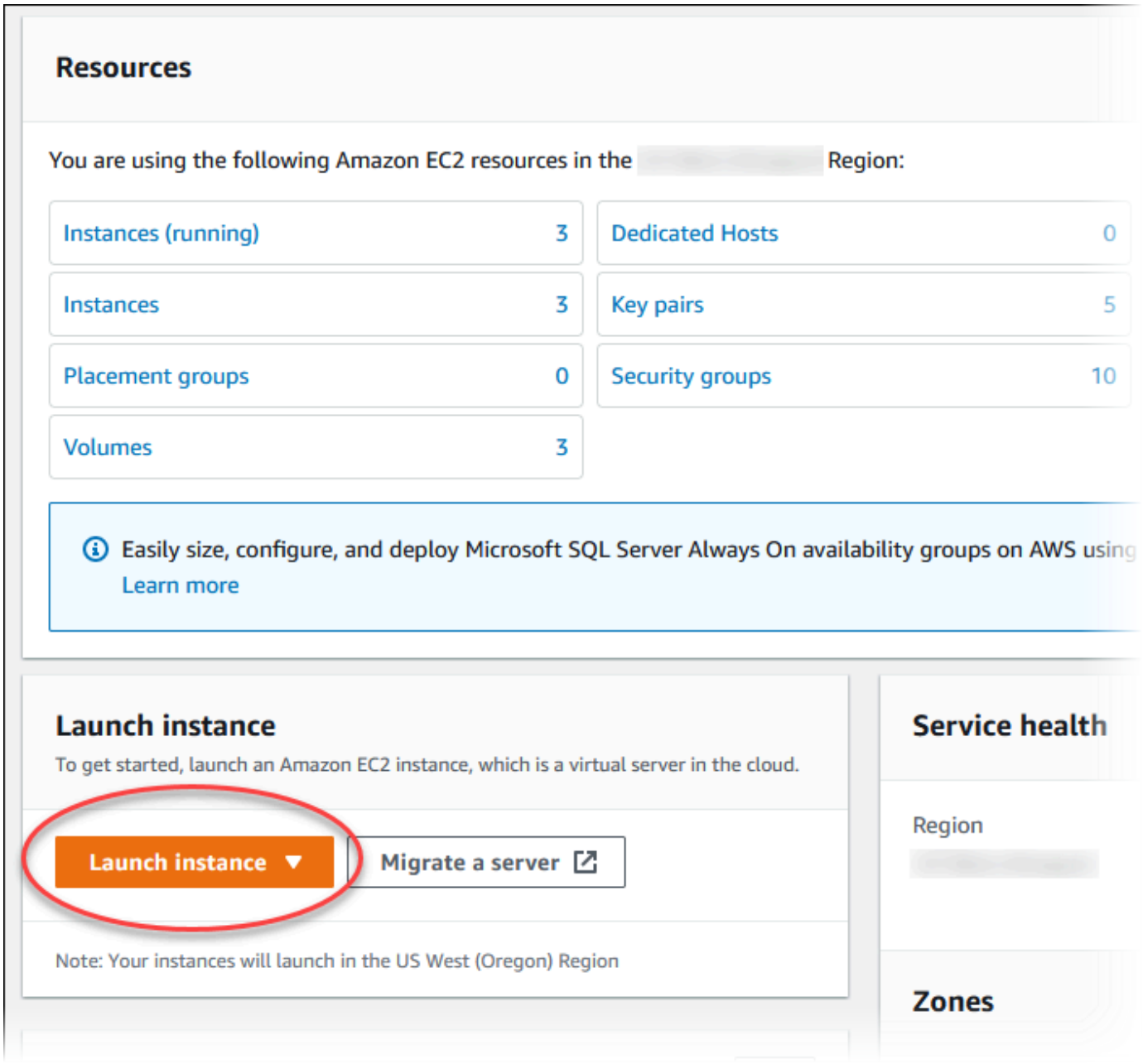
- AWS アカウント へのサインアップ

EC2 インスタンスの作成

データベースへの接続に使用する Amazon EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、EC2 インスタンスを作成する AWS リージョンを選択します。
3. 次の図に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの起動] を選択します。

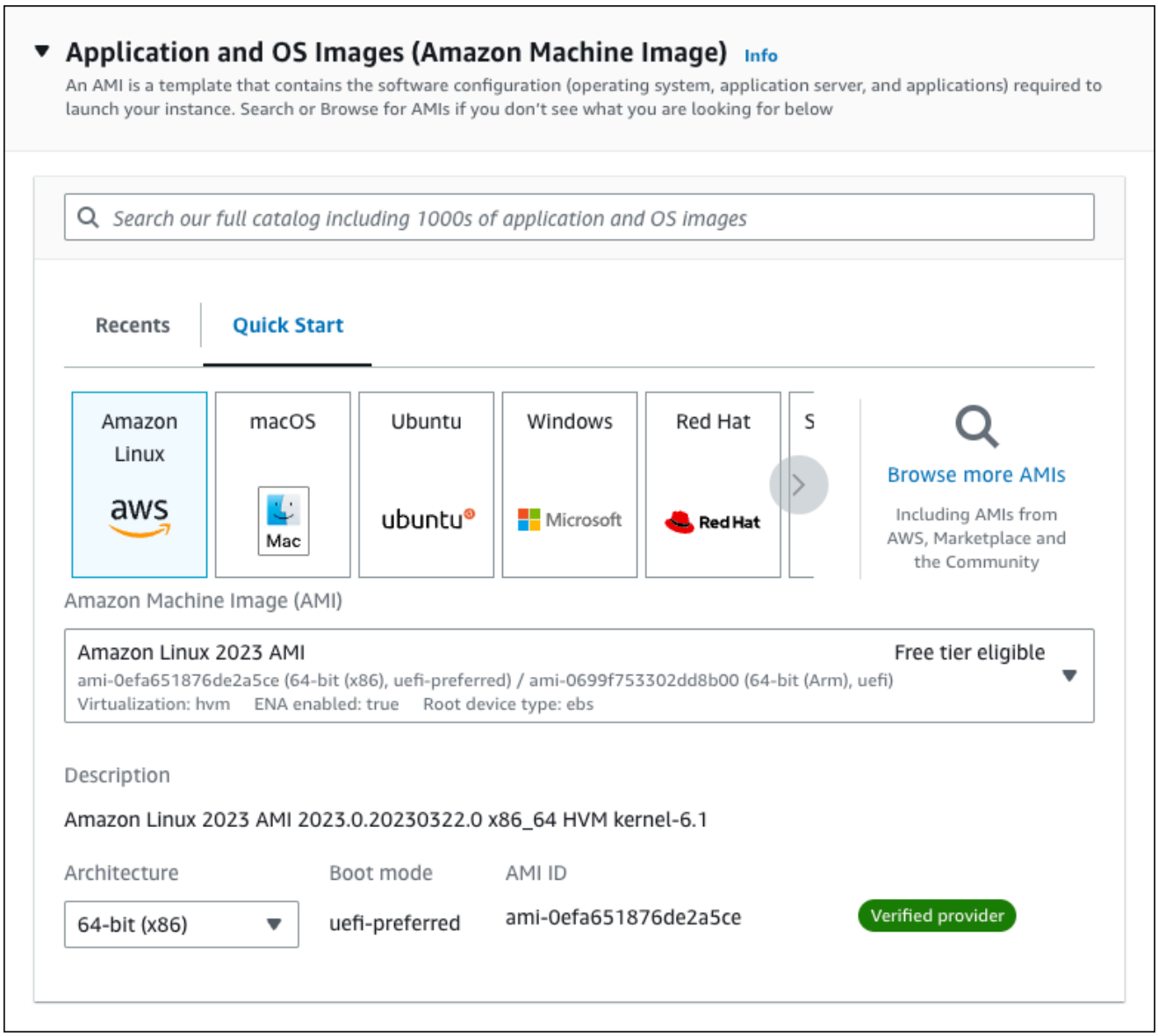


[インスタンスを起動] ページが開きます。

4. [インスタンスを起動] ページで次の設定を選択します。

   a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`ec2-database-connect`** と入力します。

   b. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[Amazon Linux] を選択し、[Amazon Linux 2023 AMI] を選択します。他の選択肢は、デフォルトの選択のままにします。



   c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

   d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するに

は、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

新しいキーペアの作成については、「Amazon EC2 ユーザーガイド」の「キーペアの作成」を参照してください。

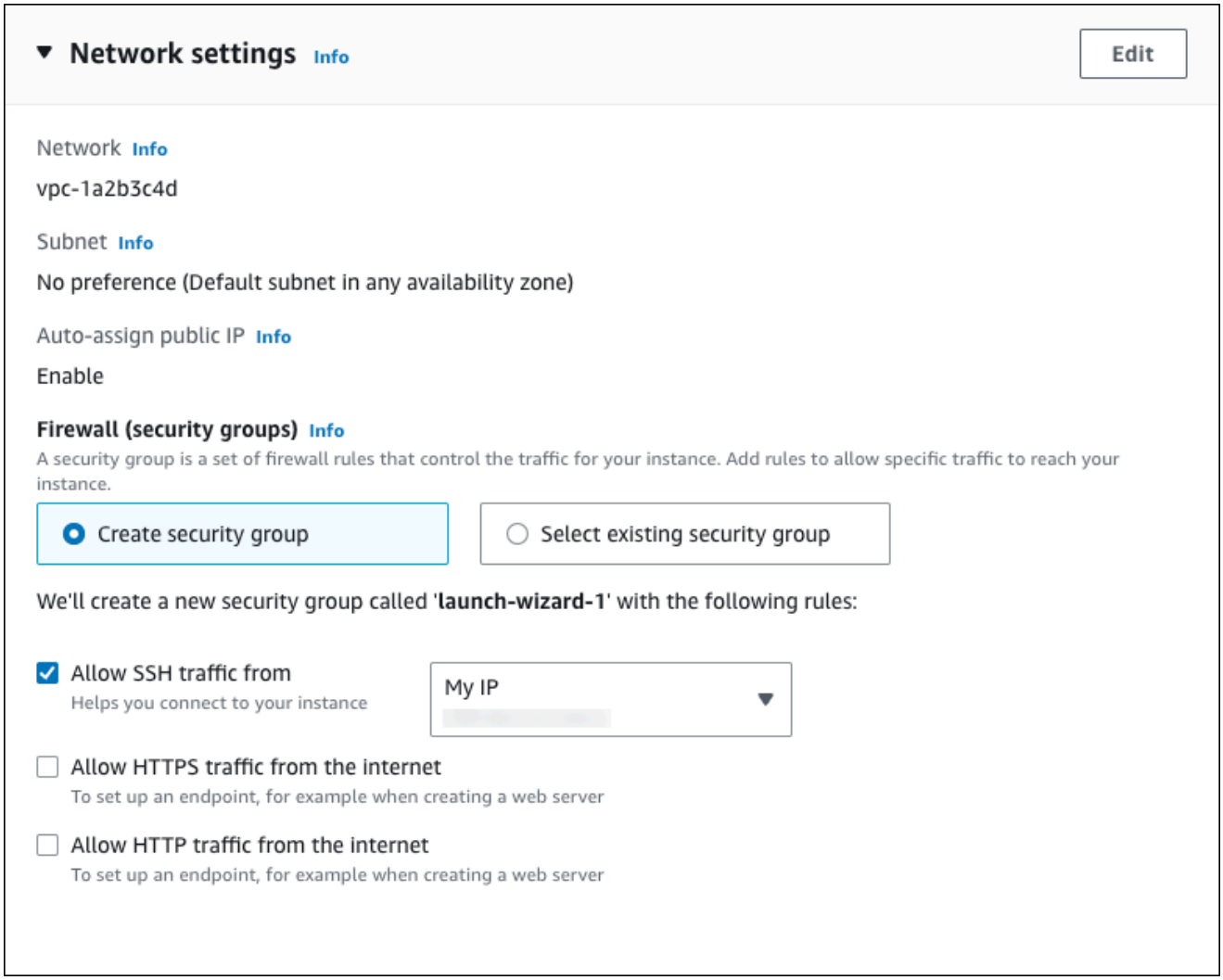
e. ネットワーク設定の [SSH トラフィックを許可] で、EC2 インスタンスへの SSH 接続のソースを選択します。

表示された IP アドレスが SSH 接続に適している場合は、[My IP] (マイ IP)を選択できます。それ以外の場合は、Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は 192.0.2.1/32 です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

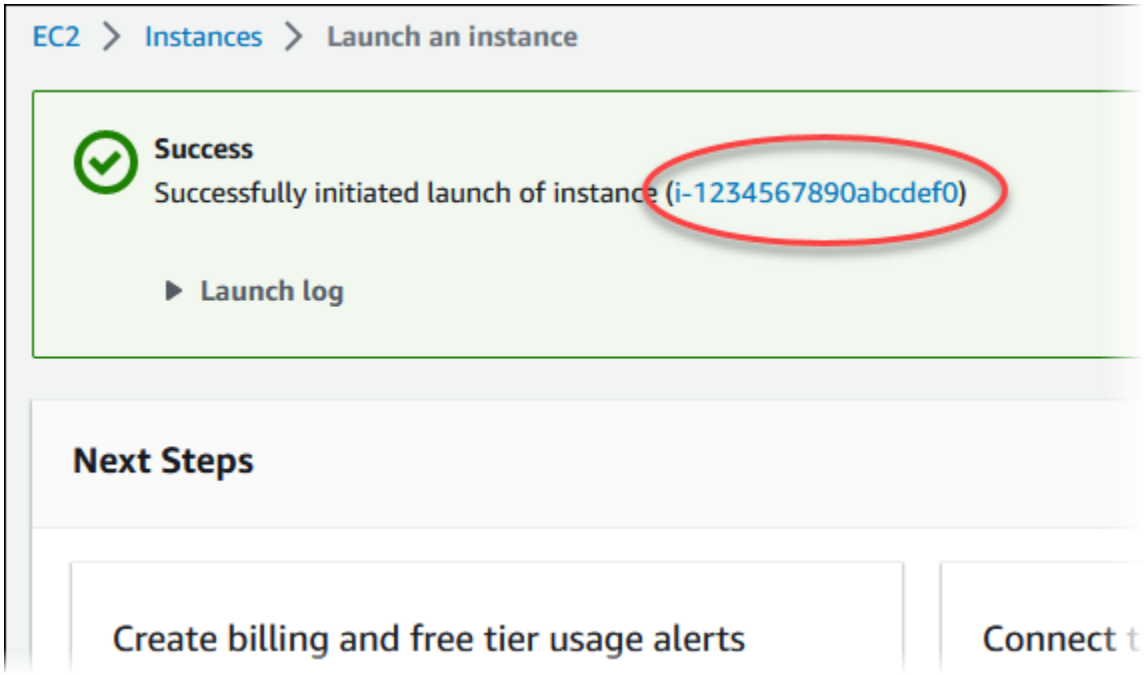
以下のイメージは、[ネットワーク設定] セクションの例を示しています。



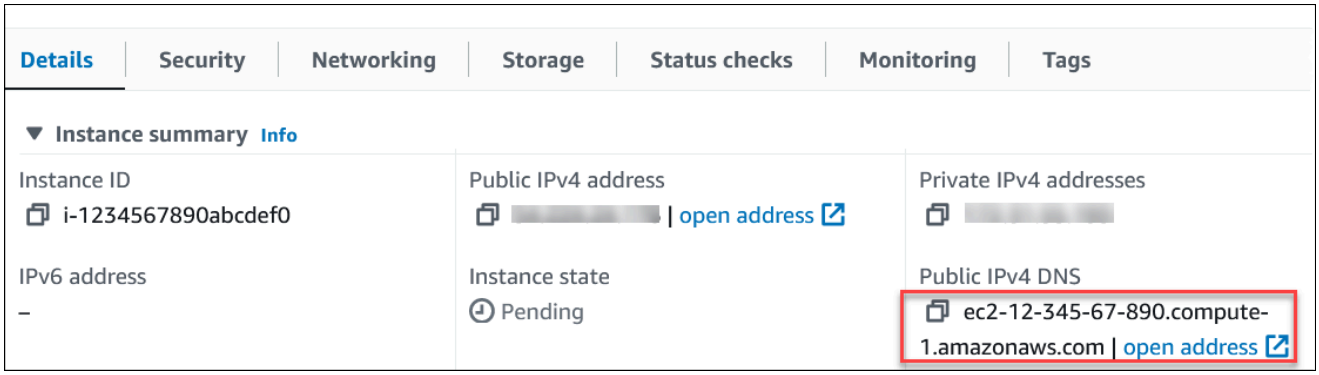
f. 残りのセクションをデフォルト値のままにします。

g. [概要] パネルで、EC2 インスタンス設定の概要を確認し、準備ができたら、[インスタンスの起動] を選択します。

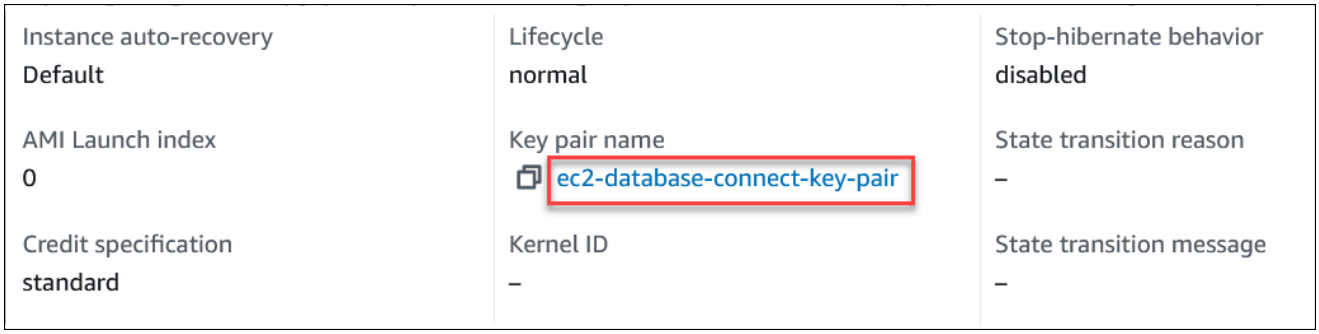
5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開き、EC2 インスタンスを選択します。
7. [詳細] タブで、SSH を使用して接続するときに必要な次の値を書き留めます。
   a. [インスタンスの概要] で、[パブリック IPv4 DNS] の値を書き留めます。



   b. [インスタンスの詳細] で、[キーペア名] の値を書き留めます。



8. EC2 インスタンスの [インスタンス状態] が [実行中] になるまで待ってから、続行します。

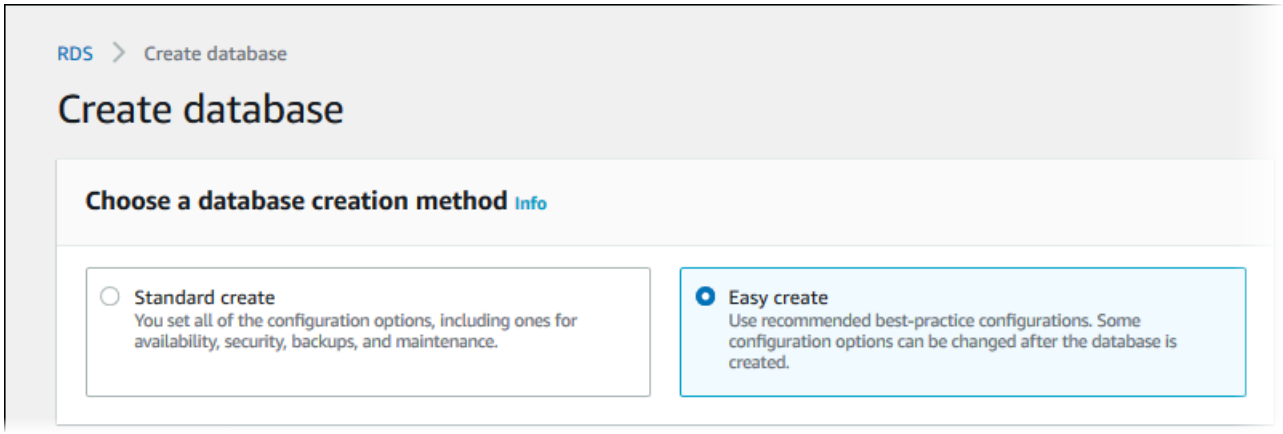
MariaDB DB インスタンスを作成する

Amazon RDS の基本的な構成要素は DB インスタンスです。これは、MariaDB データベースを実行する環境です。

この例では、[簡易作成] を使用して、db.t4g.micro インスタンスクラスの MariaDB データベースエンジンを実行する DB インスタンスを作成します。

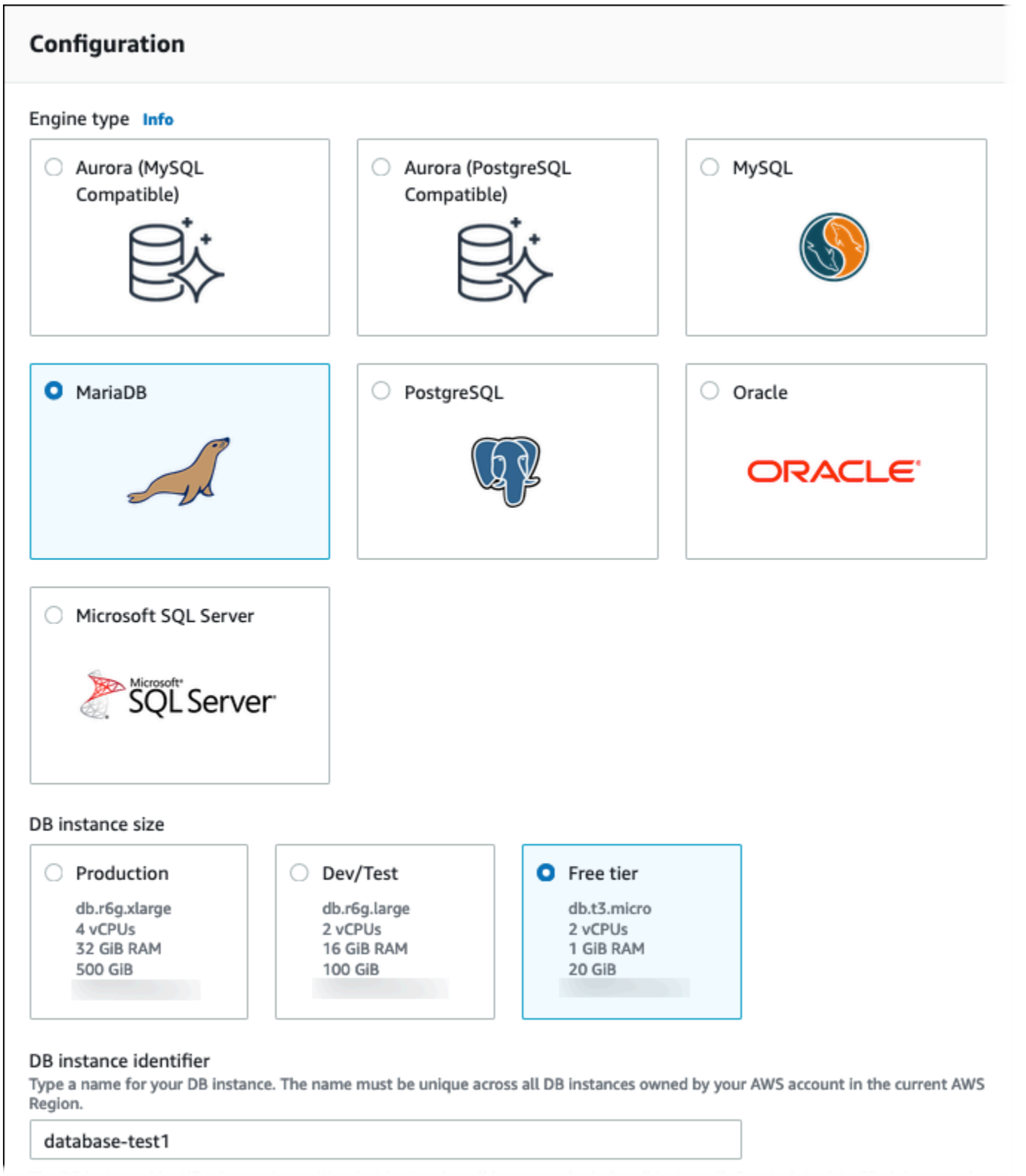
簡易作成を有効にして MariaDB DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[Databases] (データベース) を選択します。
4. [データベースの作成] を選択し、[簡易作成] を選択します。



5. [設定] で、[MariaDB] を選択します。
6. [DB インスタンスサイズ] で、[無料利用枠] または [サンドボックス] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。
7. [DB instance identifier] (DB インスタンス識別子) に**`database-test1`** と入力します。
8. [マスターユーザー名] に、マスターユーザーの名前を入力するか、デフォルト名のままにします。

[データベースの作成] ページは、次のイメージのようになります。無料プランアカウントの場合、[無料利用枠] が表示されます。有料プランアカウントの場合、[サンドボックス] が表示されます。

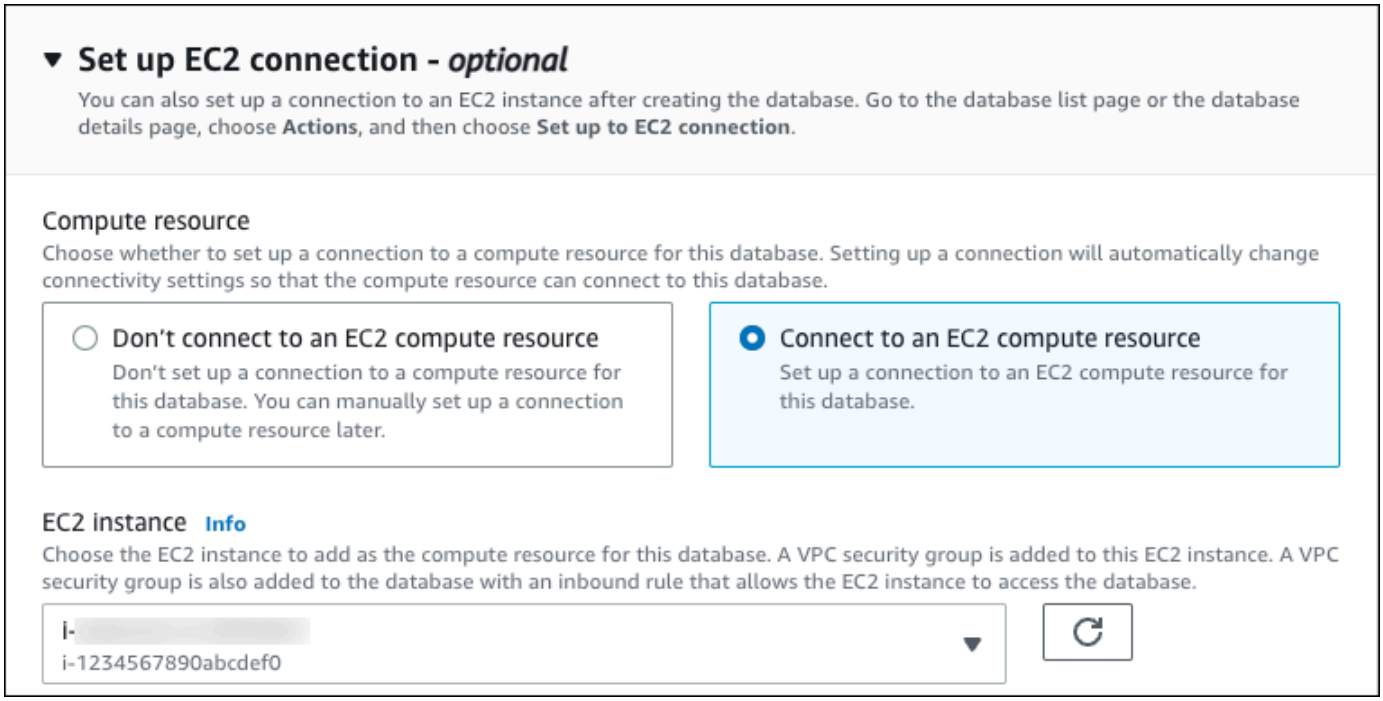


9. DB インスタンス用に自動生成されたマスターパスワードを使用するには、[パスワードの自動生成] を選択します。

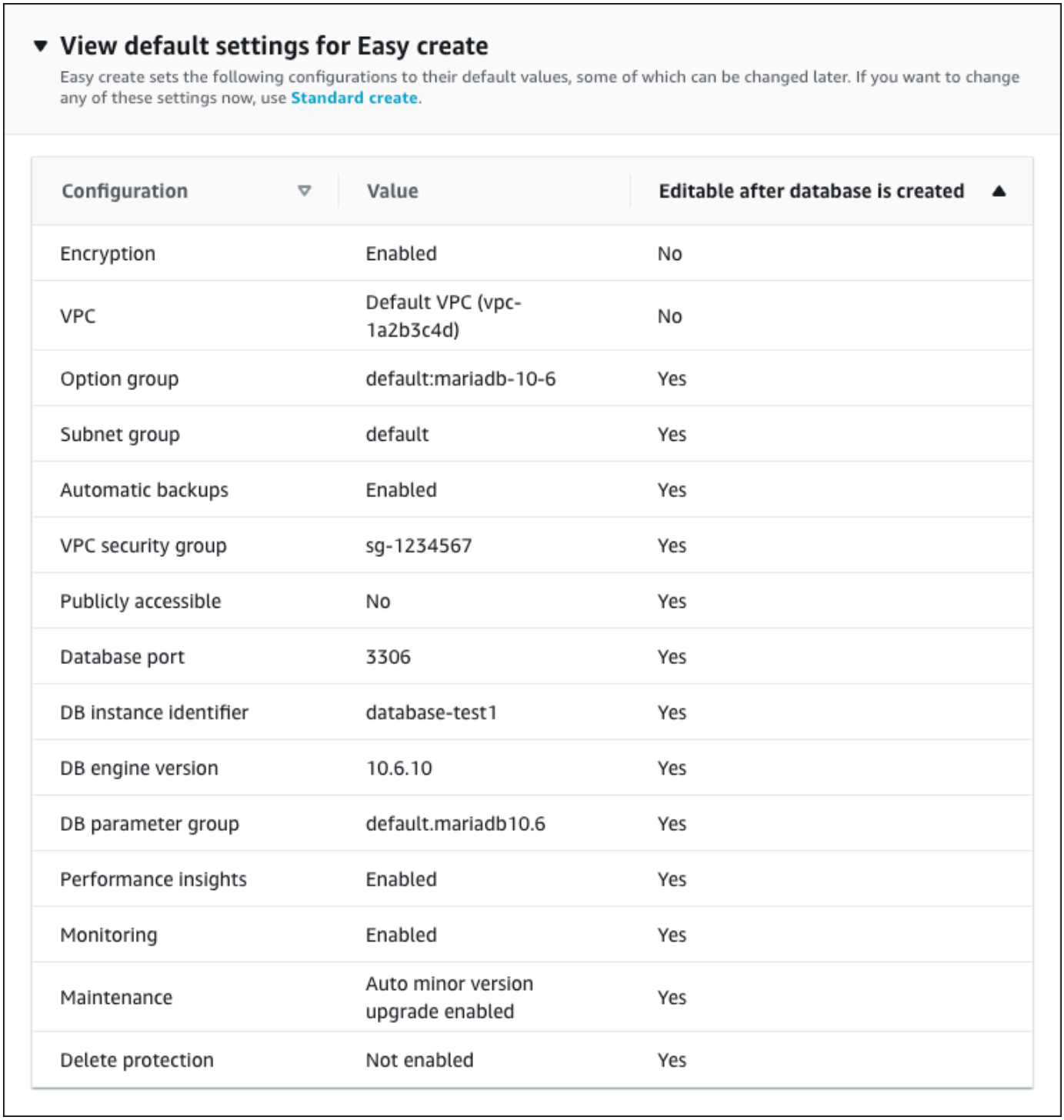
マスターパスワードを入力するには、[パスワードの自動生成] チェックボックスをオフにして、[マスターパスワード] と [パスワードの確認] に同じパスワードを入力します。

10. 以前に作成した EC2 インスタンスとの接続をセットアップするには、[EC2 接続のセットアップ - オプション] を展開します。

    [EC2 コンピューティングリソースに接続] を選択します。以前に作成した EC2 インスタンスを選択します。



11. [簡易作成のデフォルト設定を表示] を展開します。



▼ View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use Standard create.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:mariadb-10-6	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB instance identifier	database-test1	Yes
DB engine version	10.6.10	Yes
DB parameter group	default.mariadb10.6	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[Easy Create (簡易作成)] で使用されるデフォルト設定を調べることができます。[データベース作成後に編集可能] 列には、データベース作成後に変更できるオプションが表示されます。

- その列の設定に [いいえ] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成できます。

- その列の設定に [はい] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成するか、DB インスタンスの作成後に設定を変更できます。

12. [データベースの作成] を選択します。

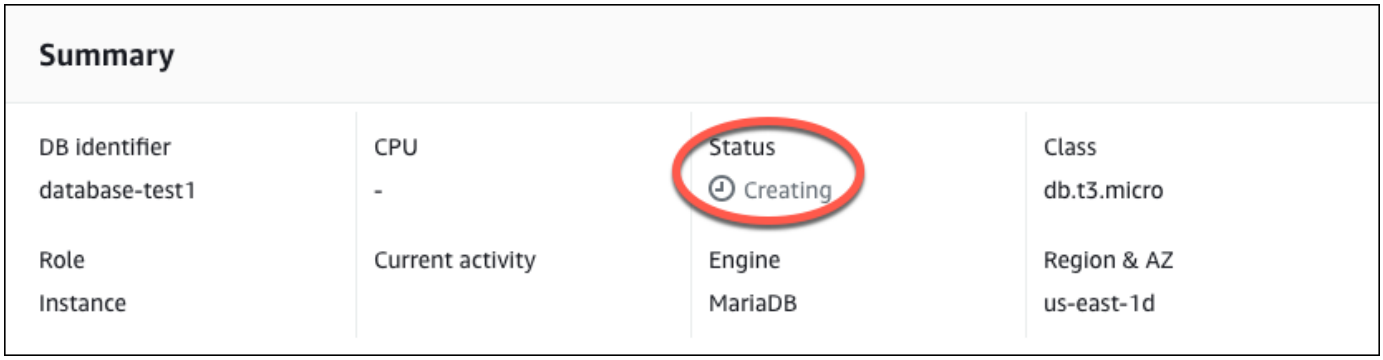
    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    表示されるユーザー名とパスワードを使用して、マスターユーザーとして DB インスタンスに接続できます。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。
    > DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンスを変更することができます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

13. [データベース] リストで、新しい MariaDB DB インスタンスの名前を選択し、詳細を表示します。

    DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは [作成中] のままです。



    ステータスが [Available] (利用可能) に変わったら、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しいインスタンスを使用できるようになるまで最長 20 分かかることがあります。

(オプション) CloudFormation を使用して VPC、EC2 インスタンス、MariaDB インスタンスを作成する

コンソールを使用して VPC、EC2 インスタンス、MariaDB DB インスタンスを作成する代わりに、CloudFormation を使用して Infrastructure as Code として取り扱うことで、AWS リソースをプロビジョンできます。AWS リソースをより小さく管理しやすい単位に整理するには、CloudFormation のネストされたスタック機能を使用できます。詳細については、「CloudFormation コンソールでのスタックの作成」と「ネストされたスタックの操作」を参照してください。

> **Important**
>
> CloudFormation は無料ですが、CloudFormation が作成するリソースは実動のものです。これらのリソースを終了するまで、標準使用料が発生します。詳細については、「RDS for MariaDB の料金」を参照してください。

- CloudFormation テンプレートをダウンロードする
- CloudFormation を使用してリソースを設定する

CloudFormation テンプレートをダウンロードする

CloudFormation テンプレートは、スタックに作成するリソースに関する設定情報が含まれる JSON または YAML のテキストファイルです。このテンプレートは、RDS インスタンスとともに VPC と踏み台ホストも作成します。

テンプレートファイルをダウンロードするには、次のリンク、MariaDB CloudFormation template を開きます。

この Github ページで、[Download raw file] ボタンをクリックしてテンプレートの YAML ファイルを保存します。

CloudFormation を使用してリソースを設定する

> **Note**
>
> このプロセスを開始する前に、AWS アカウント に EC2 インスタンスのキーペアがあることを確認してください。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

CloudFormation テンプレートを使用する場合は、適切なパラメータを選択して、リソースが正しく作成されていることを確認する必要があります。以下のステップを実行します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [テンプレートの指定] セクションで、[コンピュータからテンプレートファイルをアップロード] を選択し、[次へ] をクリックします。
4. [スタックの詳細を指定] ページで、次のパラメータを設定します。
   a. [スタック名] は [MariaDBTestStack] に設定します。
   b. [パラメータ] で、3 つのアベイラビリティーゾーンを選択して、[アベイラビリティーゾーン] を設定します。
   c. [Linux 踏み台ホスト設定] で、[キー名] に EC2 インスタンスにログインするキーペアを選択します。
   d. [Linux 踏み台ホスト設定] で、[許可された IP 範囲] を IP アドレスに設定します。Secure Shell (SSH) を使用して VPC 内の EC2 インスタンスに接続するには、https://checkip.amazonaws.com のサービスを使用してパブリック IP アドレスを確認します。IP アドレスの例は 192.0.2.1/32 です。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

e. [Database General configuration] で、[データベースインスタンスクラス] を [db.t3.micro] に設定します。

f. [データベース名] を **`database-test1`** に設定します。

g. [データベースマスターユーザー名] には、PDB のマスターユーザー名を入力します。

h. このチュートリアルでは、[Secrets Manager で DB マスターユーザーパスワードを管理] を `false` に設定します。

i. [データベースパスワード] には、任意のパスワードを設定します。このパスワードは、チュートリアルの他の手順のために覚えておいてください。

j. [Database Storage configuration] で、[Database storage type] を [gp2] に設定します。

k. [Database Monitoring configuration] で、[Enable RDS Performance Insights] を false に設定します。

l. その他の設定はすべてデフォルトの値のままにします。[次へ] をクリックして先に進みます。

5. [スタックの確認] ページで、データベースと Linux 踏み台ホストのオプションを確認した後、[送信] をクリックします。

スタックの作成プロセスが完了したら、BastionStack と RDSNS という名前のスタックを確認して、データベースへの接続に必要な情報をメモします。詳細については、「AWS マネジメントコンソール での CloudFormation スタックデータとリソースの表示」を参照してください。

MariaDB DB インスタンスに接続する

によって DB インスタンスがプロビジョニングされると、標準の SQL クライアントアプリケーションを使用してインスタンスに接続できます。この例では、mysql コマンドラインツールを使用して Maria DB インスタンス上のデータベースに接続します。

MariaDB DB インスタンスに接続する

1. DB インスタンスのエンドポイント (DNS 名) とポート番号を見つけます。

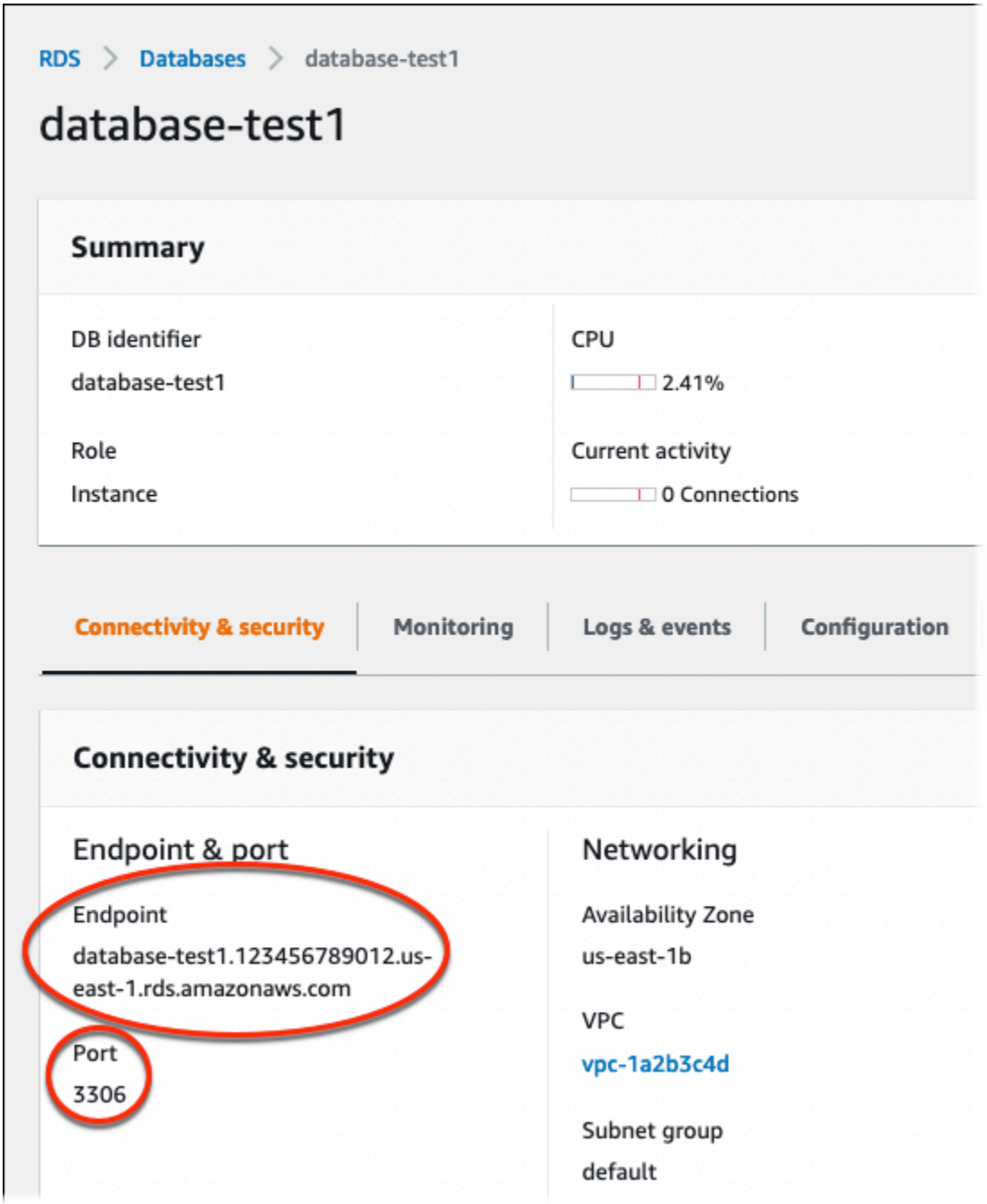
   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[Databases] (データベース) を選択します。

   d. MariaDB DB インスタンスの名前を選択して詳細を表示します。

e. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



2. 「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。

SSH を使用して EC2 インスタンスに接続することをお勧めします。SSH クライアントユーティリティが Windows、Linux、または Mac にインストールされている場合は、次のコマンド形式でインスタンスに接続できます。

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

例えば、ec2-database-connect-key-pair.pem が Linux の /dir1 に保存されていて、EC2 インスタンスのパブリック IPv4 DNS が ec2-12-345-678-90.compute-1.amazonaws.com であるとします。SSH コマンドは次のようになります。

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 インスタンスのソフトウェアを更新して、最新のバグ修正とセキュリティ更新を入手します。これを行うには、次のコマンドを使用します。

> **Note**
>
> -y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を確認するには、このオプションを省略します。

```
sudo dnf update -y
```

4. MariaDB から mysql コマンドラインクライアントをインストールします。

   MariaDB の コマンドラインクライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install mariadb105
```

5. DB インスタンスに接続します。例えば、次のコマンドを入力します。このアクションにより、MySQL クライアントを使用して MariaDB DB インスタンスに接続できます。

   endpoint の DB インスタンスの DNS 名 (エンドポイント) を、*admin* で使用したマスターユーザー名に置き換えます。パスワードの入力を求められたときに使用したマスターパスワードを入力します。

```
mysql -h endpoint -P 3306 -u admin -p
```

ユーザーのパスワードを入力すると、次のような出力が表示されます。

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 156
Server version: 10.6.10-MariaDB-log managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

MariaDB DB インスタンスへの接続の詳細については、「MariaDB DB インスタンスへの接続」を参照してください。DB インスタンスに接続できない場合は、「Amazon RDS DB インスタンスに接続できない」を参照してください。

セキュリティ上の理由で、暗号化された接続を使用することがベストプラクティスです。クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない MySQL 接続を使用します。暗号化された接続の使用については、「MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MariaDB DB インスタンスへの接続 (暗号化)」を参照してください。

6. SQL コマンドを実行する。

   例えば、次の SQL コマンドは、現在の日付と時刻を表示します。

```
SELECT CURRENT_TIMESTAMP;
```

EC2 インスタンスと DB インスタンスを削除する

作成したサンプル EC2 インスタンスと DB インスタンスに接続して、探索したら、料金がこれ以上発生しないように、それらを削除します。

CloudFormation を使用してリソースを作成した場合は、このステップをスキップして次のステップに進みます。

EC2 インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. EC2 インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Terminate (終了)] を選択してください。

EC2 インスタンスの削除の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。

最終的な DB スナップショットを作成せずに DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除する DB インスタンスを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成] と [自動バックアップの保持] をクリアします。
6. 確認を完了し、[削除] を選択します。

(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する

CloudFormation を使用してリソースを作成した場合は、サンプル EC2 インスタンスと DB インスタンスに接続して確認を済ませた後、それ以上料金が発生しないように、CloudFormation スタックを削除します。

CloudFormation リソースを削除するには

1. CloudFormation コンソールを開きます。
2. CloudFormation コンソールの [スタック] ページで、ルートスタック (VPCStack、BastionStack、または RDSNS という名前が付いていないスタック) を選択します。
3. [削除] を選択します。

4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudFormation でのスタックの削除方法の詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation コンソールでのスタックの削除」を参照してください。

(オプション) DB インスタンスを Lambda 関数に接続する

RDS for MariaDB DB インスタンスを Lambda サーバーレスコンピューティングリソースに接続することもできます。Lambda 関数を使用すると、インフラストラクチャをプロビジョニングしたり管理したりせずにコードを実行できます。Lambda 関数を使用すると、1 日に 数十件のイベントから 1 秒間に数百件のイベントまで、あらゆる規模のコード実行リクエストに自動的に応答することもできます。詳細については、「Lambda 関数と DB インスタンスを自動的に接続する」を参照してください。

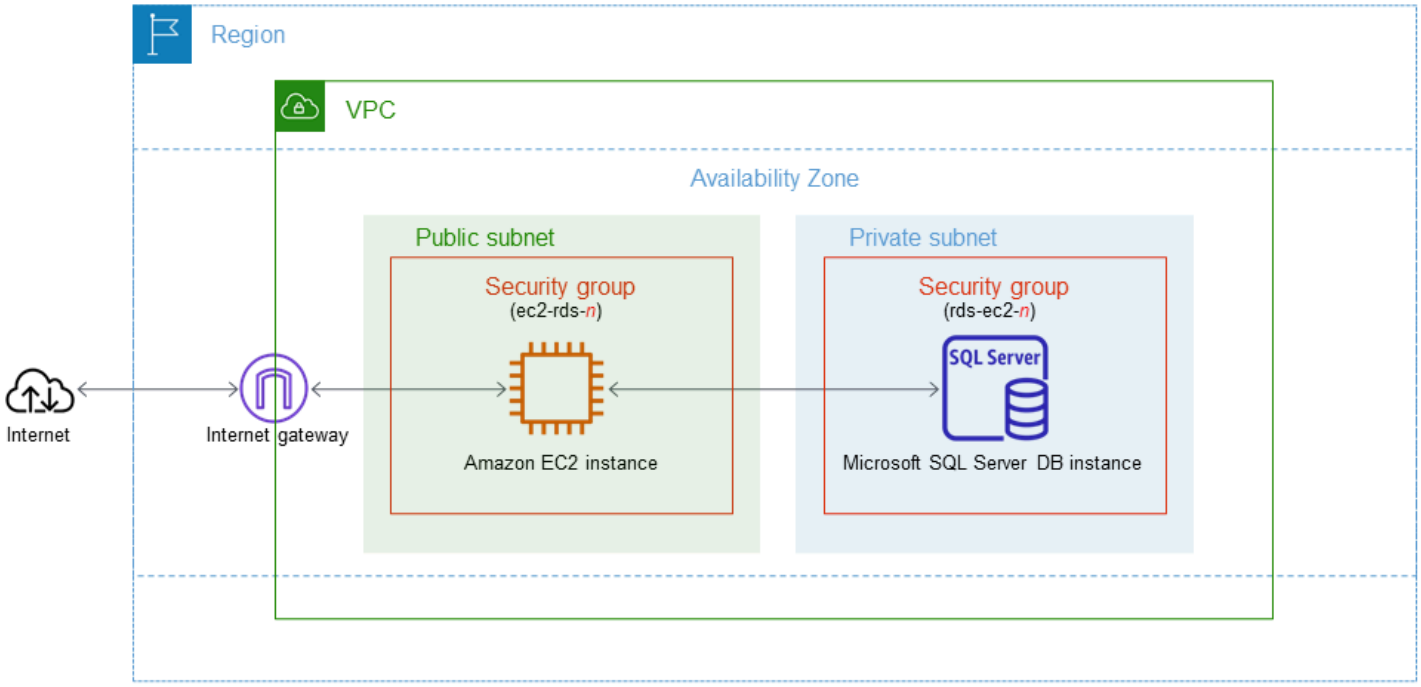
Microsoft SQL Server DB インスタンスを作成して接続する

このチュートリアルでは、EC2 インスタンスと RDS for MySQL DB インスタンスを作成します。このチュートリアルでは、Microsoft SQL Server Management Studio クライアントを使用して EC2 インスタンスから DB インスタンスにアクセスする方法を示します。ベストプラクティスとして、このチュートリアルでは、プライベート DB インスタンスを仮想プライベートクラウド (VPC) に作成します。ほとんどの場合、EC2 インスタンスなど、同じ VPC 内の他のリソースは DB インスタンスにアクセスできますが、VPC 外部のリソースはアクセスできません。

チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。1 つのアベイラビリティーゾーンで、EC2 インスタンスはパブリックサブネットにあり、DB インスタンスはプライベートサブネットにあります。

> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

次の図は、チュートリアルが完了した時点の設定を示しています。



このチュートリアルでは、次のいずれかの方法を使用してリソースを作成できます。

1. AWS マネジメントコンソール を使用する -「SQL Server DB インスタンスを作成する」と「EC2 インスタンスの作成」
2. CloudFormation を使用してデータベースインスタンスと EC2 インスタンスを作成する - (オプション) CloudFormationを使用して VPC、EC2 インスタンス、SQL Server インスタンスを作成する

最初の方法では、[簡単作成] を使用して、AWS マネジメントコンソール でプライベート SQL Server DB インスタンスを作成します。ここでは、DB エンジンタイプ、DB インスタンスサイズ、DB インスタンス識別子のみを指定します。[Easy create (簡易作成)] では、他の設定オプションのデフォルト設定を使用します。

代わりに [標準作成] を使用する場合は、DB インスタンスの作成時にさらに多くの設定オプションを指定できます。このようなオプションには、可用性、セキュリティ、バックアップ、メンテナンスの設定があります。パブリック DB インスタンスを作成するには、[標準作成] を使用する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

トピック

- 前提条件
- EC2 インスタンスの作成
- SQL Server DB インスタンスを作成する
- (オプション) CloudFormationを使用して VPC、EC2 インスタンス、SQL Server インスタンスを作成する
- SQL Server DB インスタンスに接続する
- サンプル SQL Server DB インスタンスを探索する
- EC2 インスタンスと DB インスタンスを削除する
- (オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する
- (オプション) DB インスタンスを Lambda 関数に接続する

前提条件

開始する前に、以下のセクションのステップを完了してください。

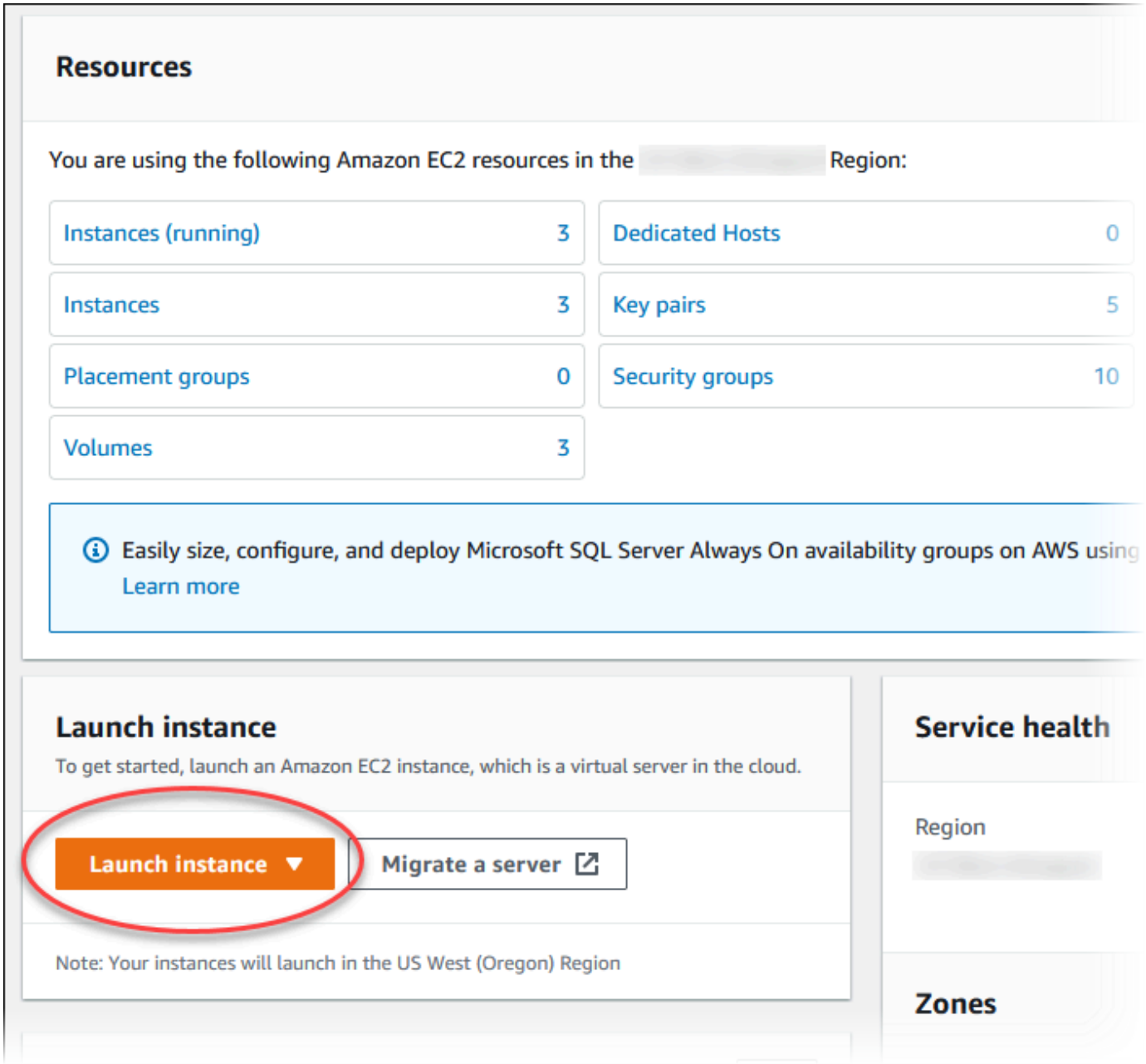
- AWS アカウント へのサインアップ

EC2 インスタンスの作成

データベースへの接続に使用する Amazon EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、以前にデータベースに使用した AWS リージョンを選択します。
3. 次の図に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの起動] を選択します。

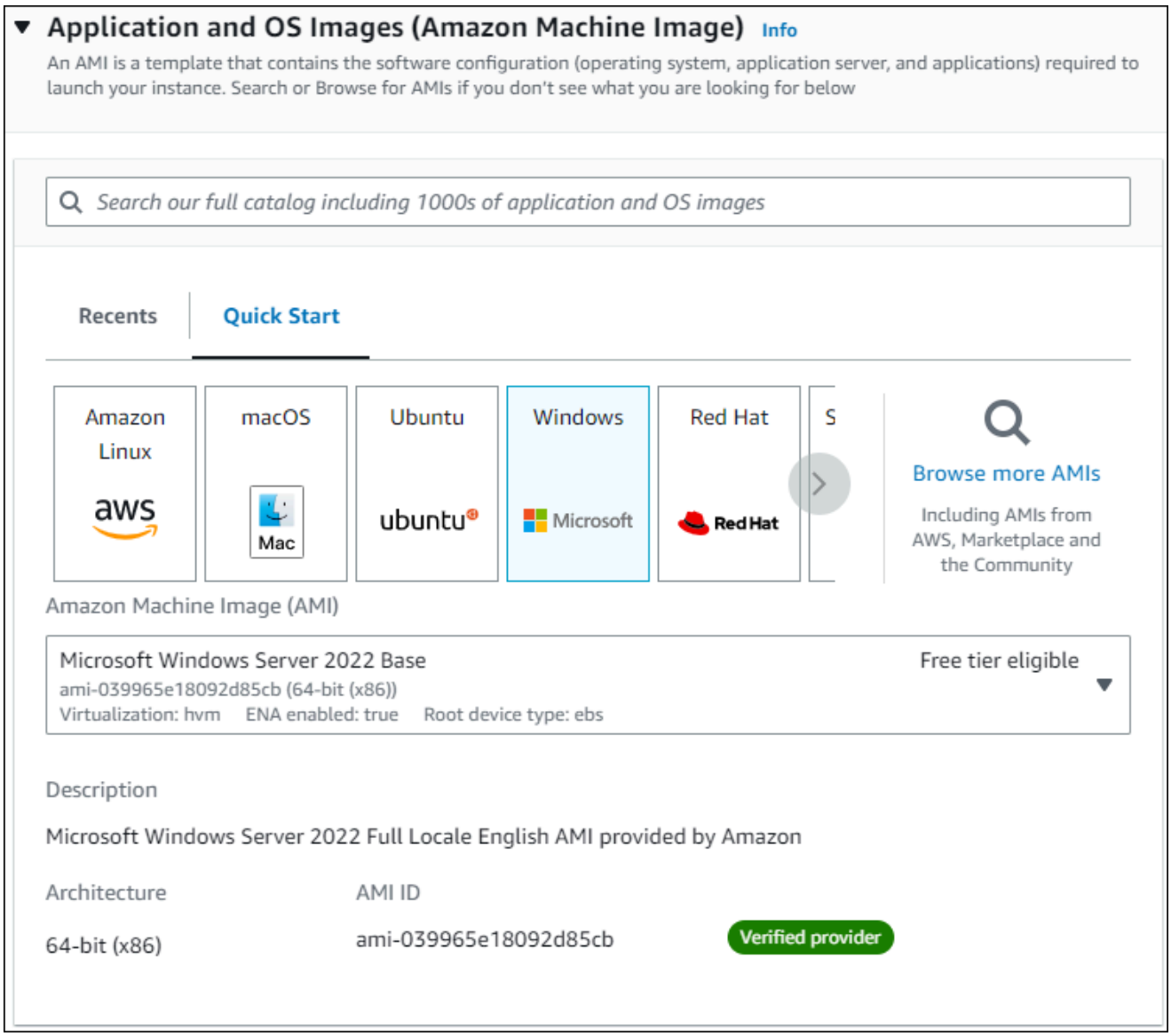


[インスタンスを起動] ページが開きます。

4. [インスタンスを起動] ページで次の設定を選択します。

   a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`ec2-database-connect`** と入力します。

   b. [アプリケーションと OS イメージ (Amazon マシンイメージ)] で [Windows] を選択し、次に [Microsoft Windows Server 2022 Base] を選択します。他の選択肢は、デフォルトの選択のままにします。



c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するには、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

   キーペアの作成については、Windows インスタンス用 Amazon EC2 ユーザーガイドの「キーペアの作成」を参照してください。

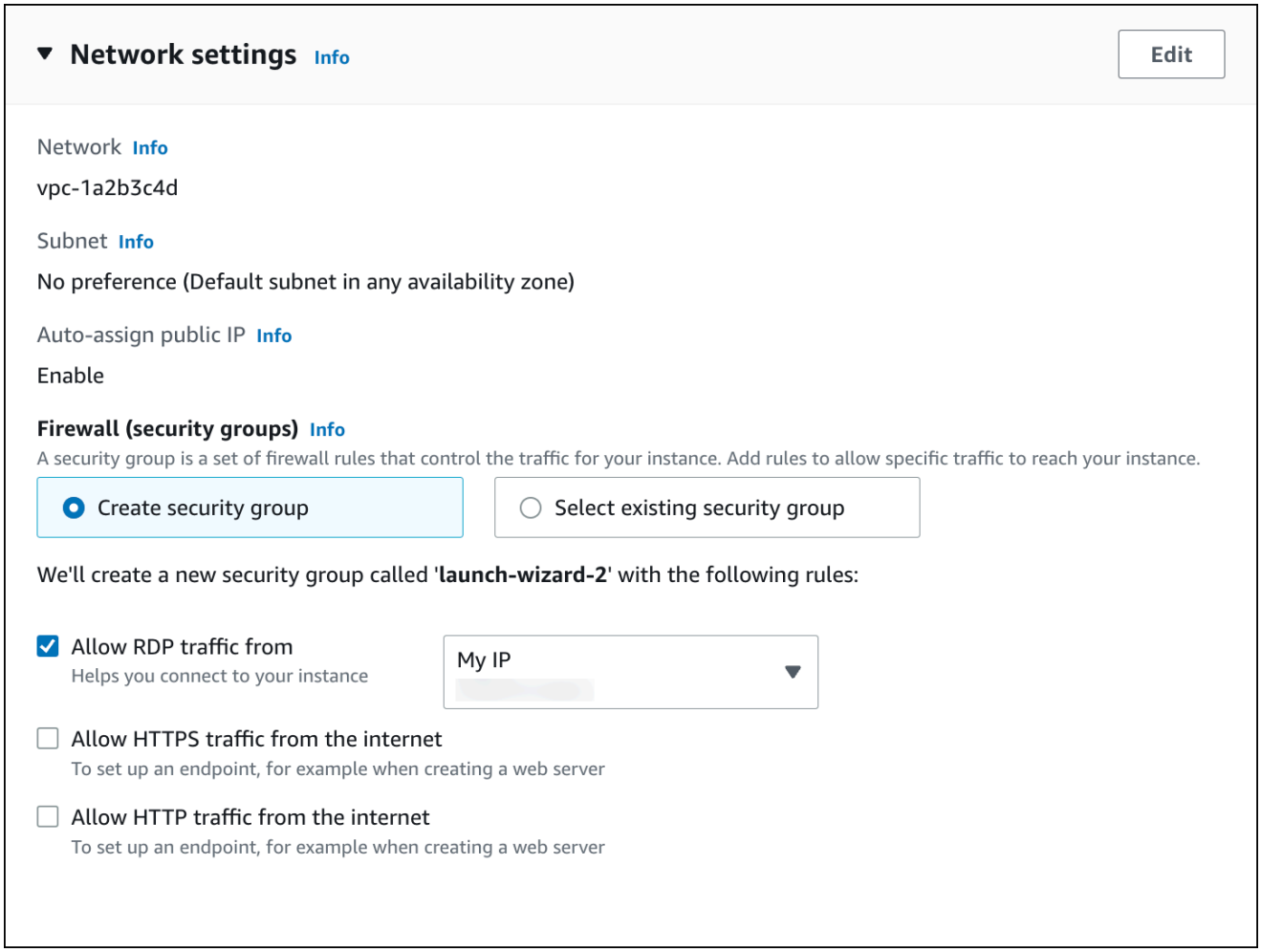
e. ネットワーク設定の [ファイアウォール (セキュリティグループ)] で、[接続元の RDP トラフィックを許可] を選択して EC2 インスタンスに接続します。

表示された IP アドレスが RDP 接続に適している場合は、[マイ IP] を選択できます。それ以外の場合は、RDP を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は 192.0.2.1/32 です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

> **Warning**
>
> RDP アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが RDP を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、RDP を使った EC2 インスタンスへのアクセスを承認します。

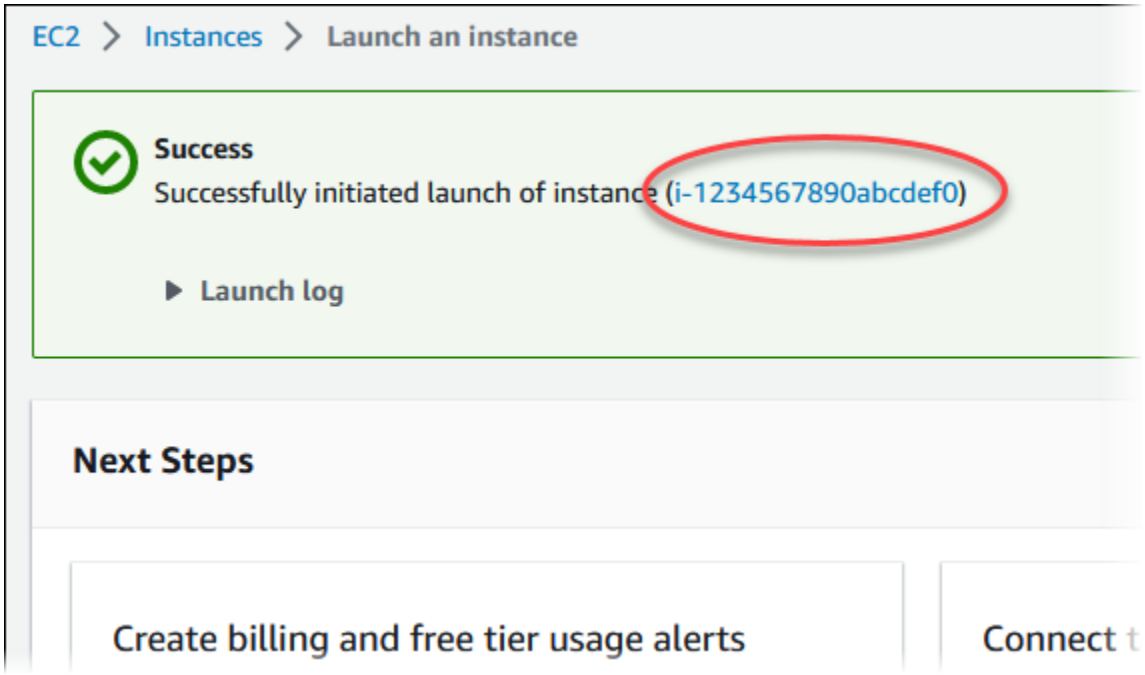
以下のイメージは、[ネットワーク設定] セクションの例を示しています。



f. 残りのセクションをデフォルト値のままにします。

g. [概要] パネルで、EC2 インスタンス設定の概要を確認し、準備ができたら、[インスタンスの起動] を選択します。

5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開きます。
7. EC2 インスタンスの [インスタンス状態] が [実行中] になるまで待ってから、続行します。

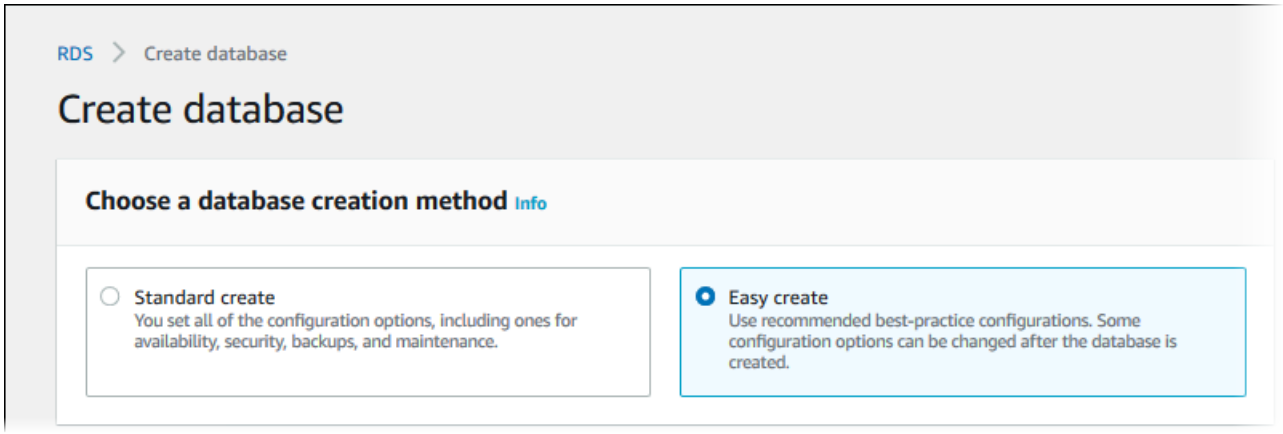
SQL Server DB インスタンスを作成する

Amazon RDS の基本的な構成要素は DB インスタンスです。これは、SQL Server データベースを実行する環境です。

この例では、[簡単作成] を使用して、db.t2.micro DB インスタンスクラスで SQL Server データベースエンジンを実行する DB インスタンスを作成します。

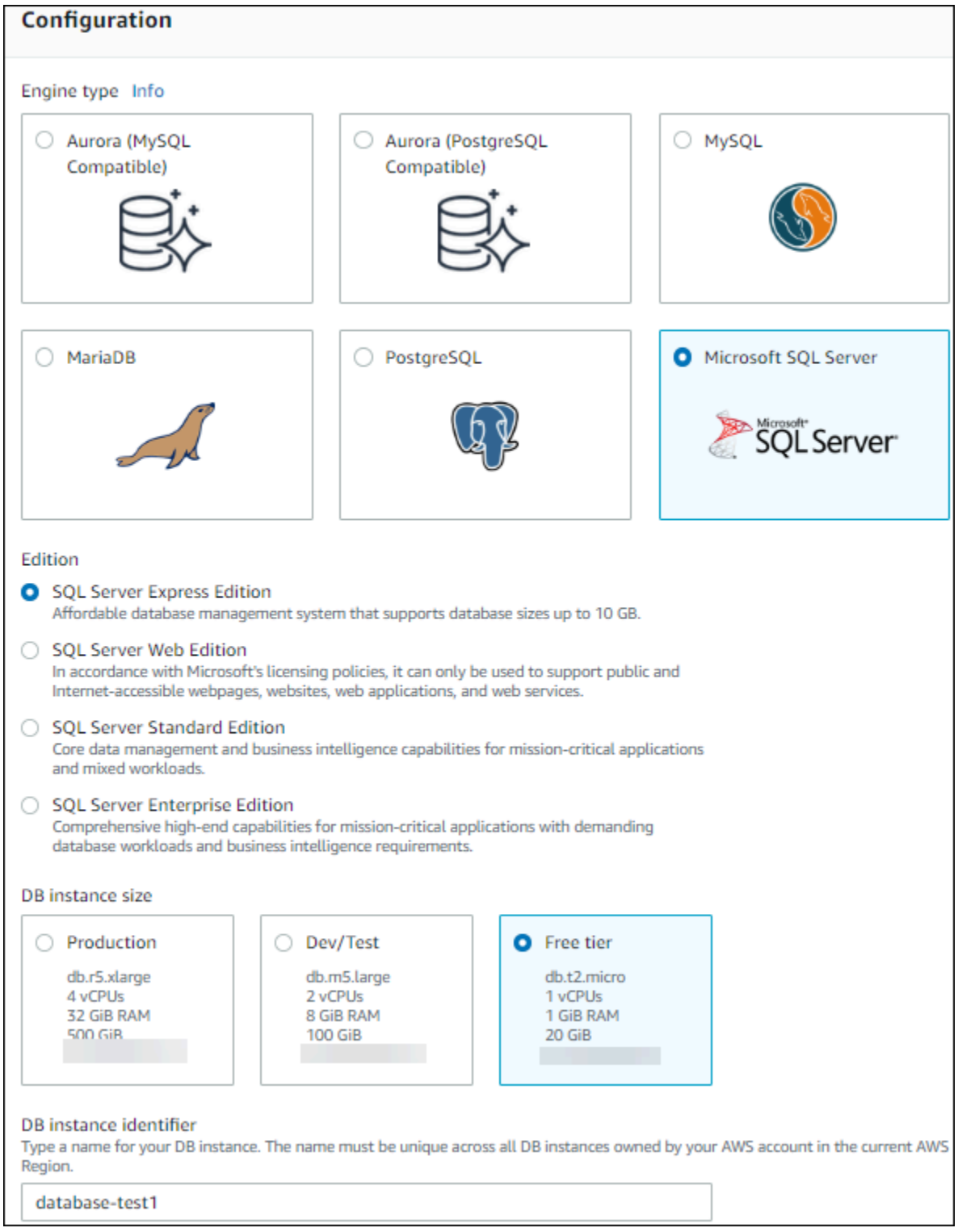
[Easy create (簡易作成)] を使用して Microsoft SQL Server DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [Create database (データベースの作成)] を選択し、[Easy create (簡易作成)] が選択されていることを確認します。



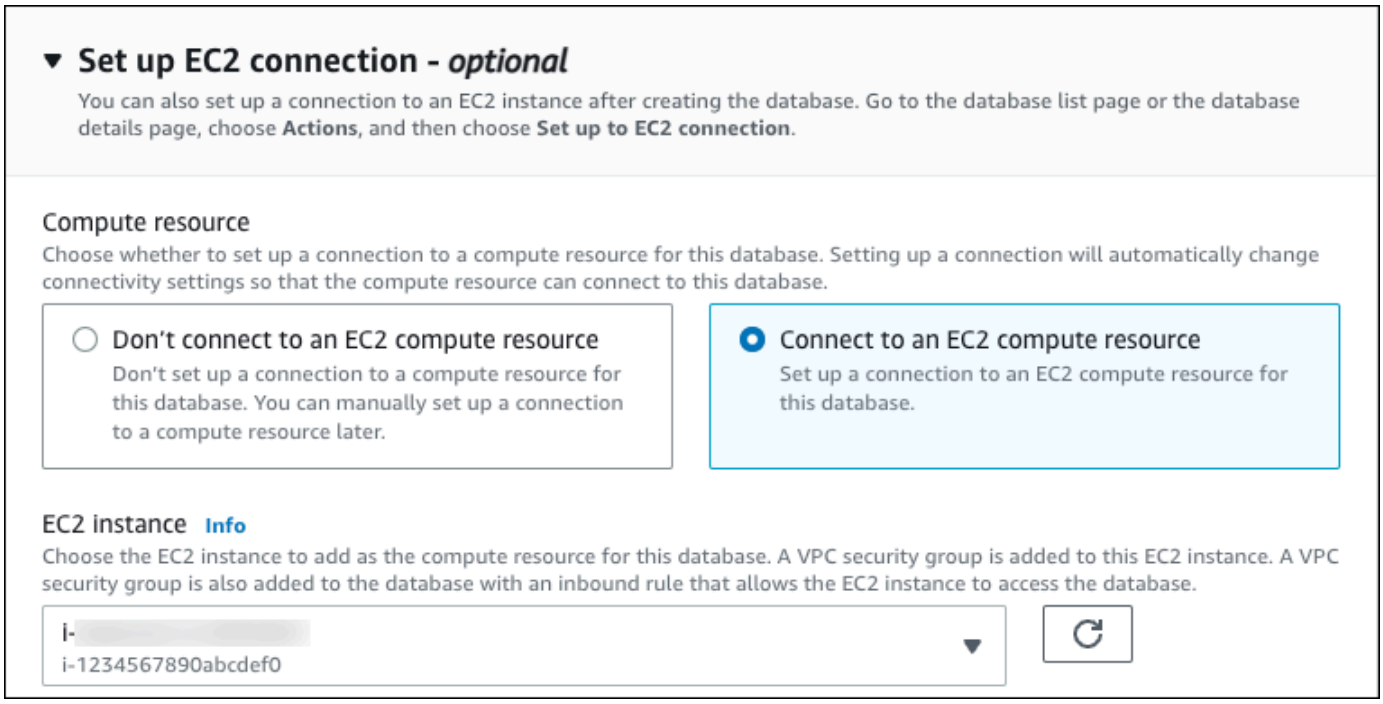
5. [設定] で、[Microsoft SQL Server] を選択します。
6. [エディション] には、[SQL Server Express Edition] を選択します。
7. [DB インスタンスサイズ] で、[無料利用枠] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。
8. [DB instance identifier] (DB インスタンス識別子) に**database-test1** と入力します。

   [データベースの作成] ページは、次のイメージのようになります。無料プランアカウントの場合、[無料利用枠] が表示されます。有料プランアカウントの場合、[サンドボックス] が表示されます。



9. [マスターユーザー名] に、マスターユーザーの名前を入力するか、デフォルト名のままにします。
10. 以前に作成した EC2 インスタンスとの接続をセットアップするには、[EC2 接続のセットアップ - オプション] を開きます。

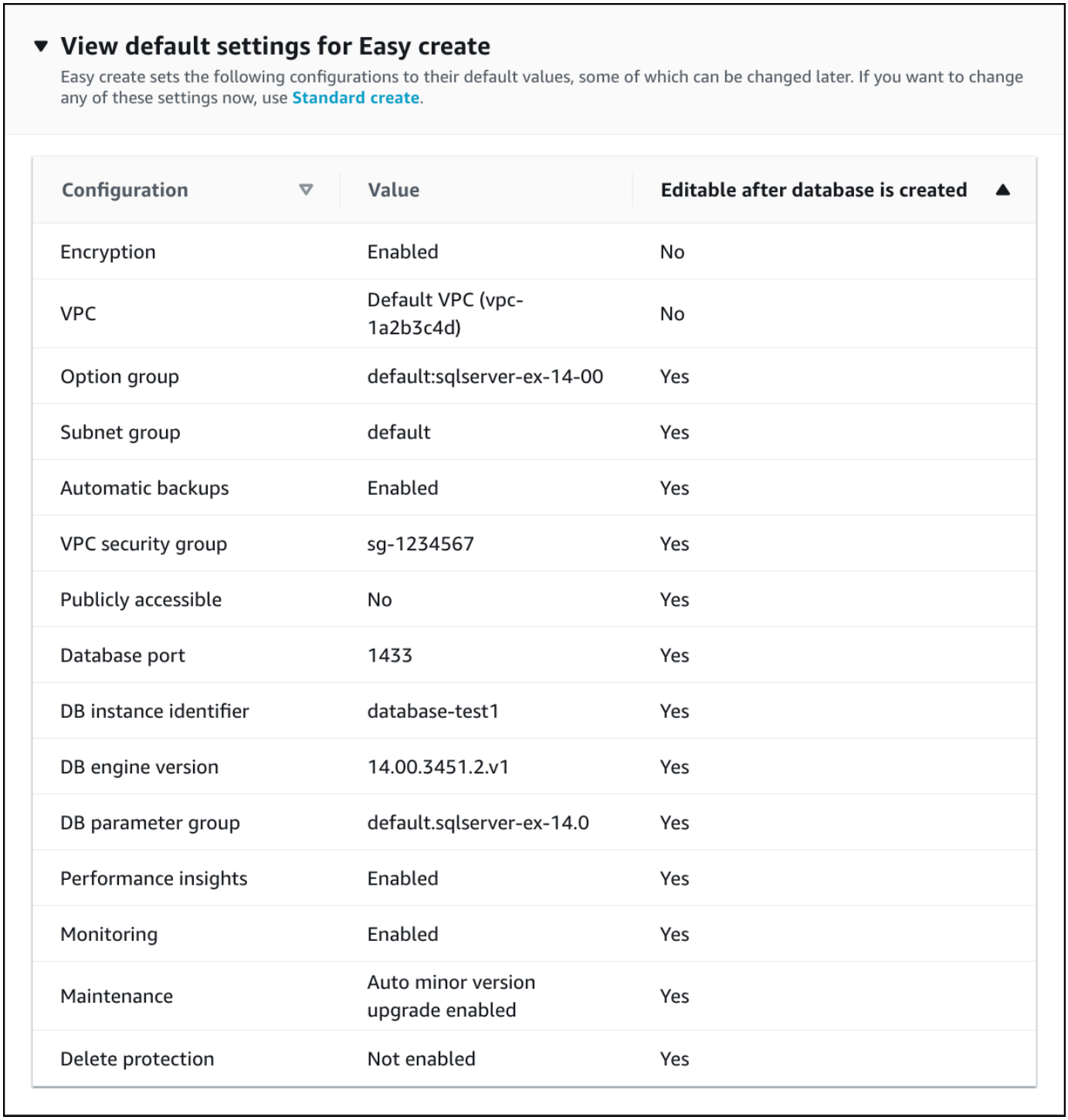
[EC2 コンピューティングリソースに接続] を選択します。以前に作成した EC2 インスタンスを選択します。



11. DB インスタンス用に自動生成されたマスターパスワードを使用するには、[Auto generate a password (パスワードの自動生成)] を選択します。

    マスターパスワードを入力するには、[Auto generate a password (パスワードの自動生成)] チェックボックスをオフにして、[Master password (マスターパスワード)] と [Confirm password (パスワードの確認)] に同じパスワードを入力します。

12. [簡易作成のデフォルト設定を表示] を開きます。



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:sqlserver-ex-14-00	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	1433	Yes
DB instance identifier	database-test1	Yes
DB engine version	14.00.3451.2.v1	Yes
DB parameter group	default.sqlserver-ex-14.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[Easy Create (簡易作成)] で使用されるデフォルト設定を調べることができます。[データベース作成後に編集可能] 列には、データベース作成後に変更できるオプションが表示されます。

- その列の設定に [いいえ] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成できます。

- その列の設定に [はい] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成するか、DB インスタンスの作成後に設定を変更できます。

13. [データベースの作成] を選択します。

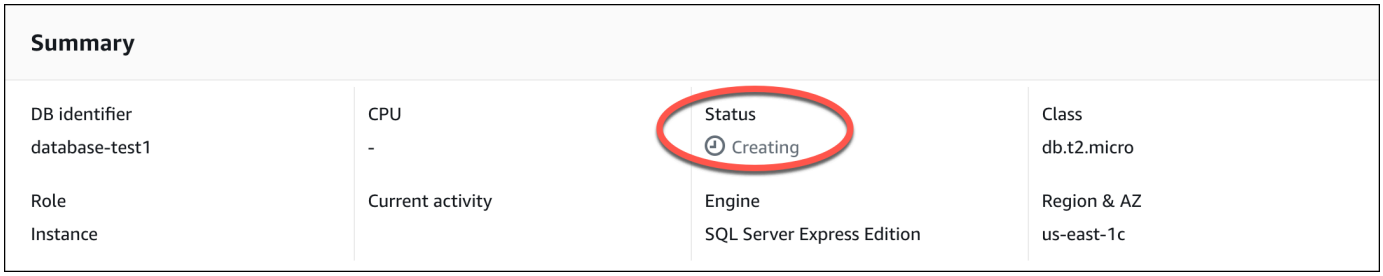
    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    表示されるユーザー名とパスワードを使用して、マスターユーザーとして DB インスタンスに接続できます。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。
    > DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンスを変更することができます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

14. [データベース] リストで、新しい SQL Server DB インスタンスの名前を選択し、詳細を表示します。

    DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは [作成中] のままです。



    ステータスが [Available] (利用可能) に変わったら、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しいインスタンスを使用できるようになるまで最長 20 分かかることがあります。

(オプション) CloudFormationを使用して VPC、EC2 インスタンス、SQL Server インスタンスを作成する

コンソールを使用して VPC、EC2 インスタンス、SQL Server インスタンスを作成する代わりに、CloudFormation を使用して Infrastructure as Code として取り扱うことで、AWS リソースをプロビジョンできます。AWS リソースをより小さく管理しやすい単位に整理するには、CloudFormation のネストされたスタック機能を使用できます。詳細については、「CloudFormation コンソールでのスタックの作成」と「ネストされたスタックの操作」を参照してください。

> **Important**
>
> CloudFormation には課金されません。ただし、CloudFormation が作成するリソースは本番稼働です。これらのリソースを終了するまで、標準使用料が発生します。詳細については、「RDS for SQL Server の料金」を参照してください。

CloudFormation コンソールを使用してリソースを作成するには、以下のステップを実行します。

- CloudFormation テンプレートをダウンロードする
- CloudFormation を使用してリソースを設定する

CloudFormation テンプレートをダウンロードする

CloudFormation テンプレートは、スタックに作成するリソースに関する設定情報が含まれる JSON または YAML のテキストファイルです。このテンプレートは、RDS インスタンスとともに VPC と踏み台ホストも作成します。

テンプレートファイルをダウンロードするには、次のリンク、SQL Server CloudFormation template を開きます。

この Github ページで、[Download raw file] ボタンをクリックしてテンプレートの YAML ファイルを保存します。

CloudFormation を使用してリソースを設定する

> **Note**
>
> このプロセスを開始する前に、AWS アカウント に EC2 インスタンスのキーペアがあることを確認してください。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

CloudFormation テンプレートを使用する場合は、適切なパラメータを選択して、リソースが正しく作成されていることを確認する必要があります。以下のステップを実行します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [テンプレートの指定] セクションで、[コンピュータからテンプレートファイルをアップロード] を選択し、[次へ] をクリックします。
4. [スタックの詳細を指定] ページで、次のパラメータを設定します。
   a. [スタック名] は [SQLServerTestStack] に設定します。
   b. [パラメータ] で、3 つのアベイラビリティーゾーンを選択して、[アベイラビリティーゾーン] を設定します。
   c. [Linux 踏み台ホスト設定] で、[キー名] に EC2 インスタンスにログインするキーペアを選択します。
   d. [Linux 踏み台ホスト設定] で、[許可された IP 範囲] を IP アドレスに設定します。Secure Shell (SSH) を使用して VPC 内の EC2 インスタンスに接続するには、https://checkip.amazonaws.com のサービスを使用してパブリック IP アドレスを確認します。IP アドレスの例は 192.0.2.1/32 です。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

e. [Database General configuration] で、[データベースインスタンスクラス] を [db.t3.micro] に設定します。

f. [データベース名] を **`database-test1`** に設定します。

g. [データベースマスターユーザー名] には、PDB のマスターユーザー名を入力します。

h. このチュートリアルでは、[Secrets Manager で DB マスターユーザーパスワードを管理] を `false` に設定します。

i. [データベースパスワード] には、任意のパスワードを設定します。このパスワードは、チュートリアルの他の手順のために覚えておいてください。

j. [Database Storage configuration] で、[Database storage type] を [gp2] に設定します。

k. [Database Monitoring configuration] で、[Enable RDS Performance Insights] を false に設定します。

l. その他の設定はすべてデフォルトの値のままにします。[次へ] をクリックして続行します。

5. [スタックオプションの設定] ページでは、すべてのデフォルトオプションをそのまま使用します。[次へ] をクリックして続行します。

6. [スタックの確認] ページで、データベースと Linux 踏み台ホストのオプションを確認した後、[送信] をクリックします。

スタックの作成プロセスが完了したら、BastionStack と RDSNS という名前のスタックを確認して、データベースへの接続に必要な情報をメモします。詳細については、「AWS マネジメントコンソール での CloudFormation スタックデータとリソースの表示」を参照してください。

SQL Server DB インスタンスに接続する

次の手順では、Microsoft SQL Server Management Studio (SSMS) を使用して、DB インスタンスに接続します。

SSMS を使用して、RDS for SQL Server DB インスタンスに接続するには

1. DB インスタンスのエンドポイント (DNS 名) とポート番号を見つけます。

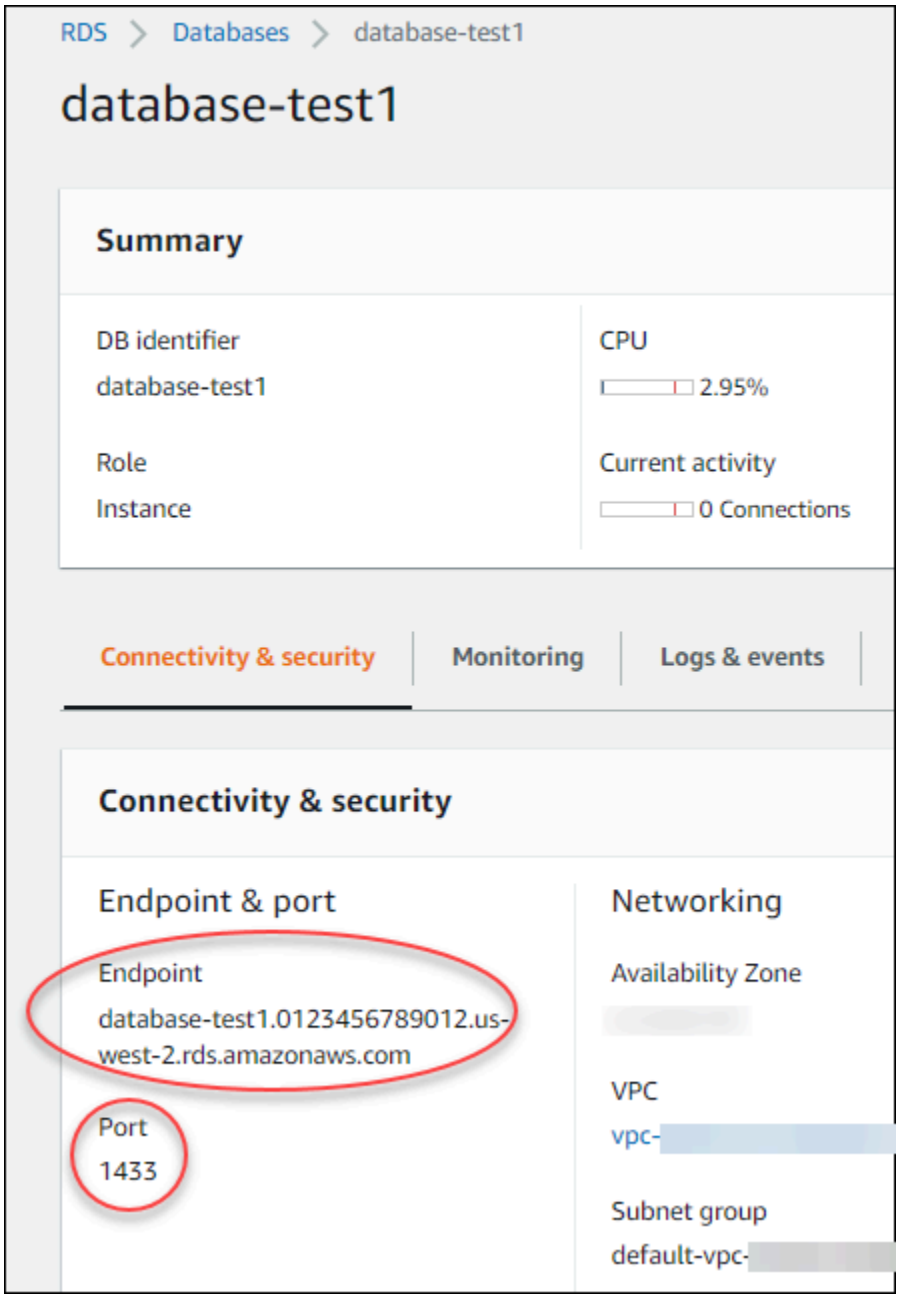
   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[Databases] (データベース) を選択します。

   d. SQL Server DB インスタンスの名前を選択して詳細を表示します。

e. [接続] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



2. Windows インスタンスに関する Amazon EC2 ユーザーガイドの「Microsoft Windows インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。
3. SQL Server Management Studio (SSMS) クライアントをインストールします。

   この SSMS のスタンドアロンバージョンを EC2 インスタンスにダウンロードするには、Microsoft ドキュメントの 「Download SQL Server Management Studio (SSMS)」を参照してください。

a. スタートメニューを使用して、Internet Explorer を開きます。

b. Internet Explorer を使用して、SSMS のスタンドアロンバージョンをダウンロードしてインストールします。サイトが信頼されていないことを示すメッセージが表示されたら、そのサイトを信頼済みサイトのリストに追加します。

4. SQL Server Management Studio (SSMS) を起動します。

   [サーバーに接続] ダイアログボックスが表示されます。

5. サンプルの DB インスタンスに以下の情報を入力します。

   a. [サーバーの種類] で、[データベース エンジン] を選択します。

   b. [サーバー名] に、DNS 名、カンマ、ポート番号 (デフォルトポートは 1433) の順に入力します。サーバー名は以下の例のようになります。

   ```
   database-test1.0123456789012.us-west-2.rds.amazonaws.com,1433
   ```

   c. [認証] で、[SQL Server 認証] を選択します。

   d. [ログイン] に、サンプル DB インスタンスを使用するために選択したユーザー名を入力します。これは、マスターユーザー名とも呼ばれます。

   e. [パスワード] に、先ほど選択した、サンプル DB インスタンスのパスワードを入力します。これは、マスターユーザーパスワードとも呼ばれます。

6. [接続] を選択します。

   しばらくすると、SSMS が DB インスタンスに接続されます。セキュリティのためには、暗号化された接続を使用することがベストプラクティスです。クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない SQL Server 接続を使用します。暗号化された接続の使用については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください

Microsoft SQL Server DB インスタンスへの接続の詳細については、SQL Server DB インスタンスへの接続 を参照してください。

接続の問題については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

サンプル SQL Server DB インスタンスを探索する

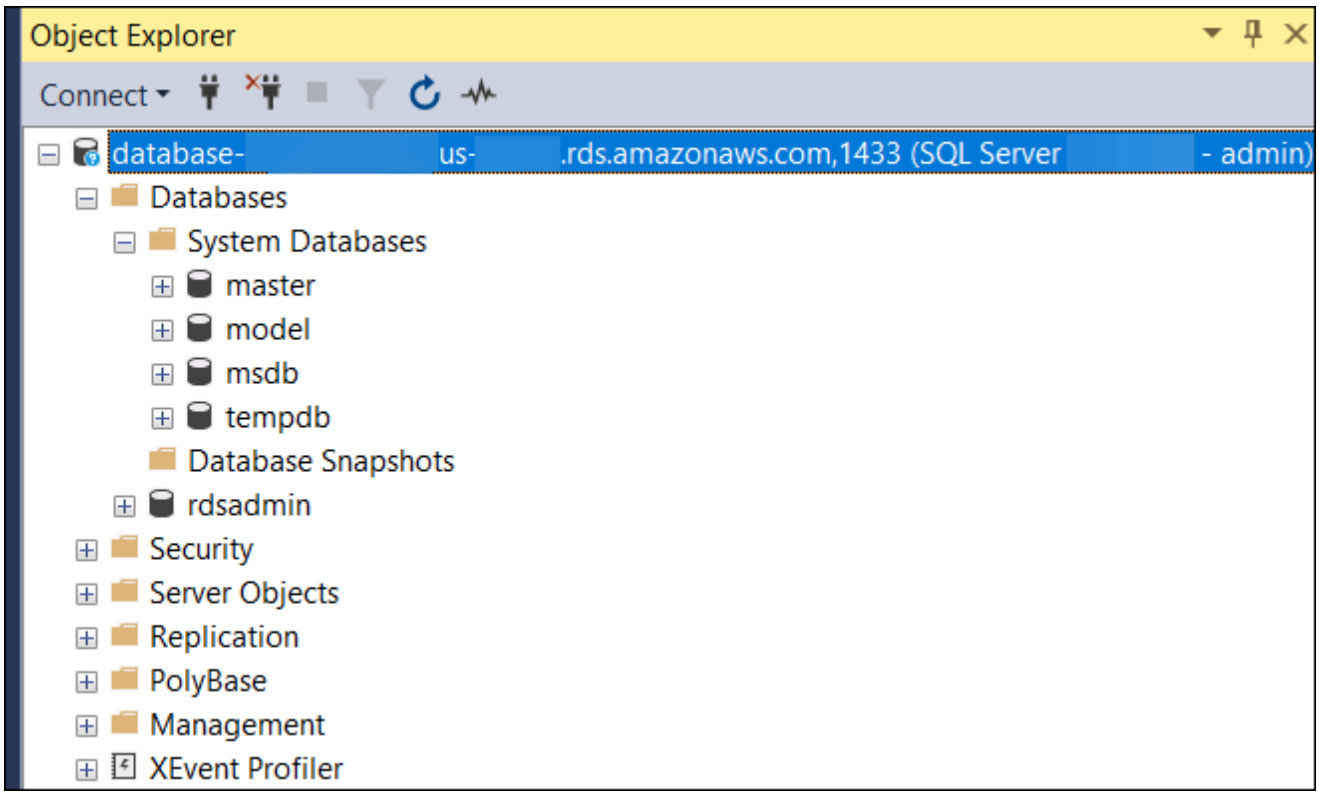
Microsoft SQL Server Management Studio (SSMS) を使用して、サンプル DB インスタンスを探索できます。

SSMS を使用して DB インスタンスを試すには

1. SQL Server DB インスタンスには、SQL Server の標準内蔵システムデータベース (master、model、msdb、tempdb) が含まれています。システムデータベースを閲覧するには、次を実行します。

   a. SSMS の [ビュー] メニューで、[オブジェクト エクスプローラー] を選択します。

   b. 図のように、DB インスタンスを展開し、[Databases (データベース)] を展開します。その後、以下のように [System Databases (システムデータベース)] を展開します。



   SQL Server DB インスタンスには、`rdsadmin` という名前のデータベースも含まれています。Amazon RDS では、このデータベースを使用して、データベースを管理するために使用するオブジェクトを保存します。`rdsadmin` データベースには、詳細なタスクを実行するために保存された手順も含まれます。

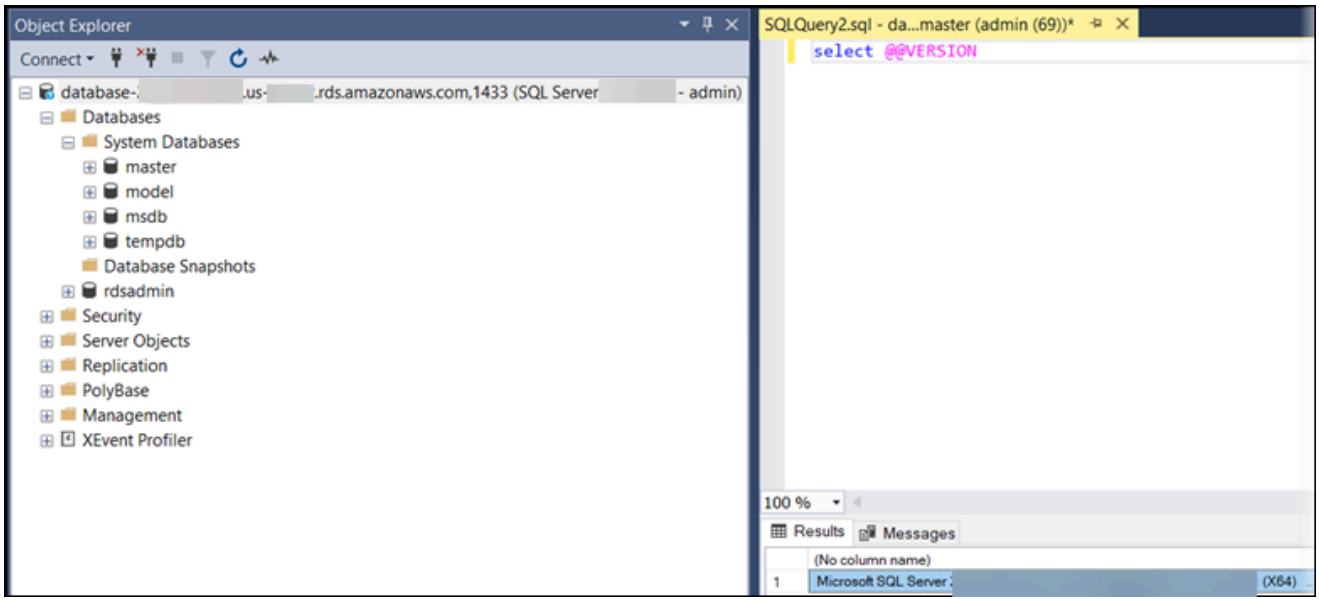
2. 独自のデータベース作成、および通常の DB インスタンスとデータベースに対するクエリの実行をスタートします。サンプルの DB インスタンスに対してテストクエリを実行するには、次を実行します。

   a. SSMS の [ファイル] メニューで [新規] をポイントし、[クエリを現在の接続で実行] を選択します。

b. 次の SQL クエリを入力します。

```
select @@VERSION
```

c. クエリを実行します。SSMS は、Amazon RDS DB インスタンスの SQL Server のバージョンを返します。



EC2 インスタンスと DB インスタンスを削除する

作成したサンプル EC2 インスタンスと DB インスタンスに接続して、探索したら、料金がこれ以上発生しないように、それらを削除します。

CloudFormation を使用してリソースを作成した場合は、このステップをスキップして次のステップに進みます。

EC2 インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. EC2 インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Terminate (終了)] を選択してください。

EC2 インスタンスの削除の詳細については、Windows インスタンス用ユーザーガイドの「インスタンスの終了」を参照してください。

最終的な DB スナップショットを作成せずに DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除する DB インスタンスを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成] と [自動バックアップの保持] をクリアします。
6. 確認を完了し、[削除] を選択します。

(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する

CloudFormation を使用してリソースを作成した場合は、サンプル EC2 インスタンスと DB インスタンスに接続して確認を済ませた後、それ以上料金が発生しないように、CloudFormation スタックを削除します。

CloudFormation リソースを削除するには

1. CloudFormation コンソールを開きます。
2. CloudFormation コンソールの [スタック] ページで、ルートスタック (VPCStack、BastionStack、または RDSNS という名前がついていないスタック) を選択します。
3. [削除] を選択します。
4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudFormation でのスタックの削除方法の詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation コンソールでのスタックの削除」を参照してください。

(オプション) DB インスタンスを Lambda 関数に接続する

RDS for SQL Server DB インスタンスを Lambda サーバーレスコンピューティングリソースに接続することもできます。Lambda 関数を使用すると、インフラストラクチャをプロビジョニングしたり

管理したりせずにコードを実行できます。Lambda 関数を使用すると、1 日に 数十件のイベントから 1 秒間に数百件のイベントまで、あらゆる規模のコード実行リクエストに自動的に応答することもできます。詳細については、「Lambda 関数と DB インスタンスを自動的に接続する」を参照してください。

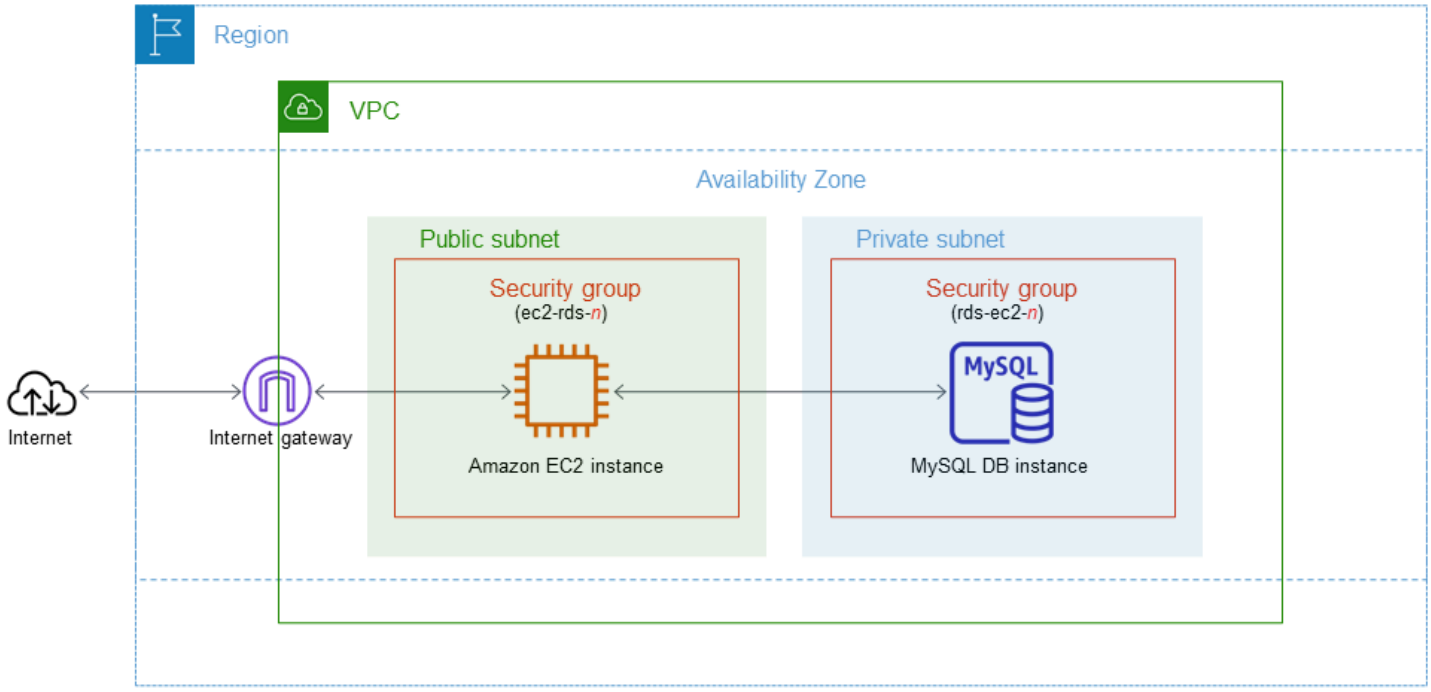
MySQL DB インスタンスの作成と接続

このチュートリアルでは、EC2 インスタンスと RDS for MySQL DB インスタンスを作成します。このチュートリアルでは、標準の MySQL クライアントを使用して EC2 インスタンスから DB インスタンスにアクセスする方法を示します。ベストプラクティスとして、このチュートリアルでは、プライベート DB インスタンスを仮想プライベートクラウド (VPC) に作成します。ほとんどの場合、EC2 インスタンスなど、同じ VPC 内の他のリソースは DB インスタンスにアクセスできますが、VPC 外部のリソースはアクセスできません。

チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。1 つのアベイラビリティーゾーンで、EC2 インスタンスはパブリックサブネットにあり、DB インスタンスはプライベートサブネットにあります。

> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

次の図は、チュートリアルが完了した時点の設定を示しています。



このチュートリアルでは、次のいずれかの方法を使用してリソースを作成できます。

1. AWS マネジメントコンソール を使用する -「MySQL DB インスタンスを作成する」と「EC2 インスタンスの作成」
2. CloudFormation を使用してデータベースインスタンスと EC2 インスタンスを作成する - (オプション) CloudFormation を使用して VPC、EC2 インスタンス、MySQL インスタンスを作成する

最初の方法では、[簡単作成] を使用して、AWS マネジメントコンソール でプライベート MySQL DB インスタンスを作成します。ここでは、DB エンジンタイプ、DB インスタンスサイズ、DB インスタンス識別子のみを指定します。[Easy create (簡易作成)] では、他の設定オプションのデフォルト設定を使用します。

代わりに [標準作成] を使用する場合は、DB インスタンスの作成時にさらに多くの設定オプションを指定できます。このようなオプションには、可用性、セキュリティ、バックアップ、メンテナンスの設定があります。パブリック DB インスタンスを作成するには、[標準作成] を使用する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

トピック

- 前提条件
- EC2 インスタンスの作成
- MySQL DB インスタンスを作成する
- (オプション) CloudFormation を使用して VPC、EC2 インスタンス、MySQL インスタンスを作成する
- MySQL DB インスタンスに接続する
- EC2 インスタンスと DB インスタンスを削除する
- (オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する
- (オプション) DB インスタンスを Lambda 関数に接続する

前提条件

開始する前に、以下のセクションのステップを完了してください。

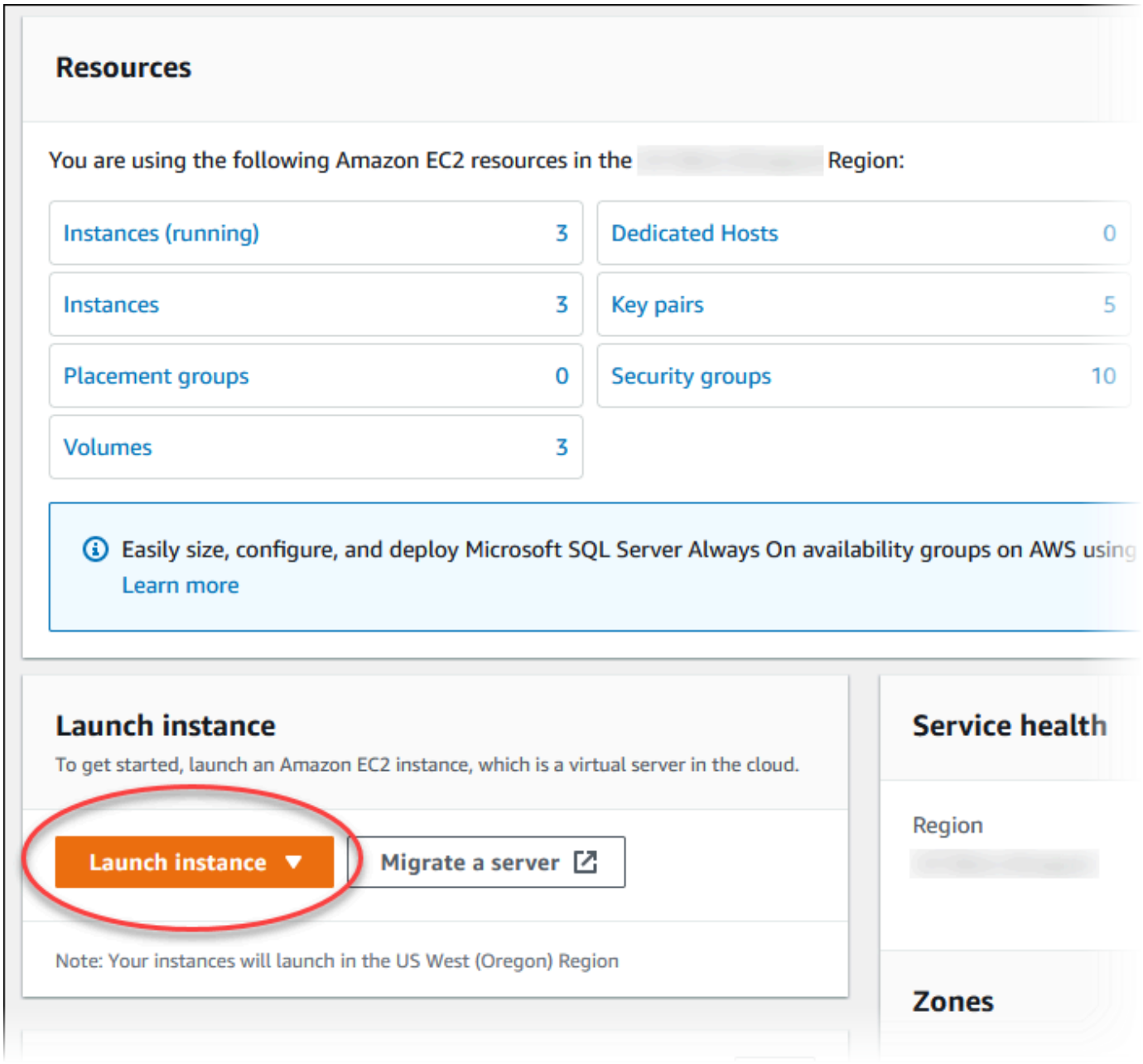
- AWS アカウント へのサインアップ

EC2 インスタンスの作成

データベースへの接続に使用する Amazon EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、EC2 インスタンスを作成する AWS リージョンを選択します。
3. 次の図に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの起動] を選択します。

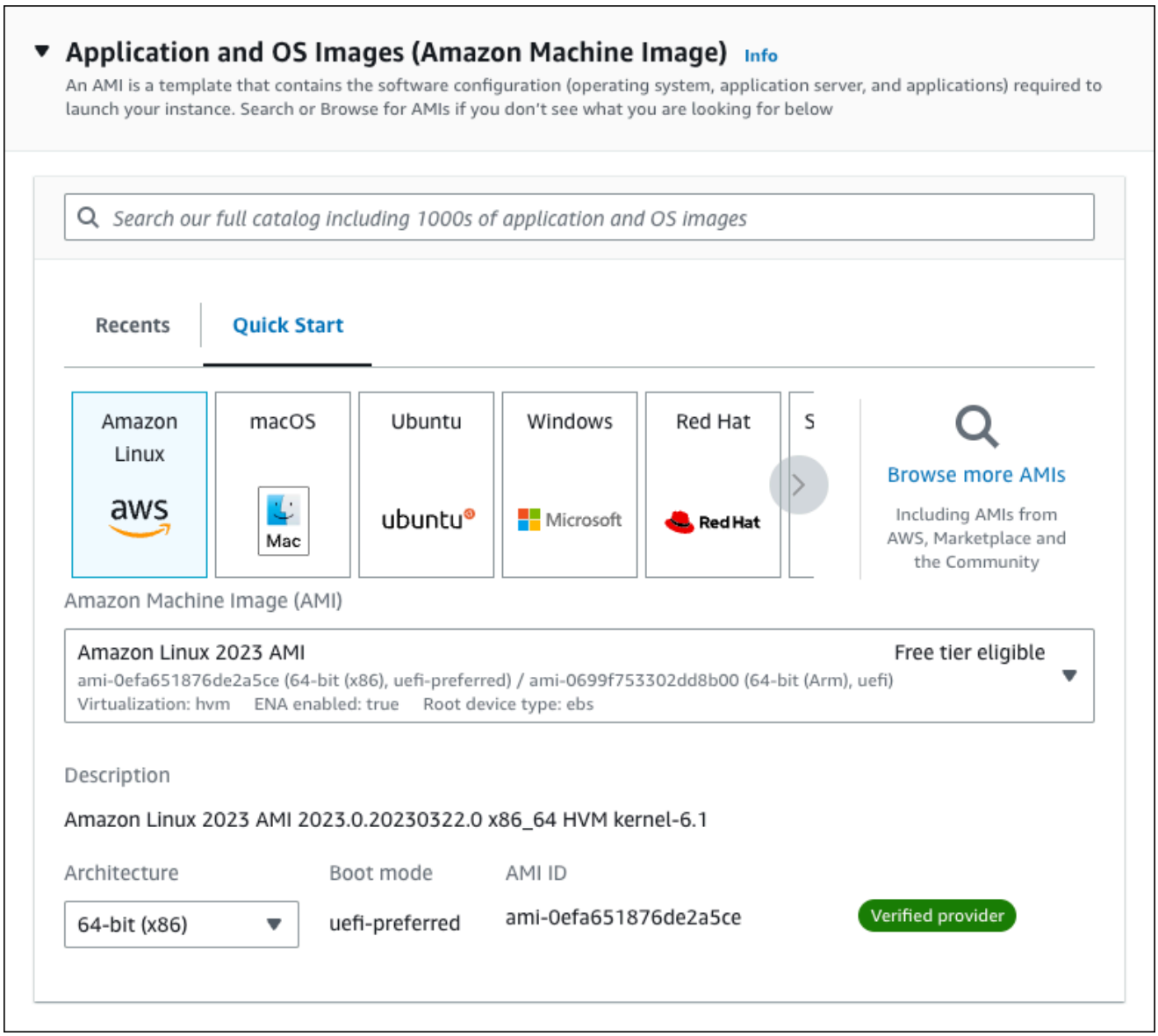


[インスタンスを起動] ページが開きます。

4. [インスタンスを起動] ページで次の設定を選択します。

   a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`ec2-database-connect`** と入力します。

   b. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[Amazon Linux] を選択し、[Amazon Linux 2023 AMI] を選択します。他の選択肢は、デフォルトの選択のままにします。



   c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

   d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するに

は、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

新しいキーペアの作成については、「Amazon EC2 ユーザーガイド」の「キーペアの作成」を参照してください。

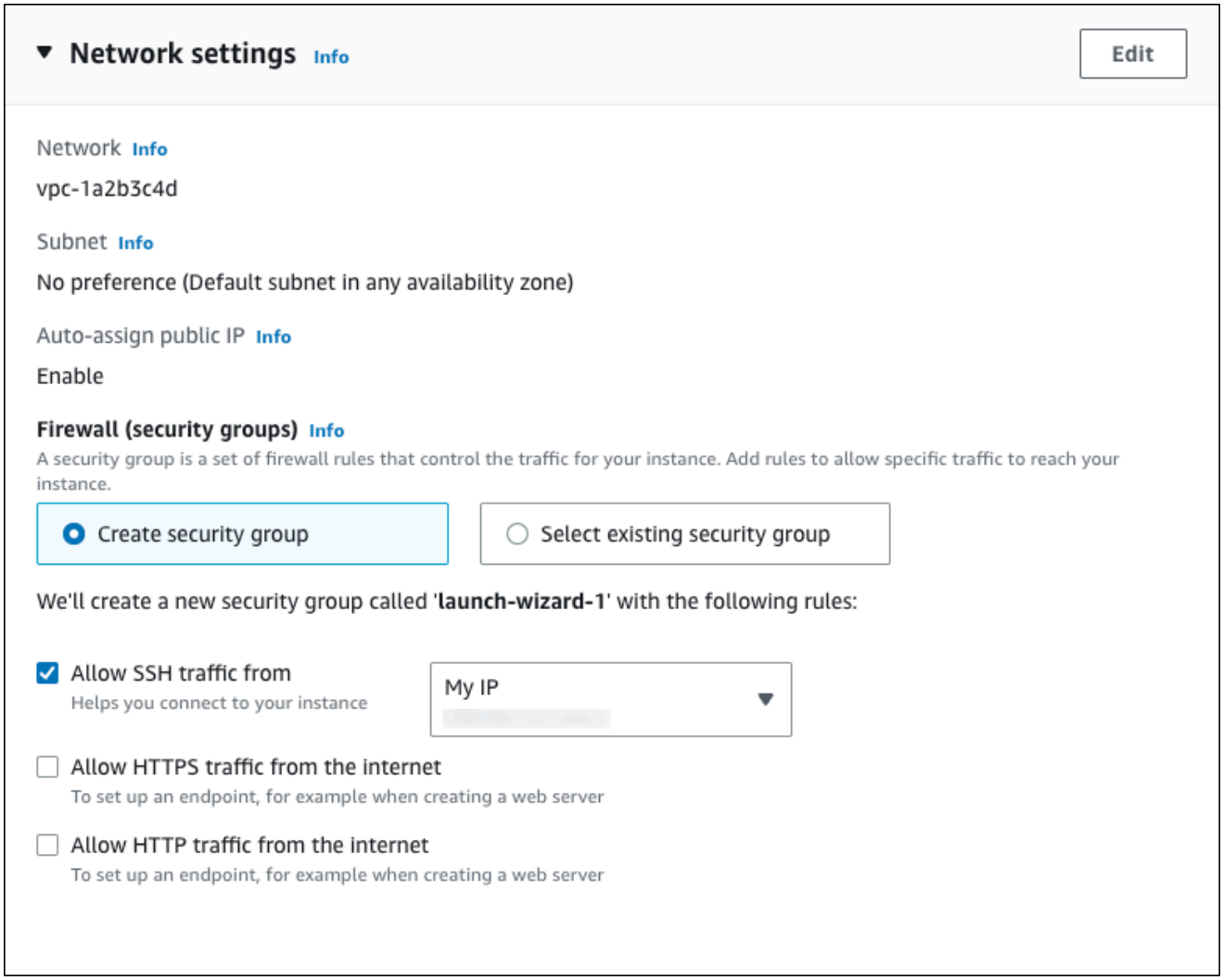
e. ネットワーク設定の [SSH トラフィックを許可] で、EC2 インスタンスへの SSH 接続のソースを選択します。

表示された IP アドレスが SSH 接続に適している場合は、[My IP] (マイ IP)を選択できます。それ以外の場合は、Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は 192.0.2.1/32 です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

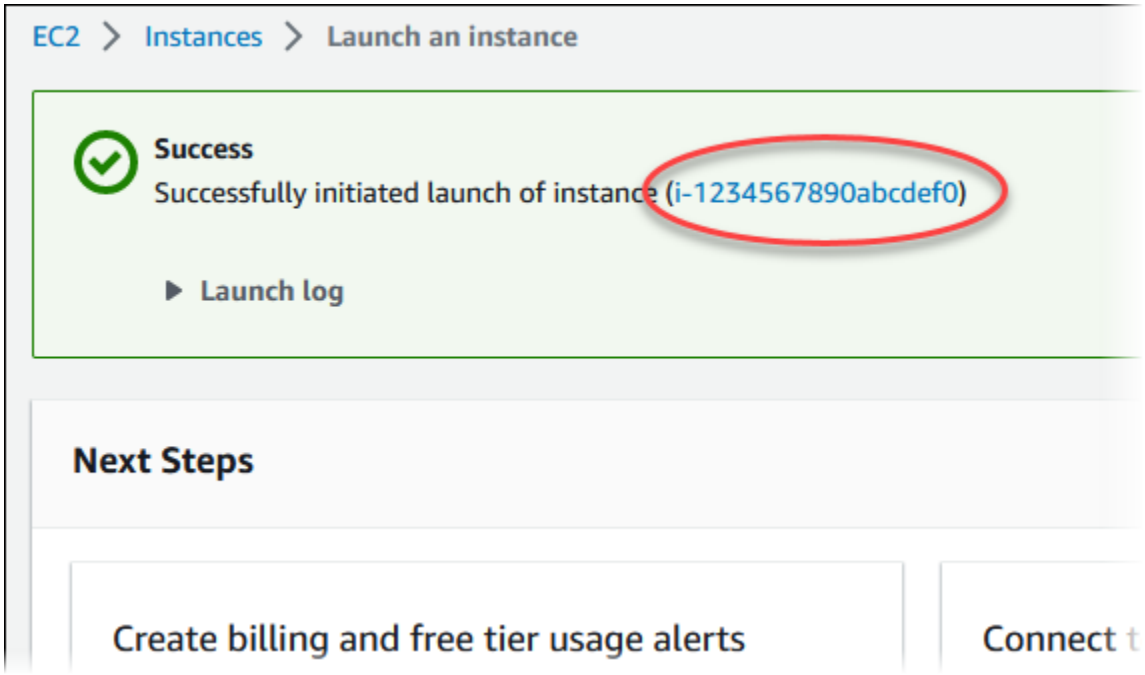
以下のイメージは、[ネットワーク設定] セクションの例を示しています。



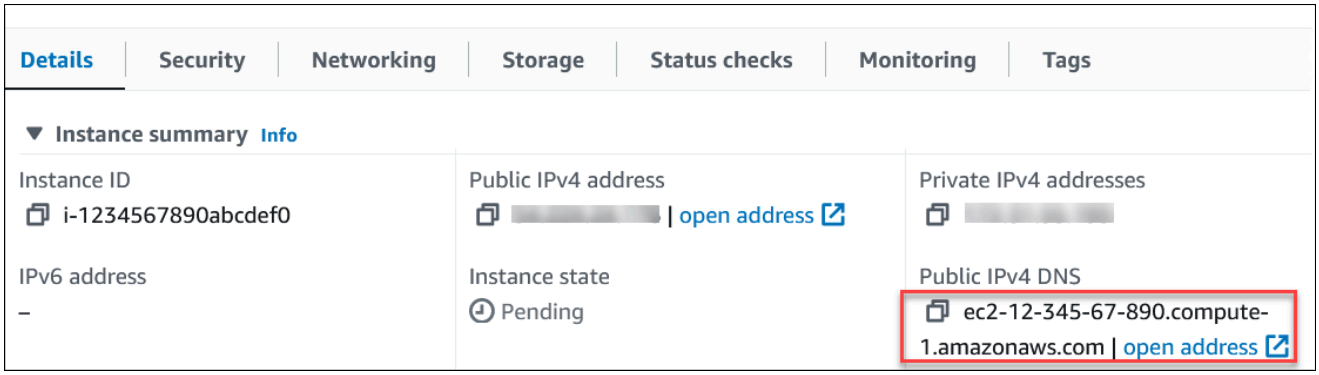
f. 残りのセクションをデフォルト値のままにします。

g. [概要] パネルで、EC2 インスタンス設定の概要を確認し、準備ができたら、[インスタンスの起動] を選択します。

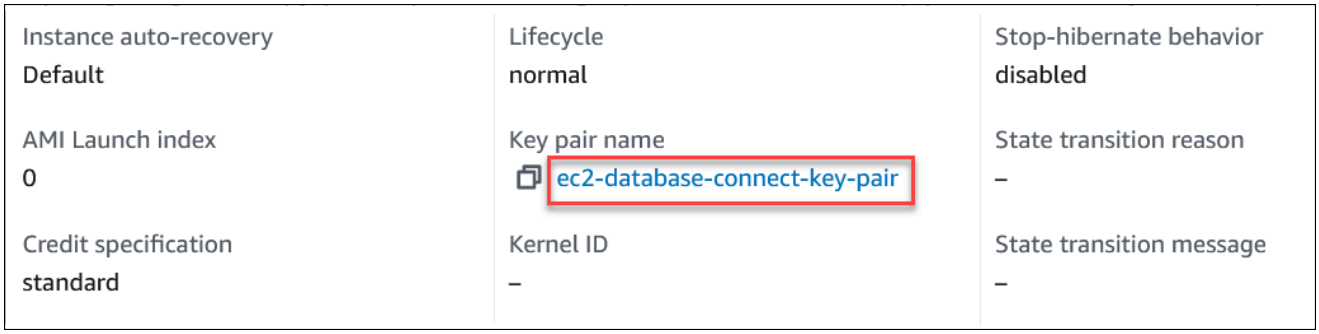
5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開き、EC2 インスタンスを選択します。
7. [詳細] タブで、SSH を使用して接続するときに必要な次の値を書き留めます。
   a. [インスタンスの概要] で、[パブリック IPv4 DNS] の値を書き留めます。



   b. [インスタンスの詳細] で、[キーペア名] の値を書き留めます。



8. EC2 インスタンスの [インスタンス状態] が [実行中] になるまで待ってから、続行します。

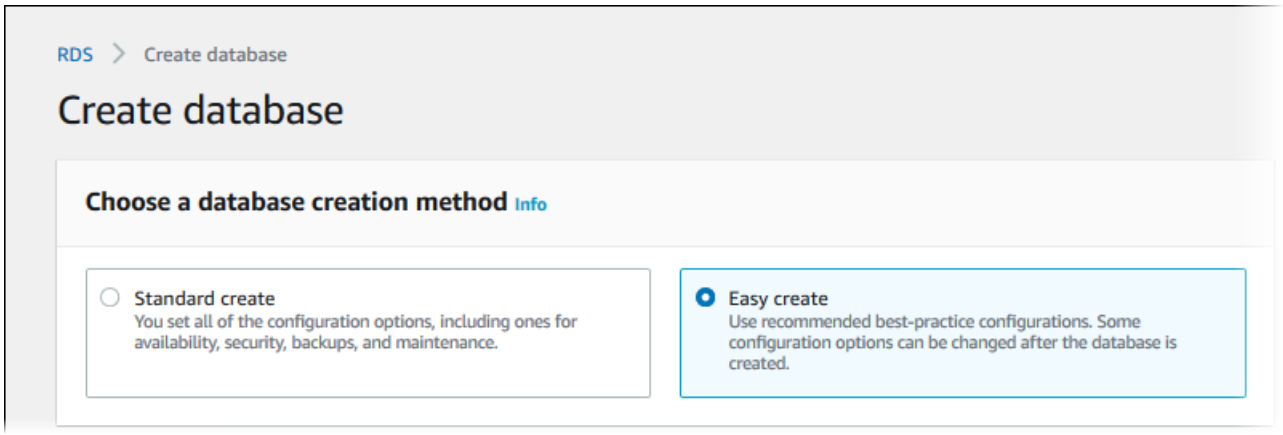
MySQL DB インスタンスを作成する

Amazon RDS の基本的な構成要素は DB インスタンスです。これは、MySQL データベースを実行する環境です。

この例では、[簡易作成] を使用して、db.t3.micro DB インスタンスクラスで MySQL データベースエンジンを実行する DB インスタンスを作成します。

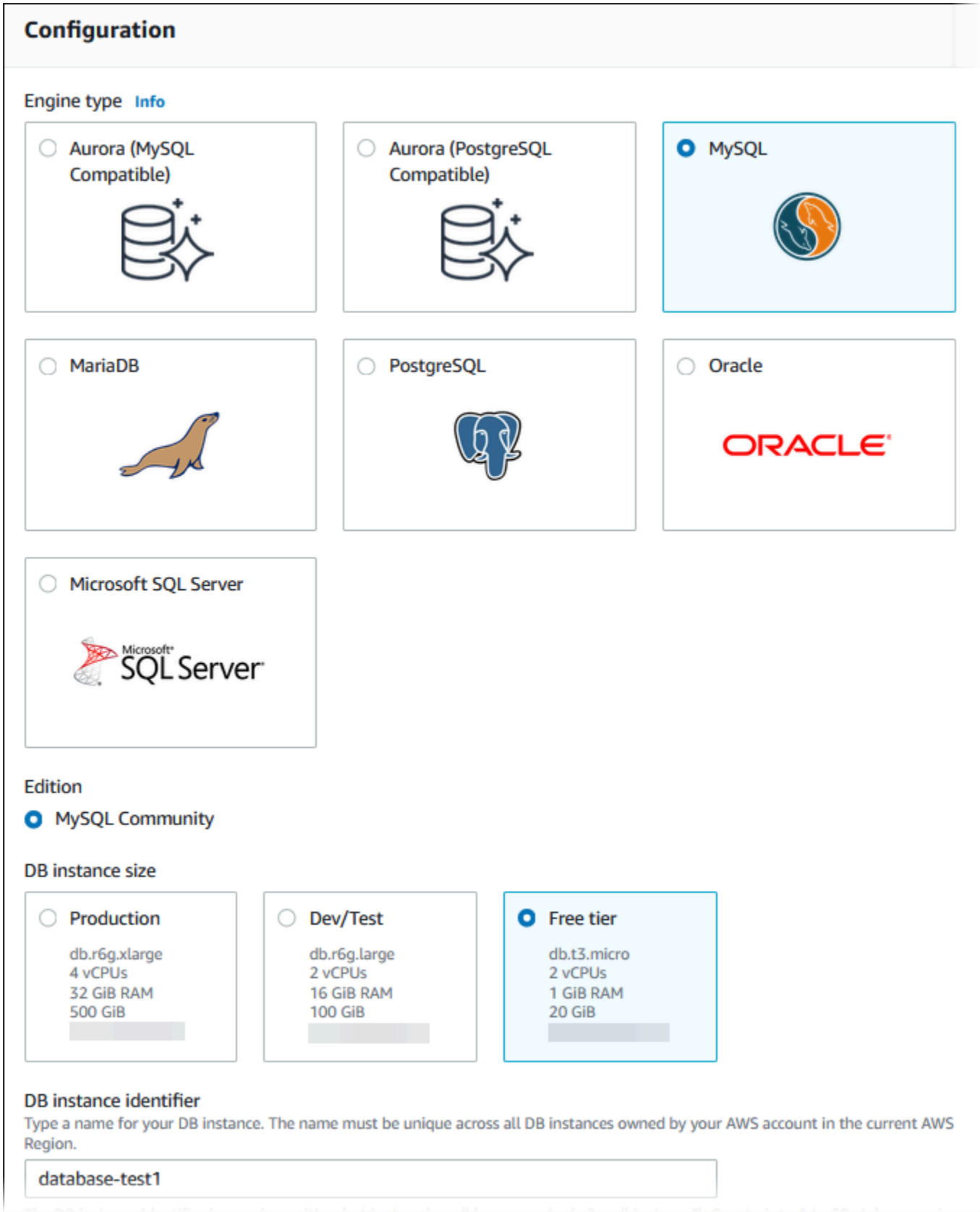
簡易作成で MySQL DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、以前 EC2 インスタンスに使用した AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [Create database (データベースの作成)] を選択し、[Easy create (簡易作成)] が選択されていることを確認します。



5. [設定] で、[MySQL] を選択します。
6. [DB インスタンスサイズ] で、[無料利用枠] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。
7. [DB instance identifier] (DB インスタンス識別子) に**`database-test1`** と入力します。
8. [マスターユーザー名] に、マスターユーザーの名前を入力するか、デフォルト名のままにします。

[データベースの作成] ページは、次のイメージのようになります。無料プランアカウントの場合、[無料利用枠] が表示されます。有料プランアカウントの場合、[サンドボックス] が表示されます。

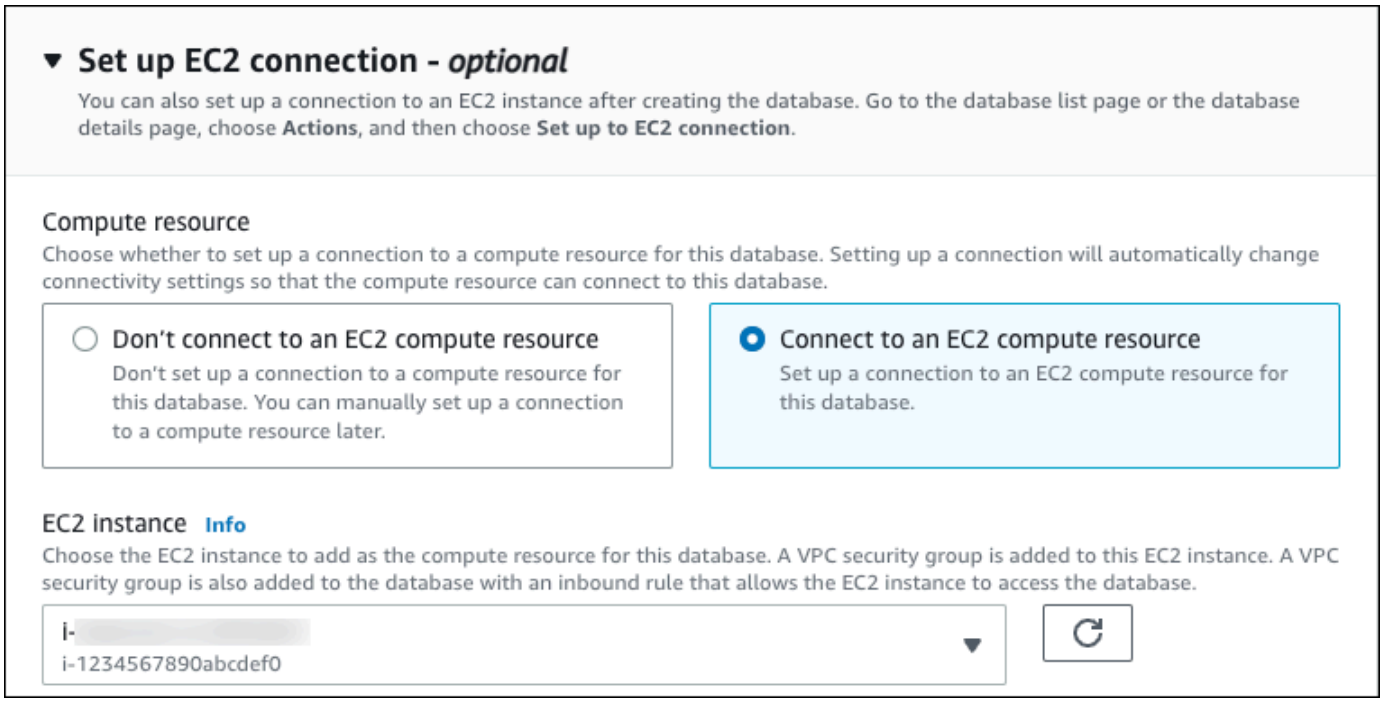

Configuration
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
Edition
MySQL Community
DB instance size
Production
db.r6g.xlarge
4 vCPUs
32 GiB RAM
500 GiB
Dev/Test
db.r6g.large
2 vCPUs
16 GiB RAM
100 GiB
Free tier
db.t3.micro
2 vCPUs
1 GiB RAM
20 GiB
DB instance identifier
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
database-test1

9. DB インスタンス用に自動生成されたマスターパスワードを使用するには、[パスワードの自動生成] を選択します。

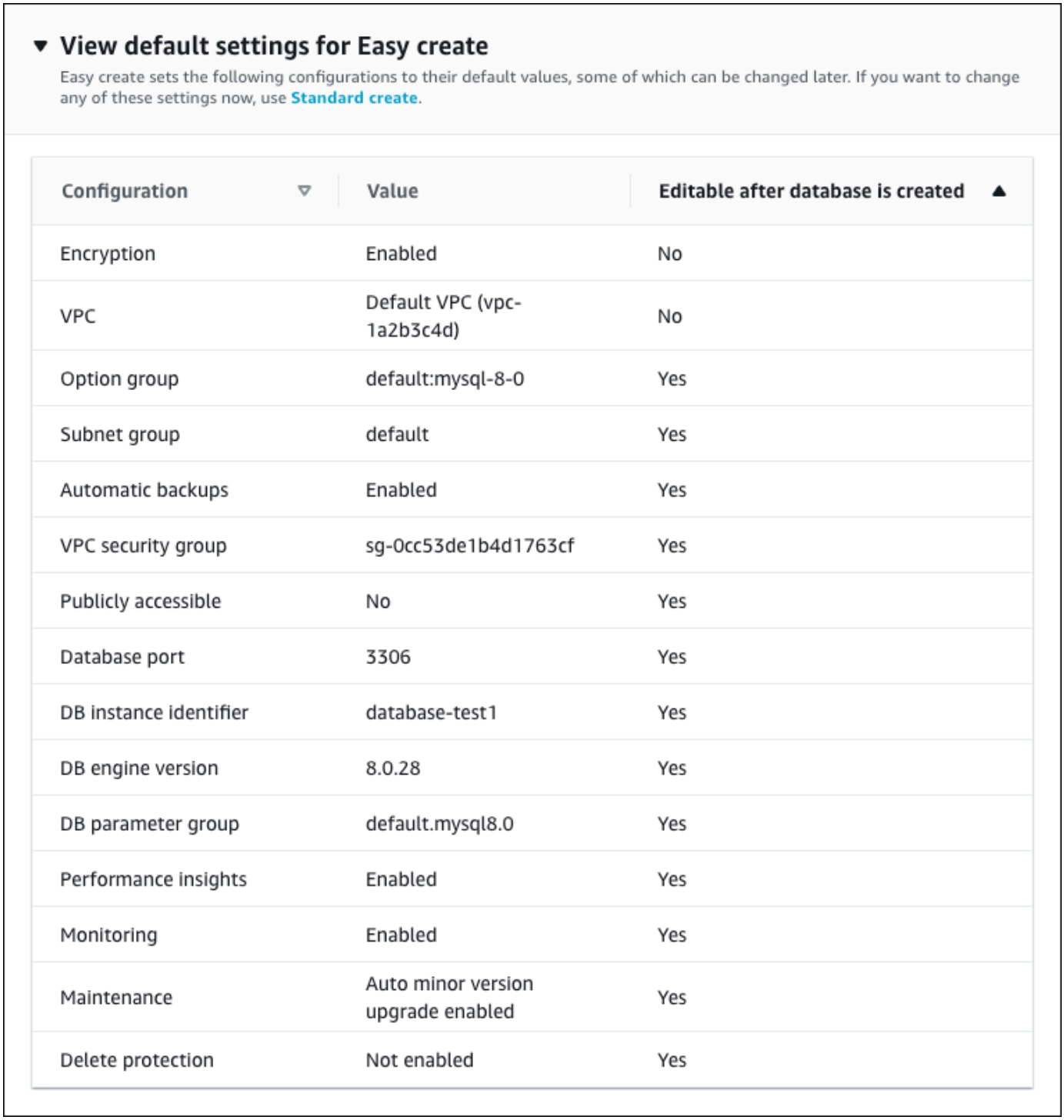
   マスターパスワードを入力するには、[パスワードの自動生成] チェックボックスをオフにして、[マスターパスワード] と [パスワードの確認] に同じパスワードを入力します。

10. 以前に作成した EC2 インスタンスとの接続をセットアップするには、[EC2 接続のセットアップ - オプション] を開きます。

    [EC2 コンピューティングリソースに接続] を選択します。以前に作成した EC2 インスタンスを選択します。



11. (オプション) [View default settings for Easy create (簡易作成のデフォルト設定を表示)] を開きます。



▼ View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use Standard create.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:mysql-8-0	Yes
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-0cc53de1b4d1763cf	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB instance identifier	database-test1	Yes
DB engine version	8.0.28	Yes
DB parameter group	default.mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[Easy Create (簡易作成)] で使用されるデフォルト設定を調べることができます。[データベース作成後に編集可能] 列には、データベース作成後に変更できるオプションが表示されます。

- その列の設定に [いいえ] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成できます。

- その列の設定に [はい] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成するか、DB インスタンスの作成後に設定を変更できます。

12. [データベースの作成] を選択します。

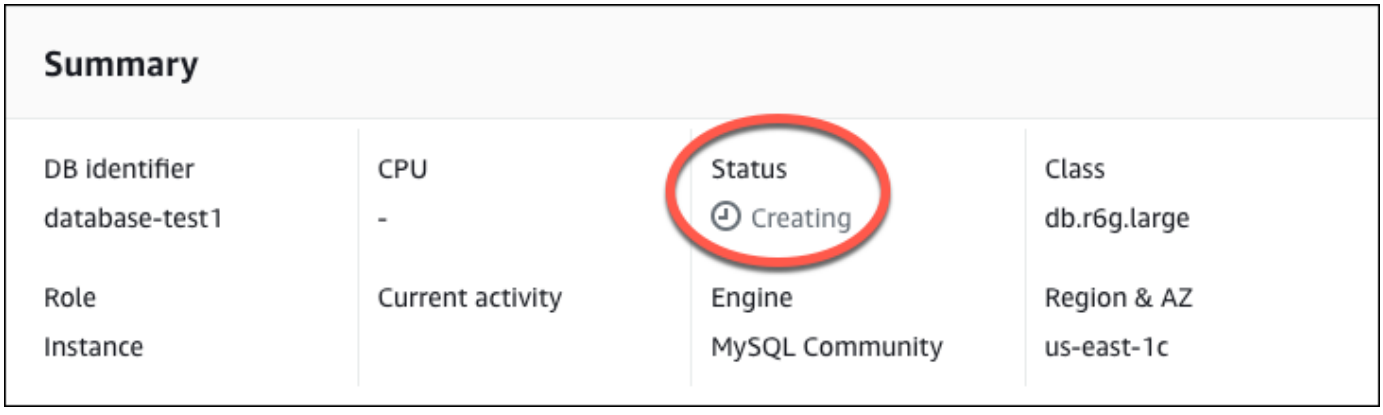
    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    表示されるユーザー名とパスワードを使用して、マスターユーザーとして DB インスタンスに接続できます。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。
    > DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンスを変更することができます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

13. [データベース] リストで、新しい MySQL DB インスタンスの名前を選択し、詳細を表示します。

    DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは [作成中] のままです。



    ステータスが [Available] (利用可能) に変わったら、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しいインスタンスを使用できるようになるまで最長 20 分かかることがあります。

(オプション) CloudFormation を使用して VPC、EC2 インスタンス、MySQL インスタンスを作成する

コンソールを使用して VPC、EC2 インスタンス、MySQL DB インスタンスを作成する代わりに、CloudFormation を使用して Infrastructure as Code として取り扱うことで、AWS リソースをプロビジョンできます。AWS リソースをより小さく管理しやすい単位に整理するには、CloudFormation のネストされたスタック機能を使用できます。詳細については、「CloudFormation コンソールでのスタックの作成」と「ネストされたスタックの操作」を参照してください。

> **Important**
>
> CloudFormation は無料ですが、CloudFormation が作成するリソースは実動のものです。これらのリソースを終了するまで、標準使用料が発生します。詳細については、「RDS for MySQL の料金」を参照してください。

CloudFormation コンソールを使用してリソースを作成するには、以下のステップを実行します。

- CloudFormation テンプレートをダウンロードする
- CloudFormation を使用してリソースを設定する

CloudFormation テンプレートをダウンロードする

CloudFormation テンプレートは、スタックに作成するリソースに関する設定情報が含まれる JSON または YAML のテキストファイルです。このテンプレートは、RDS インスタンスとともに VPC と踏み台ホストも作成します。

テンプレートファイルをダウンロードするには、次のリンク、MySQL CloudFormation template を開きます。

この Github ページで、[Download raw file] ボタンをクリックしてテンプレートの YAML ファイルを保存します。

CloudFormation を使用してリソースを設定する

> **Note**
>
> このプロセスを開始する前に、AWS アカウント に EC2 インスタンスのキーペアがあることを確認してください。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

CloudFormation テンプレートを使用する場合は、適切なパラメータを選択して、リソースが正しく作成されていることを確認する必要があります。以下のステップを実行します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [テンプレートの指定] セクションで、[コンピュータからテンプレートファイルをアップロード] を選択し、[次へ] をクリックします。
4. [スタックの詳細を指定] ページで、次のパラメータを設定します。
   a. [スタック名] は [MySQLTestStack] に設定します。
   b. [パラメータ] で、3 つのアベイラビリティーゾーンを選択して、[アベイラビリティーゾーン] を設定します。
   c. [Linux 踏み台ホスト設定] で、[キー名] に EC2 インスタンスにログインするキーペアを選択します。
   d. [Linux 踏み台ホスト設定] で、[許可された IP 範囲] を IP アドレスに設定します。Secure Shell (SSH) を使用して VPC 内の EC2 インスタンスに接続するには、https://checkip.amazonaws.com のサービスを使用してパブリック IP アドレスを確認します。IP アドレスの例は 192.0.2.1/32 です。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

e. [Database General configuration] で、[データベースインスタンスクラス] を [db.t3.micro] に設定します。

f. [データベース名] を **`database-test1`** に設定します。

g. [データベースマスターユーザー名] には、PDB のマスターユーザー名を入力します。

h. このチュートリアルでは、[Secrets Manager で DB マスターユーザーパスワードを管理] を `false` に設定します。

i. [データベースパスワード] には、任意のパスワードを設定します。このパスワードは、チュートリアルの他の手順のために覚えておいてください。

j. [Database Storage configuration] で、[Database storage type] を [gp2] に設定します。

k. [Database Monitoring configuration] で、[Enable RDS Performance Insights] を false に設定します。

l. その他の設定はすべてデフォルトの値のままにします。[次へ] をクリックして続行します。

5. [スタックオプションの設定] ページでは、すべてのデフォルトオプションをそのまま使用します。[次へ] をクリックして続行します。

6. [スタックの確認] ページで、データベースと Linux 踏み台ホストのオプションを確認した後、[送信] をクリックします。

スタックの作成プロセスが完了したら、BastionStack と RDSNS という名前のスタックを確認して、データベースへの接続に必要な情報をメモします。詳細については、「AWS マネジメントコンソール での CloudFormation スタックデータとリソースの表示」を参照してください。

MySQL DB インスタンスに接続する

標準の SQL クライアントアプリケーションを使用して、DB インスタンスに接続できます。この例では、mysql コマンドラインクライアントを使用して、MySQL DB インスタンスに接続します。

MySQL DB インスタンスに接続するには

1. DB インスタンスのエンドポイント (DNS 名) とポート番号を見つけます。

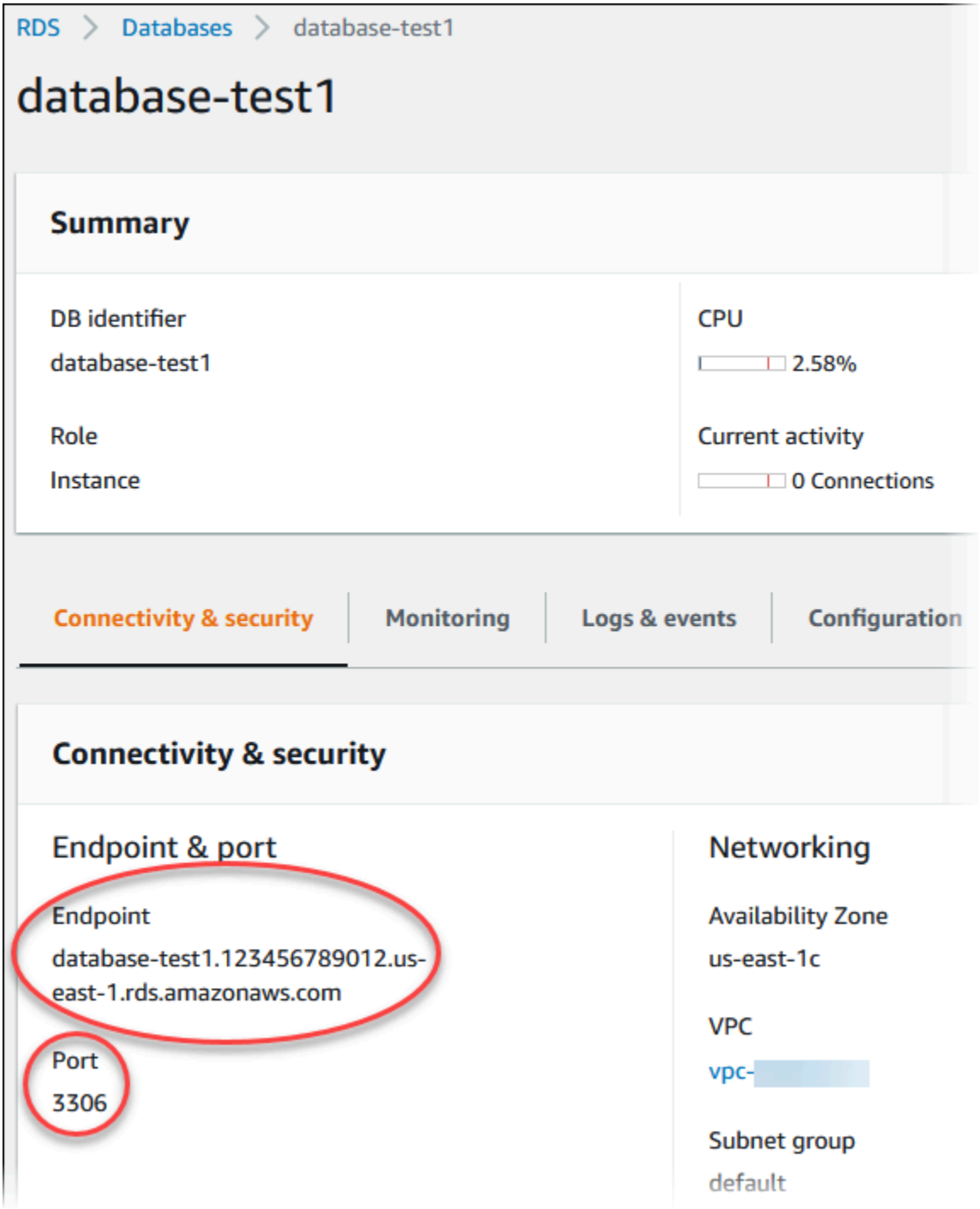
   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[データベース] を選択します。

   d. MySQL DB インスタンスの名前を選択して詳細を表示します。

e. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



2. 「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。

SSH を使用して EC2 インスタンスに接続することをお勧めします。SSH クライアントユーティリティが Windows、Linux、または Mac にインストールされている場合は、次のコマンド形式でインスタンスに接続できます。

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

例えば、ec2-database-connect-key-pair.pem が Linux の /dir1 に保存されていて、EC2 インスタンスのパブリック IPv4 DNS が ec2-12-345-678-90.compute-1.amazonaws.com であるとします。SSH コマンドは次のようになります。

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 インスタンスのソフトウェアを更新して、最新のバグ修正とセキュリティ更新を入手します。これを行うには、次のコマンドを使用します。

> **Note**
>
> -y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を確認するには、このオプションを省略します。

```
sudo dnf update -y
```

4. MariaDB の mysql コマンドラインクライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install mariadb105
```

5. MySQL DB インスタンスに接続します。例えば、次のコマンドを入力します。このアクションにより、MySQL クライアントを使用して、MySQL DB インスタンスに接続できます。

   endpoint の DB インスタンスの DNS 名 (エンドポイント) を、*admin* で使用したマスターユーザー名に置き換えます。パスワードの入力を求められたときに使用したマスターパスワードを入力します。

```
mysql -h endpoint -P 3306 -u admin -p
```

ユーザーのパスワードを入力すると、次のような出力が表示されます。

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 3082
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

MySQL DB インスタンスへの接続の詳細については、「MySQL DB インスタンスへの接続」を参照してください。DB インスタンスに接続できない場合は、「Amazon RDS DB インスタンスに接続できない」を参照してください。

セキュリティのためには、暗号化された接続を使用することがベストプラクティスです。クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない MySQL 接続を使用します。暗号化された接続の使用については、「MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MySQL DB インスタンスへの接続 (暗号化)」を参照してください。

6. SQL コマンドを実行する。

   例えば、次の SQL コマンドは、現在の日付と時刻を表示します。

```
SELECT CURRENT_TIMESTAMP;
```

EC2 インスタンスと DB インスタンスを削除する

作成したサンプル EC2 インスタンスと DB インスタンスに接続して、探索したら、料金がこれ以上発生しないように、それらを削除します。

CloudFormation を使用してリソースを作成した場合は、このステップをスキップして次のステップに進みます。

EC2 インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. EC2 インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Terminate (終了)] を選択してください。

EC2 インスタンスの削除の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。

最終的な DB スナップショットを作成せずに DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除する DB インスタンスを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成] と [自動バックアップの保持] をクリアします。
6. 確認を完了し、[削除] を選択します。

(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する

CloudFormation を使用してリソースを作成した場合は、サンプル EC2 インスタンスと DB インスタンスに接続して確認を済ませた後、それ以上料金が発生しないように、CloudFormation スタックを削除します。

CloudFormation リソースを削除するには

1. CloudFormation コンソールを開きます。
2. CloudFormation コンソールの [スタック] ページで、ルートスタック (VPCStack、BastionStack、または RDSNS という名前がついていないスタック) を選択します。
3. [削除] を選択します。

4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudFormation でのスタックの削除方法の詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation コンソールでのスタックの削除」を参照してください。

(オプション) DB インスタンスを Lambda 関数に接続する

RDS for MySQL DB インスタンスを Lambda サーバーレスコンピューティングリソースに接続することもできます。Lambda 関数を使用すると、インフラストラクチャをプロビジョニングしたり管理したりせずにコードを実行できます。Lambda 関数を使用すると、1 日に 数十件のイベントから 1 秒間に数百件のイベントまで、あらゆる規模のコード実行リクエストに自動的に応答することもできます。詳細については、「Lambda 関数と DB インスタンスを自動的に接続する」を参照してください。

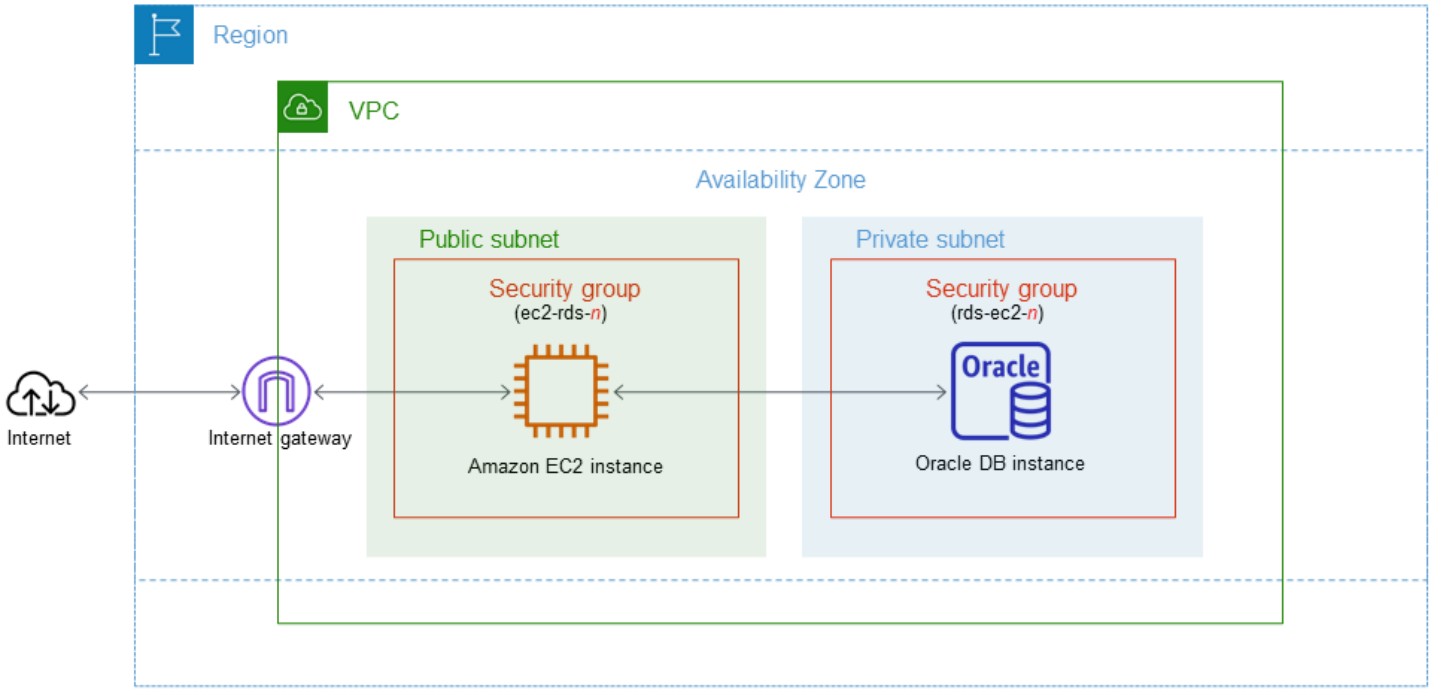
Oracle DB インスタンスを作成して接続する

このチュートリアルでは、EC2 インスタンスと RDS for Oracle DB インスタンスを作成します。このチュートリアルでは、標準の Oracle クライアントを使用して EC2 インスタンスから DB インスタンスにアクセスする方法を示します。ベストプラクティスとして、このチュートリアルでは、プライベート DB インスタンスを仮想プライベートクラウド (VPC) に作成します。ほとんどの場合、EC2 インスタンスなど、同じ VPC 内の他のリソースは DB インスタンスにアクセスできますが、VPC 外部のリソースはアクセスできません。

チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。1 つのアベイラビリティーゾーンで、EC2 インスタンスはパブリックサブネットにあり、DB インスタンスはプライベートサブネットにあります。

> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

次の図は、チュートリアルが完了した時点の設定を示しています。



このチュートリアルでは、次のいずれかの方法を使用してリソースを作成できます。

1. AWS マネジメントコンソール を使用する -「ステップ 2: Oracle DB インスタンスを作成する」と「ステップ 1: EC2 インスタンスを作成する」
2. CloudFormation を使用してデータベースインスタンスと EC2 インスタンスを作成する - (オプション) CloudFormation を使用して VPC、EC2 インスタンス、Oracle DB インスタンスを作成する

最初の方法では、[簡単作成] を使用して、AWS マネジメントコンソール でプライベート Oracle DB インスタンスを作成します。ここでは、DB エンジンタイプ、DB インスタンスサイズ、DB インスタンス識別子のみを指定します。[Easy create (簡易作成)] では、他の設定オプションのデフォルト設定を使用します。

代わりに [標準作成] を使用する場合は、DB インスタンスの作成時にさらに多くの設定オプションを指定できます。このようなオプションには、可用性、セキュリティ、バックアップ、メンテナンスの設定があります。パブリック DB インスタンスを作成するには、[標準作成] を使用する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

トピック

- 前提条件
- ステップ 1: EC2 インスタンスを作成する
- ステップ 2: Oracle DB インスタンスを作成する
- (オプション) CloudFormation を使用して VPC、EC2 インスタンス、Oracle DB インスタンスを作成する
- ステップ 3: SQL クライアントを Oracle DB インスタンスに接続する
- ステップ 4: EC2 インスタンスと DB インスタンスを削除する
- (オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する
- (オプション) DB インスタンスを Lambda 関数に接続する

前提条件

開始する前に、以下のセクションのステップを完了してください。

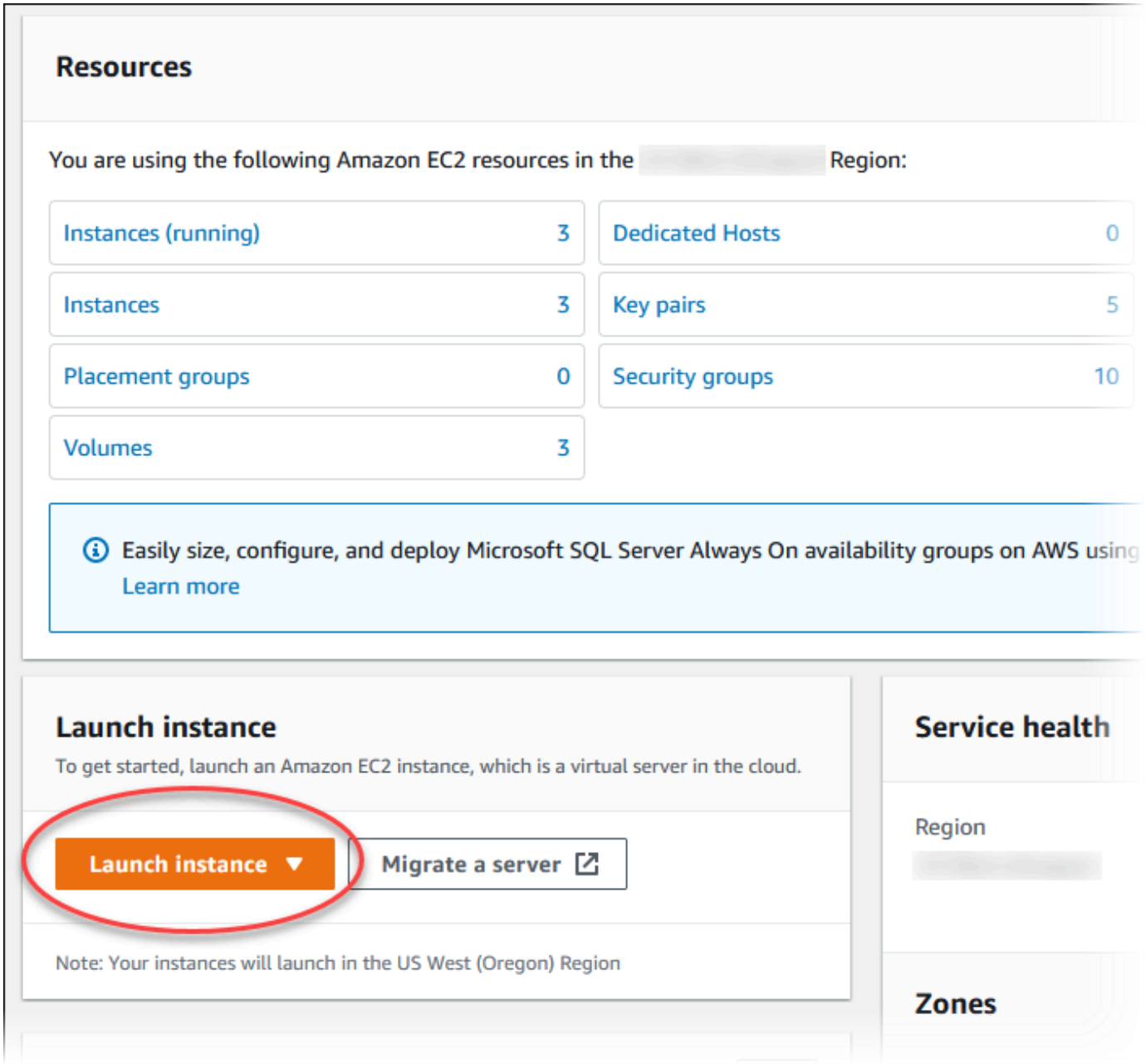
- AWS アカウント へのサインアップ

ステップ 1: EC2 インスタンスを作成する

データベースへの接続に使用する Amazon EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、EC2 インスタンスを作成する AWS リージョンを選択します。
3. 次の図に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの起動] を選択します。

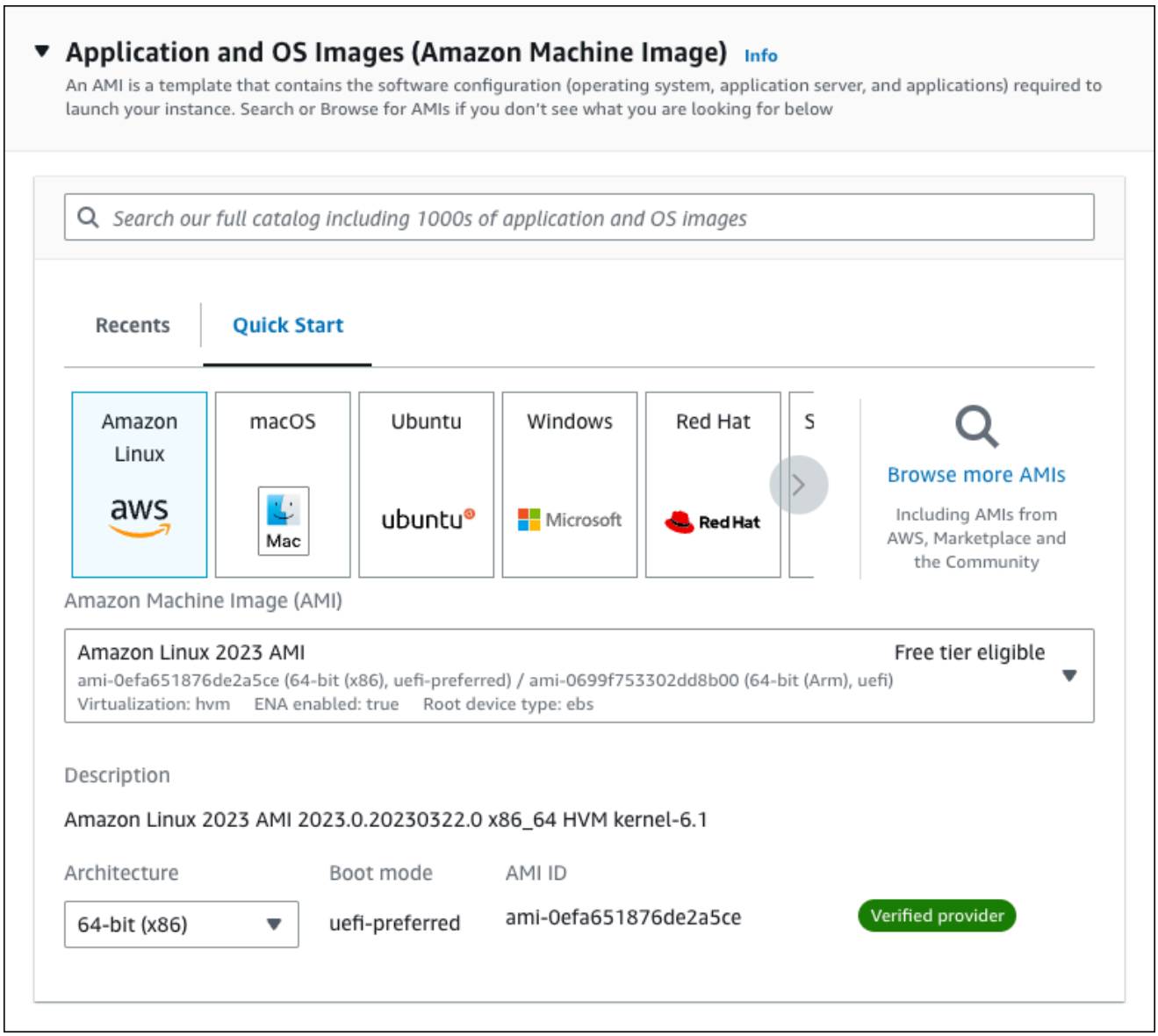


[インスタンスを起動] ページが開きます。

4. [インスタンスを起動] ページで次の設定を選択します。

   a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`ec2-database-connect`** と入力します。

   b. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[Amazon Linux] を選択し、[Amazon Linux 2023 AMI] を選択します。他の選択肢は、デフォルトの選択のままにします。



c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するには、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

   新しいキーペアの作成については、「Amazon EC2 ユーザーガイド」の「キーペアの作成」を参照してください。

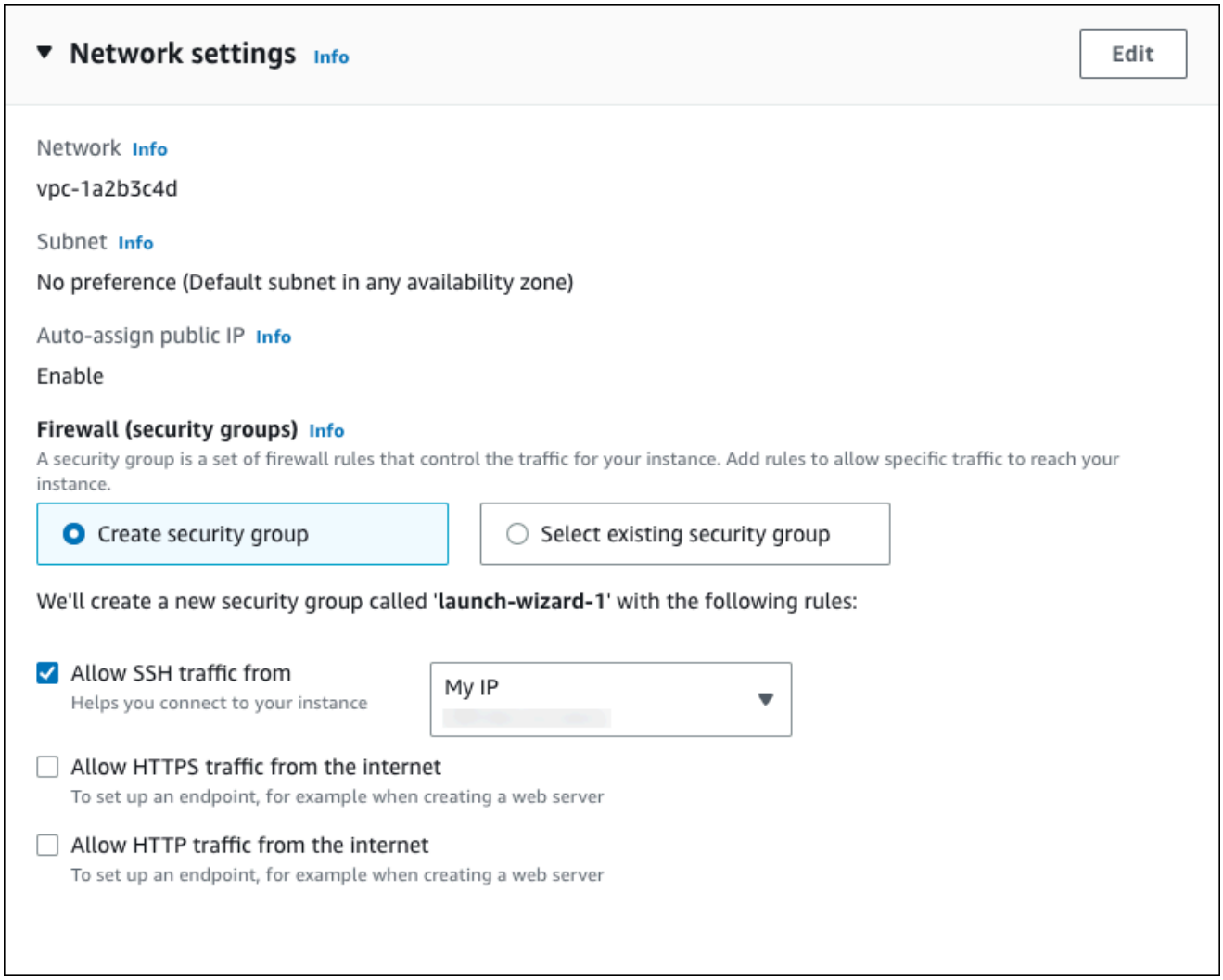
e. ネットワーク設定の [SSH トラフィックを許可] で、EC2 インスタンスへの SSH 接続のソースを選択します。

表示された IP アドレスが SSH 接続に適している場合は、[My IP] (マイ IP)を選択できます。それ以外の場合は、Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は 192.0.2.1/32 です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

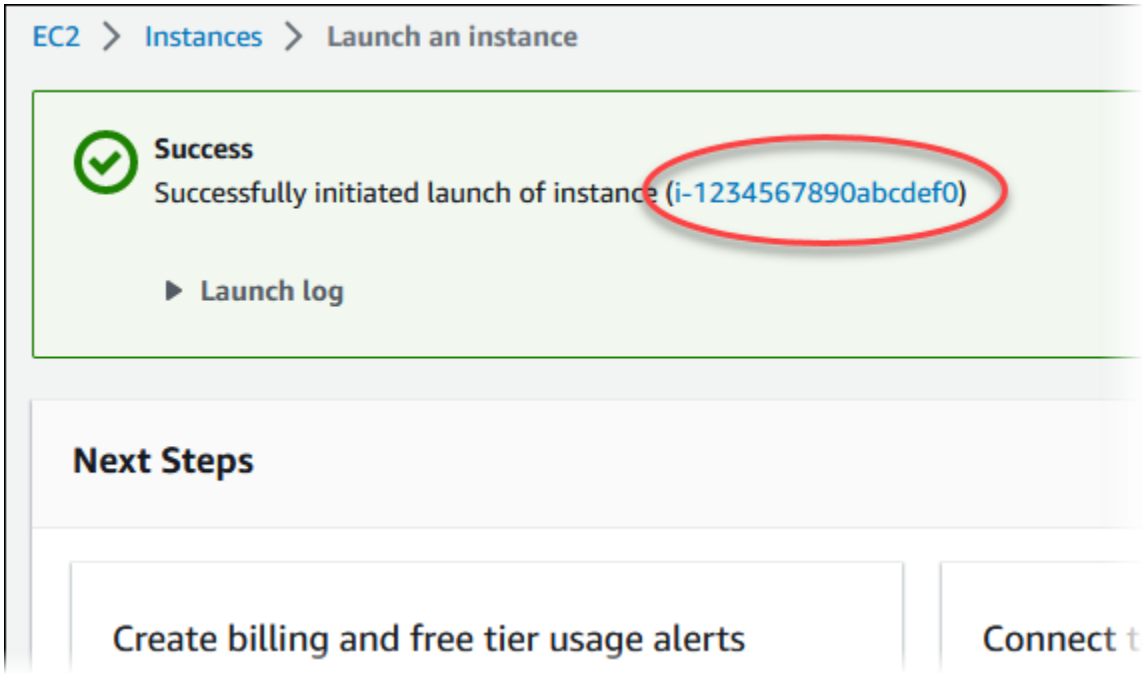
以下のイメージは、[ネットワーク設定] セクションの例を示しています。



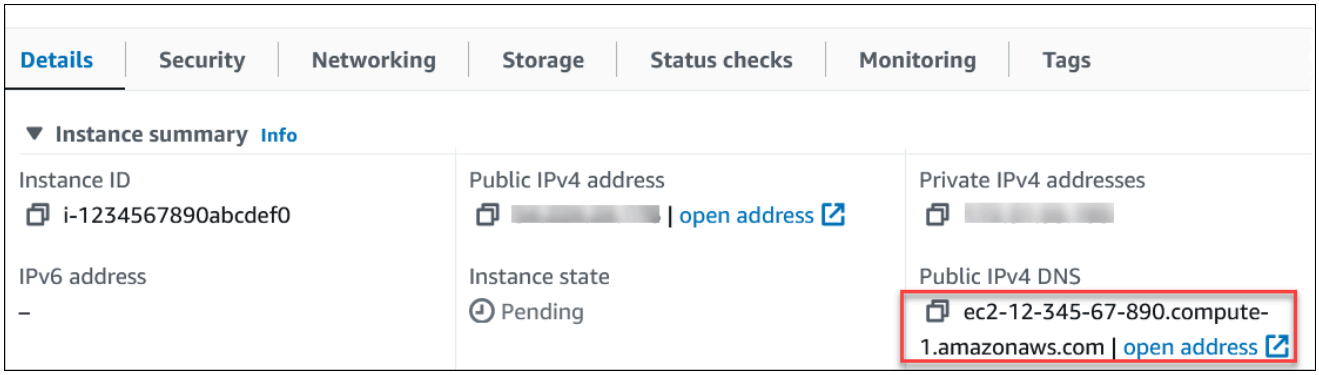
f. 残りのセクションをデフォルト値のままにします。

g. [概要] パネルで、EC2 インスタンス設定の概要を確認し、準備ができたら、[インスタンスの起動] を選択します。

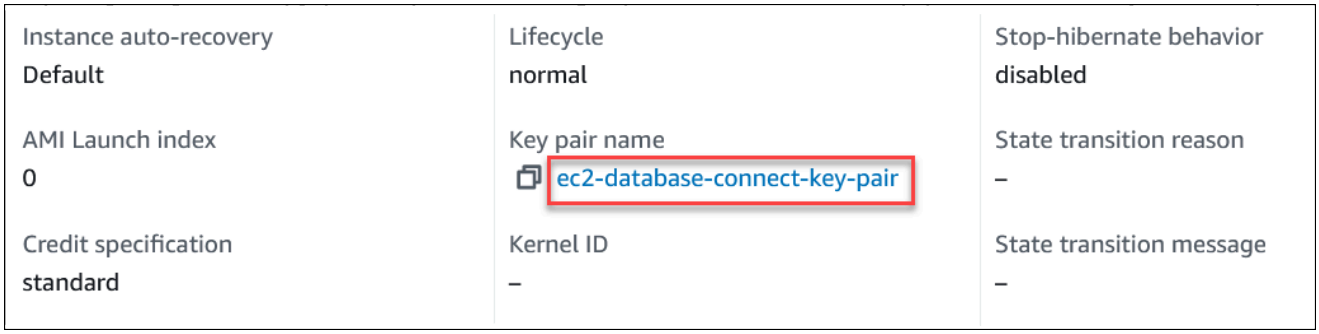
5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開き、EC2 インスタンスを選択します。
7. [詳細] タブで、SSH を使用して接続するときに必要な次の値を書き留めます。
   a. [インスタンスの概要] で、[パブリック IPv4 DNS] の値を書き留めます。



   b. [インスタンスの詳細] で、[キーペア名] の値を書き留めます。



8. EC2 インスタンスの [インスタンス状態] が [実行中] になるまで待ってから、続行します。

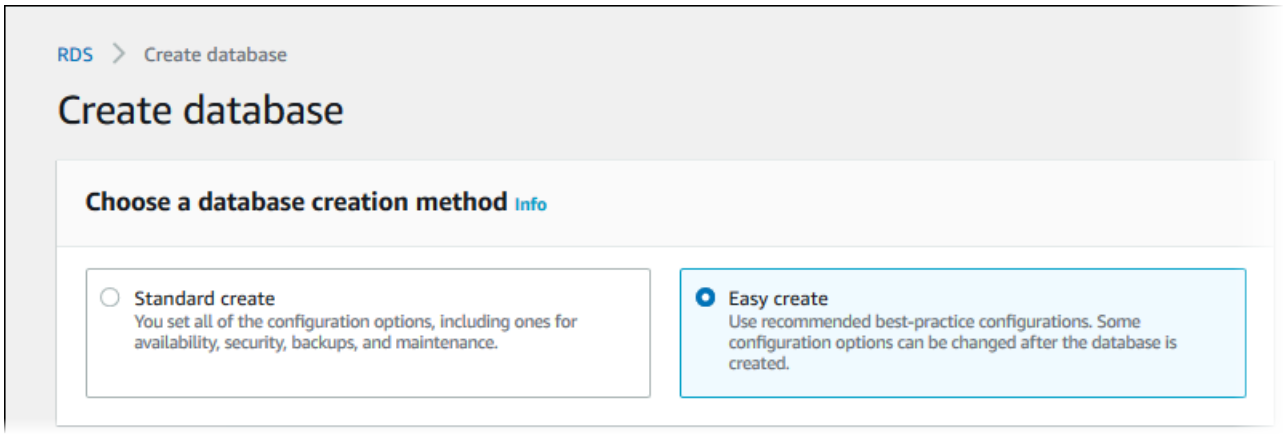
ステップ 2: Oracle DB インスタンスを作成する

Amazon RDS の基本的な構成要素は DB インスタンスです。これは、Oracle データベースを実行する環境です。

この例では、[簡単作成] を使用して、db.m5.large DB インスタンスクラスで Oracle データベースエンジンを実行する DB インスタンスを作成します。

Easy create を有効にして Oracle DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [Create database (データベースの作成)] を選択し、[Easy create (簡易作成)] が選択されていることを確認します。



5. [設定] で、[Oracle] を選択します。
6. [DB インスタンスサイズ] で、[Dev/Test] を選択します。
7. [DB instance identifier] (DB インスタンス識別子) に**`database-test1`** と入力します。
8. [マスターユーザー名] に、マスターユーザーの名前を入力するか、デフォルト名のままにします。

   [データベースの作成] ページは、次のイメージのようになります。

Configuration

Engine type Info

- Aurora (MySQL Compatible)
- Aurora (PostgreSQL Compatible)
- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Microsoft SQL Server

Edition

- Oracle Enterprise Edition
  Affordable and full-featured database management system supporting up to 16 vCPUs.
- Oracle Standard Edition Two
  Affordable and full-featured database management system supporting up to 16 vCPUs. Oracle Database Standard Edition Two is a replacement for Standard Edition and Standard Edition One.

DB instance size

- Production
  db.r5.large
  2 vCPUs
  16 GiB RAM
  500 GiB
- Dev/Test
  db.m5.large
  2 vCPUs
  8 GiB RAM
  100 GiB

DB instance identifier

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-test1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Master username Info

Type a login ID for the master user of your DB instance.

admin

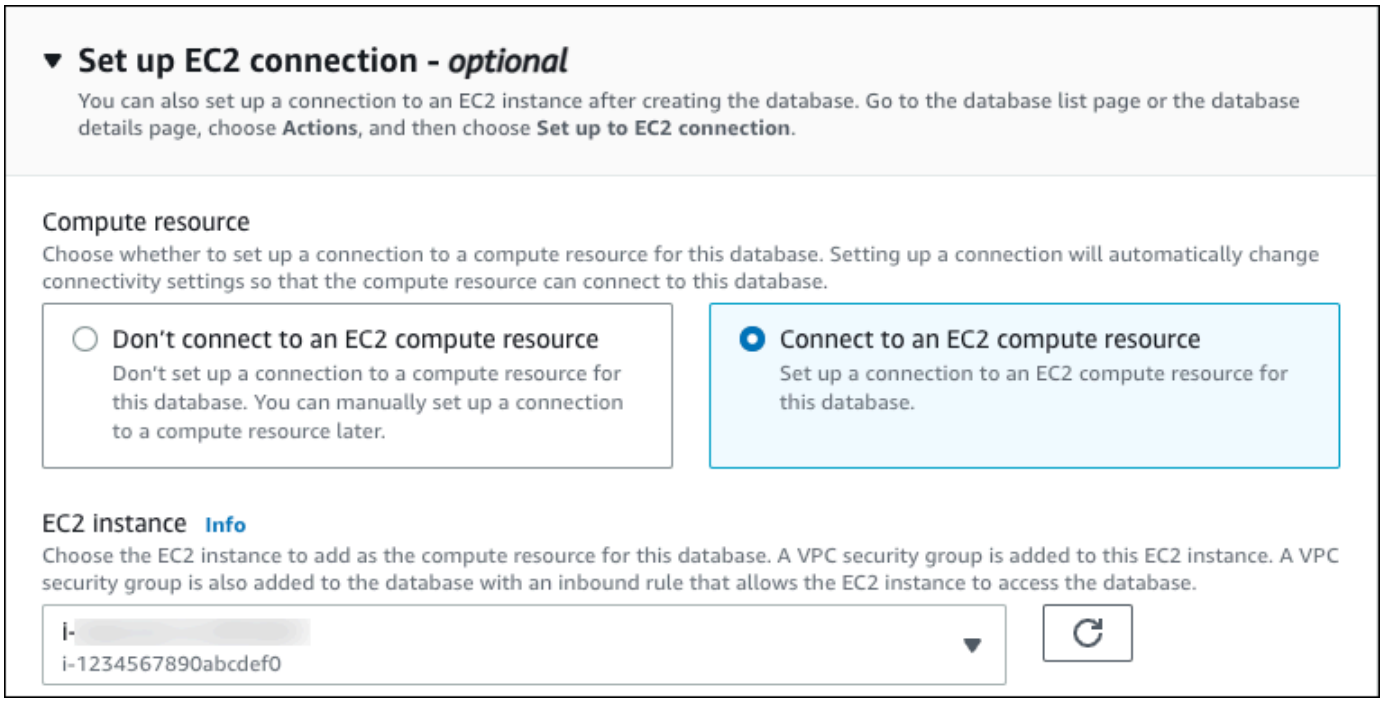
1 to 16 alphanumeric characters. First character must be a letter.

9. DB インスタンス用に自動生成されたマスターパスワードを使用するには、[パスワードの自動生成] を選択します。

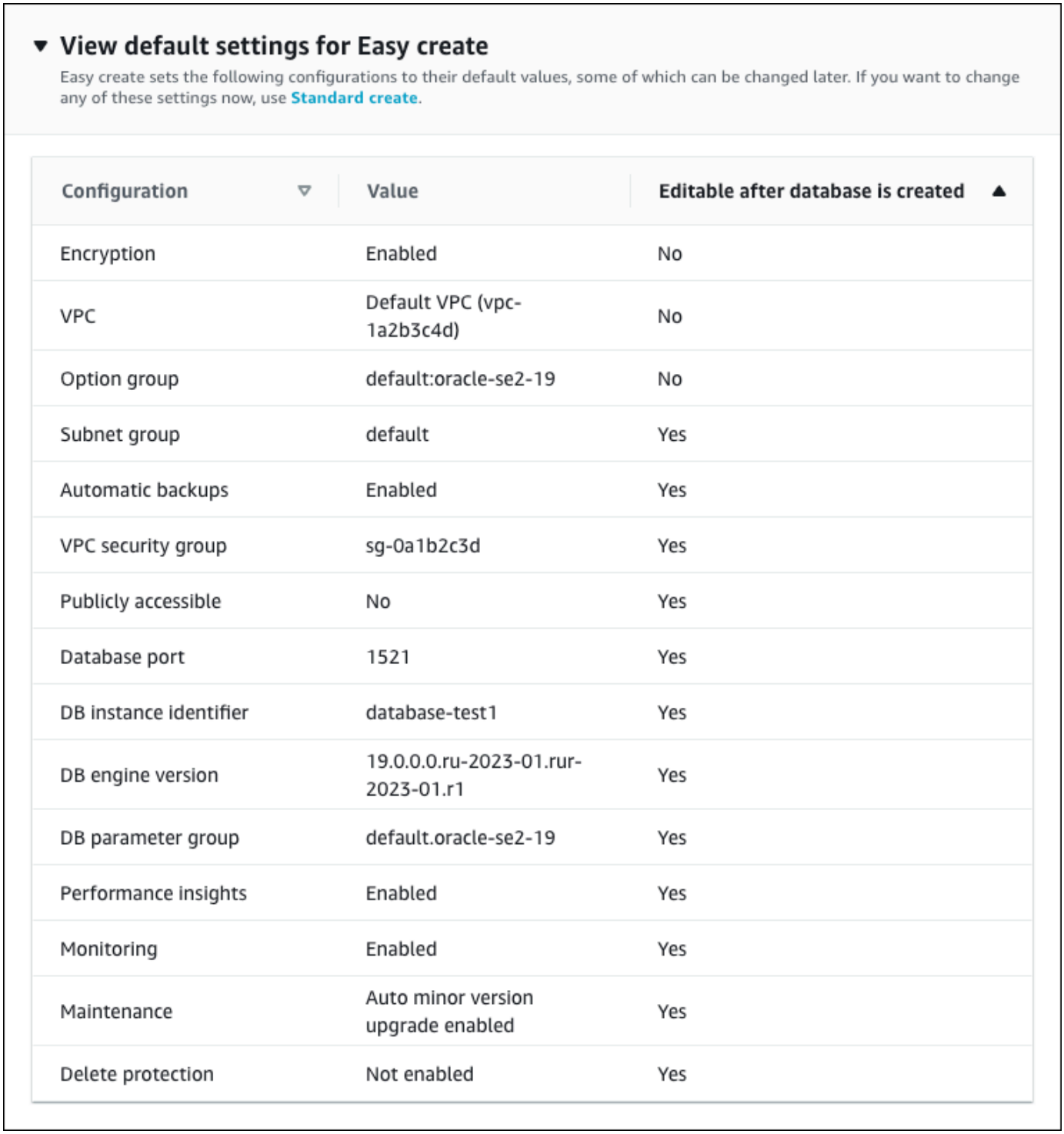
   マスターパスワードを入力するには、[パスワードの自動生成] チェックボックスをオフにして、[マスターパスワード] と [パスワードの確認] に同じパスワードを入力します。

10. 以前に作成した EC2 インスタンスとの接続をセットアップするには、[EC2 接続のセットアップ - オプション] を開きます。

    [EC2 コンピューティングリソースに接続] を選択します。以前に作成した EC2 インスタンスを選択します。



11. [簡易作成のデフォルト設定を表示] を開きます。



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:oracle-se2-19	No
Subnet group	default	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-0a1b2c3d	Yes
Publicly accessible	No	Yes
Database port	1521	Yes
DB instance identifier	database-test1	Yes
DB engine version	19.0.0.0.ru-2023-01.rur-2023-01.r1	Yes
DB parameter group	default.oracle-se2-19	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

[Easy Create (簡易作成)] で使用されるデフォルト設定を調べることができます。[データベース作成後に編集可能] 列には、データベース作成後に変更できるオプションが表示されます。

- その列の設定に [いいえ] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成できます。

- その列の設定に [はい] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成するか、DB インスタンスの作成後に設定を変更できます。

12. [データベースの作成] を選択します。

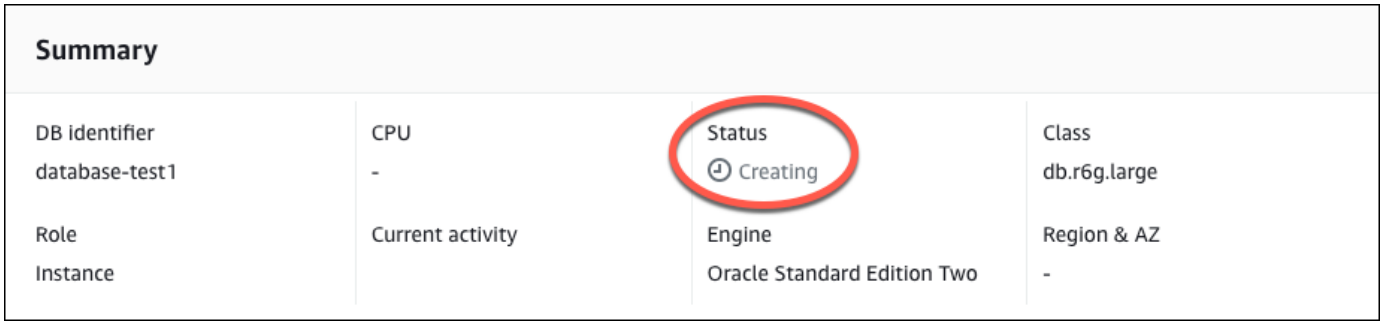
    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    表示されるユーザー名とパスワードを使用して、マスターユーザーとして DB インスタンスに接続できます。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。
    > DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンスを変更することができます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

13. [データベース] リストで、新しい Oracle DB インスタンスの名前を選択し、詳細を表示します。

    DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは [作成中] のままです。



    ステータスが [Available] (利用可能) に変わったら、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しいインスタンスを使用できるようになるまで最長 20 分かかることがあります。DB インスタンスの作成中に、次のステップに進んで EC2 インスタンスを作成できます。

(オプション) CloudFormation を使用して VPC、EC2 インスタンス、Oracle DB インスタンスを作成する

コンソールを使用して VPC、EC2 インスタンス、Oracle DB インスタンスを作成する代わりに、CloudFormation を使用して Infrastructure as Code として取り扱うことで、AWS リソースをプロビジョンできます。AWS リソースをより小さく管理しやすい単位に整理するには、CloudFormation のネストされたスタック機能を使用できます。詳細については、「CloudFormation コンソールでのスタックの作成」と「ネストされたスタックの操作」を参照してください。

> **Important**
>
> CloudFormation は無料ですが、CloudFormation が作成するリソースは実動のものです。これらのリソースを終了するまで、標準使用料が発生します。詳細については、「RDS for Oracle の料金」を参照してください。

CloudFormation コンソールを使用してリソースを作成するには、以下のステップを実行します。

- ステップ 1: CloudFormation テンプレートをダウンロードする
- ステップ 2: CloudFormation を使用してリソースを設定する

CloudFormation テンプレートをダウンロードする

CloudFormation テンプレートは、スタックに作成するリソースに関する設定情報が含まれる JSON または YAML のテキストファイルです。このテンプレートは、RDS インスタンスとともに VPC と踏み台ホストも作成します。

テンプレートファイルをダウンロードするには、次のリンク、Oracle CloudFormation template を開きます。

この Github ページで、[Download raw file] ボタンをクリックしてテンプレートの YAML ファイルを保存します。

CloudFormation を使用してリソースを設定する

> **Note**
>
> このプロセスを開始する前に、AWS アカウント に EC2 インスタンスのキーペアがあることを確認してください。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

CloudFormation テンプレートを使用する場合は、適切なパラメータを選択して、リソースが正しく作成されていることを確認する必要があります。以下のステップを実行します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [テンプレートの指定] セクションで、[コンピュータからテンプレートファイルをアップロード] を選択し、[次へ] をクリックします。
4. [スタックの詳細を指定] ページで、次のパラメータを設定します。
   a. [スタック名] は [OracleTestStack] に設定します。
   b. [パラメータ] で、3 つのアベイラビリティーゾーンを選択して、[アベイラビリティーゾーン] を設定します。
   c. [Linux 踏み台ホスト設定] で、[キー名] に EC2 インスタンスにログインするキーペアを選択します。
   d. [Linux 踏み台ホスト設定] で、[許可された IP 範囲] を IP アドレスに設定します。Secure Shell (SSH) を使用して VPC 内の EC2 インスタンスに接続するには、https://checkip.amazonaws.com のサービスを使用してパブリック IP アドレスを確認します。IP アドレスの例は 192.0.2.1/32 です。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

e. [Database General configuration] で、[データベースインスタンスクラス] を [db.t3.micro] に設定します。

f. [データベース名] を **database-test1** に設定します。

g. [データベースマスターユーザー名] には、PDB のマスターユーザー名を入力します。

h. このチュートリアルでは、[Secrets Manager で DB マスターユーザーパスワードを管理] を `false` に設定します。

i. [データベースパスワード] には、任意のパスワードを設定します。このパスワードは、チュートリアルの他の手順のために覚えておいてください。

j. [Database Storage configuration] で、[Database storage type] を [gp2] に設定します。

k. [Database Monitoring configuration] で、[Enable RDS Performance Insights] を false に設定します。

l. その他の設定はすべてデフォルトの値のままにします。[次へ] をクリックして続行します。

5. [スタックオプションの設定] ページでは、すべてのデフォルトオプションをそのまま使用します。[次へ] をクリックして続行します。

6. [スタックの確認] ページで、データベースと Linux 踏み台ホストのオプションを確認した後、[送信] をクリックします。

スタックの作成プロセスが完了したら、BastionStack と RDSNS という名前のスタックを確認して、データベースへの接続に必要な情報をメモします。詳細については、「AWS マネジメントコンソール での CloudFormation スタックデータとリソースの表示」を参照してください。

ステップ 3: SQL クライアントを Oracle DB インスタンスに接続する

標準の SQL クライアントアプリケーションを使用して、DB インスタンスに接続できます。この例では、Oracle コマンドラインツールを使用して Oracle DB インスタンスに接続します。

Oracle DB インスタンスに接続するには

1. DB インスタンスのエンドポイント (DNS 名) とポート番号を見つけます。

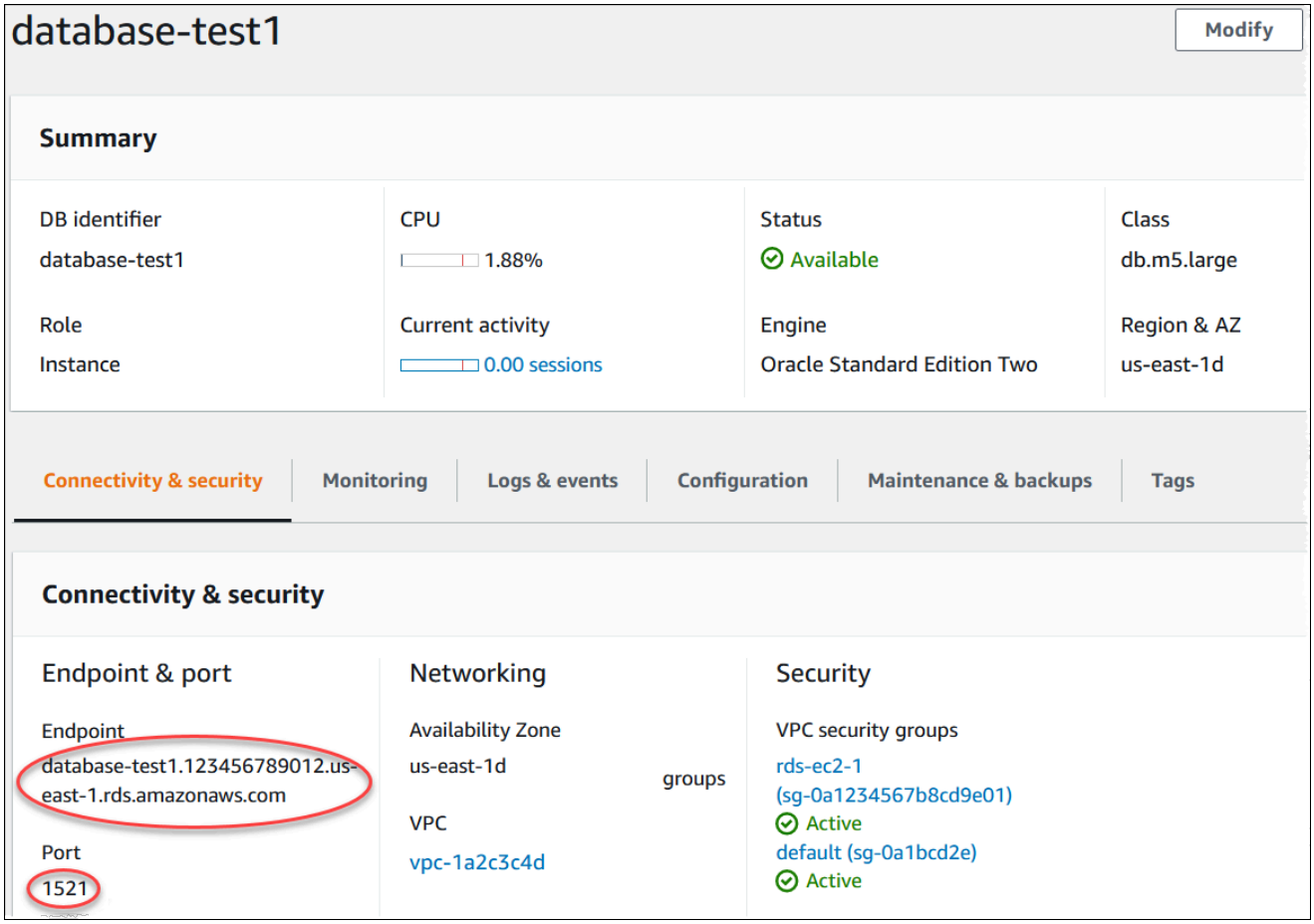
   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[Databases] (データベース) を選択します。

   d. 詳細を表示する Oracle DB インスタンスの名前を選択します。

e. 接続とセキュリティ タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



2. 「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。

SSH を使用して EC2 インスタンスに接続することをお勧めします。SSH クライアントユーティリティが Windows、Linux、または Mac にインストールされている場合は、次のコマンド形式でインスタンスに接続できます。

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

例えば、`ec2-database-connect-key-pair.pem` が Linux の `/dir1` に保存されていて、EC2 インスタンスのパブリック IPv4 DNS が `ec2-12-345-678-90.compute-1.amazonaws.com` であるとします。SSH コマンドは次のようになります。

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 インスタンスのソフトウェアを更新して、最新のバグ修正とセキュリティ更新を入手します。そのためには、次のコマンドを使用します。

> **Note**
>
> -y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を確認するには、このオプションを省略します。

```
sudo dnf update -y
```

4. ウェブブラウザで、https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html に進みます。
5. ウェブページに表示される最新のデータベースバージョンについては、Instant Client Basic Package と SQL*Plus Package の .rpm リンク (.zip リンクではない) をコピーします。例えば、次のリンクは Oracle Database バージョン 21.9 用です。

   - https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm
   - https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm

6. SSH セッションで、wget コマンドを実行して、前のステップで取得したリンクから .rpm ファイルをダウンロードします。次の例では、Oracle Database バージョン 21.9 の .rpm ファイルをダウンロードします。

```
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-
instantclient-basic-21.9.0.0.0-1.el8.x86_64.rpm
wget https://download.oracle.com/otn_software/linux/instantclient/219000/oracle-
instantclient-sqlplus-21.9.0.0.0-1.el8.x86_64.rpm
```

7. 以下の dnf コマンドを実行してパッケージをインストールします。

```
sudo dnf install oracle-instantclient-*.rpm
```

8. SQL*Plus を起動し、Oracle DB インスタンスに接続します。例えば、次のコマンドを入力します。

   oracle-db-instance-endpoint の DB インスタンスエンドポイント (DNS 名) を置き換え、*admin* で使用したマスターユーザー名に置き換えます。Oracle の[簡単作成] を使用する場合、データベース名は DATABASE です。パスワードの入力を求められたときに使用したマスターパスワードを入力します。

   ```
   sqlplus admin@oracle-db-instance-endpoint:1521/DATABASE
   ```

   ユーザーのパスワードを入力すると、次のような出力が表示されます。

   ```
   SQL*Plus: Release 21.0.0.0.0 - Production on Wed Mar 1 16:41:28 2023
   Version 21.9.0.0.0

   Copyright (c) 1982, 2022, Oracle.  All rights reserved.

   Enter password:
   Last Successful login time: Wed Mar 01 2023 16:30:52 +00:00

   Connected to:
   Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 - Production
   Version 19.18.0.0.0

   SQL>
   ```

   RDS for Oracle DB インスタンスへの接続の詳細については、「Oracle DB インスタンスへの接続」を参照してください。DB インスタンスに接続できない場合は、「Amazon RDS DB インスタンスに接続できない」を参照してください。

   セキュリティのためには、暗号化された接続を使用することがベストプラクティスです。クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない Oracle 接続を使用します。暗号化された接続の使用については、「Oracle DB インスタンス接続の保護」を参照してください。

9. SQL コマンドを実行する。

   例えば、次の SQL コマンドは、現在の日付を表示します。

   ```
   SELECT SYSDATE FROM DUAL;
   ```

ステップ 4: EC2 インスタンスと DB インスタンスを削除する

作成したサンプル EC2 インスタンスと DB インスタンスに接続して、探索したら、料金がこれ以上発生しないように、それらを削除します。

CloudFormation を使用してリソースを作成した場合は、このステップをスキップして次のステップに進みます。

EC2 インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. EC2 インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Terminate (終了)] を選択してください。

EC2 インスタンスの削除の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。

最終的な DB スナップショットを作成せずに DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除する DB インスタンスを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成] と [自動バックアップの保持] をクリアします。
6. 確認を完了し、[削除] を選択します。

(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する

CloudFormation を使用してリソースを作成した場合は、サンプル EC2 インスタンスと DB インスタンスに接続して確認を済ませた後、それ以上料金が発生しないように、CloudFormation スタックを削除します。

CloudFormation リソースを削除するには

1. CloudFormation コンソールを開きます。
2. CloudFormation コンソールの [スタック] ページで、ルートスタック (VPCStack、BastionStack、または RDSNS という名前がついていないスタック) を選択します。
3. [削除] を選択します。
4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudFormation でのスタックの削除方法の詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation コンソールでのスタックの削除」を参照してください。

(オプション) DB インスタンスを Lambda 関数に接続する

RDS for Oracle DB インスタンスを Lambda サーバーレスコンピューティングリソースに接続することもできます。Lambda 関数を使用すると、インフラストラクチャをプロビジョニングしたり管理したりせずにコードを実行できます。Lambda 関数を使用すると、1 日に 数十件のイベントから 1 秒間に数百件のイベントまで、あらゆる規模のコード実行リクエストに自動的に応答することもできます。詳細については、「Lambda 関数と DB インスタンスを自動的に接続する」を参照してください。

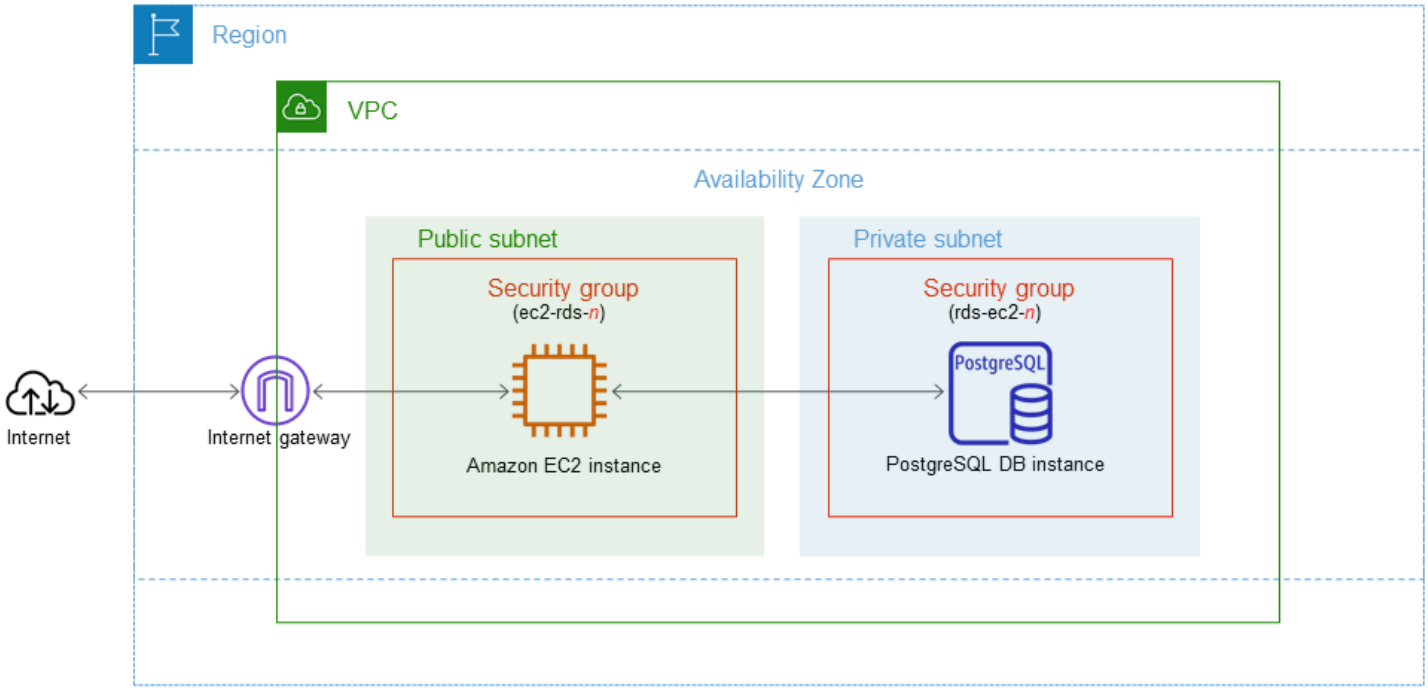
PostgreSQL DB インスタンスを作成して接続する

このチュートリアルでは、EC2 インスタンスと RDS for PostgreSQL DB インスタンスを作成します。このチュートリアルでは、標準の PostgreSQL クライアントを使用して EC2 インスタンスから DB インスタンスにアクセスする方法を示します。ベストプラクティスとして、このチュートリアルでは、プライベート DB インスタンスを仮想プライベートクラウド (VPC) に作成します。ほとんどの場合、EC2 インスタンスなど、同じ VPC 内の他のリソースは DB インスタンスにアクセスできますが、VPC 外部のリソースはアクセスできません。

チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。1 つのアベイラビリティーゾーンで、EC2 インスタンスはパブリックサブネットにあり、DB インスタンスはプライベートサブネットにあります。

> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

次の図は、チュートリアルが完了した時点の設定を示しています。



このチュートリアルでは、次のいずれかの方法を使用してリソースを作成できます。

1. AWS マネジメントコンソール を使用する -「EC2 インスタンスの作成」と「PostgreSQL DB インスタンスを作成する」
2. CloudFormation を使用してデータベースインスタンスと EC2 インスタンスを作成する - (オプション) CloudFormation を使用して VPC、EC2 インスタンス、PostgreSQL インスタンスを作成する

最初の方法では、[簡単作成] を使用して、AWS マネジメントコンソール でプライベート PostgreSQL DB インスタンスを作成します。ここでは、DB エンジンタイプ、DB インスタンスサイズ、DB インスタンス識別子のみを指定します。[Easy create (簡易作成)] では、他の設定オプションのデフォルト設定を使用します。

代わりに [標準作成] を使用する場合は、DB インスタンスの作成時にさらに多くの設定オプションを指定できます。このようなオプションには、可用性、セキュリティ、バックアップ、メンテナンスの設定があります。パブリック DB インスタンスを作成するには、[標準作成] を使用する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

トピック

- 前提条件
- EC2 インスタンスの作成
- PostgreSQL DB インスタンスを作成する
- (オプション) CloudFormation を使用して VPC、EC2 インスタンス、PostgreSQL インスタンスを作成する
- PostgreSQL DB インスタンスへの接続
- EC2 インスタンスと DB インスタンスを削除する
- (オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する
- (オプション) DB インスタンスを Lambda 関数に接続する

前提条件

開始する前に、以下のセクションのステップを完了してください。

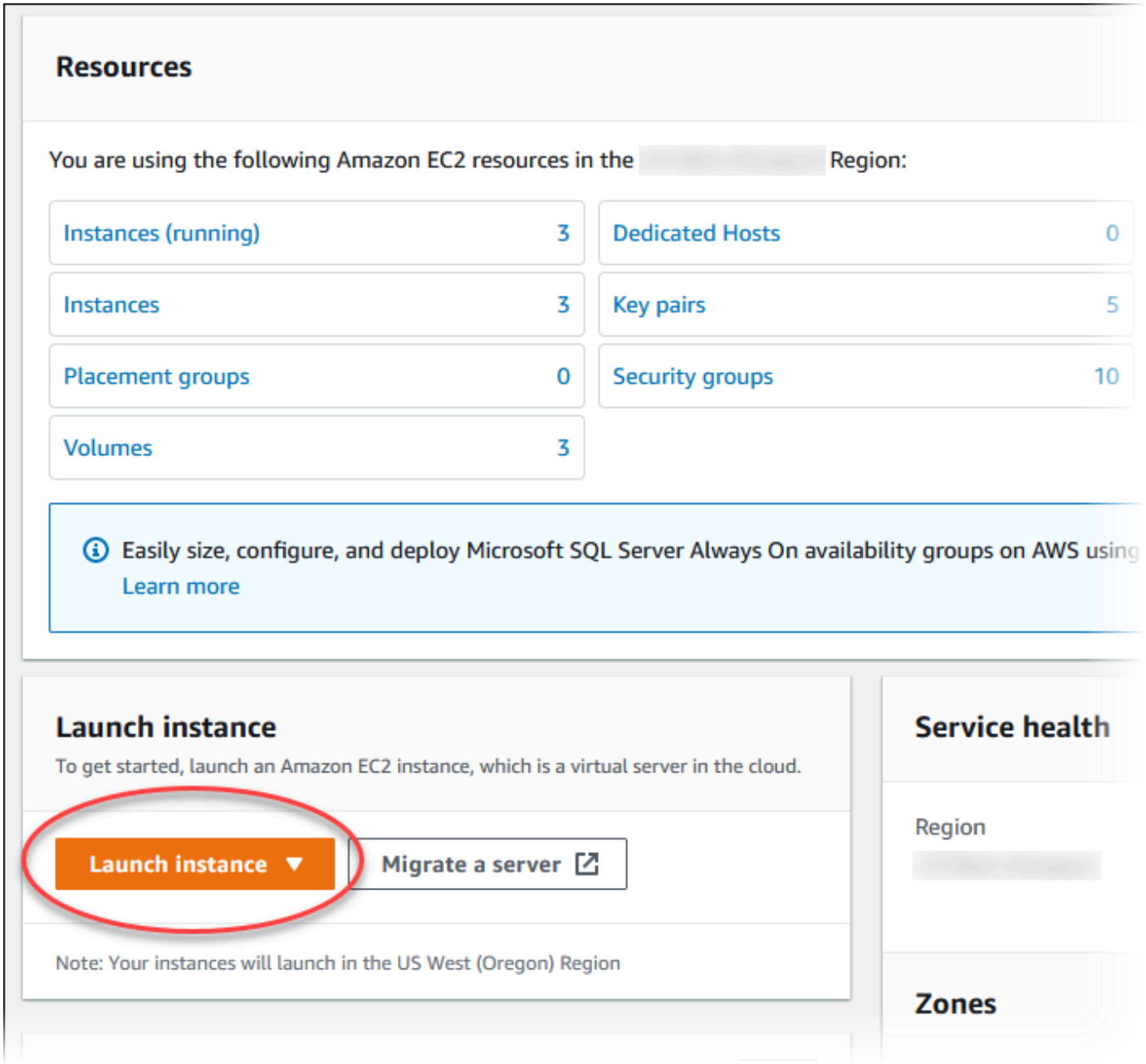
- AWS アカウント へのサインアップ

EC2 インスタンスの作成

データベースへの接続に使用する Amazon EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、EC2 インスタンスを作成する AWS リージョンを選択します。
3. 次の図に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの起動] を選択します。

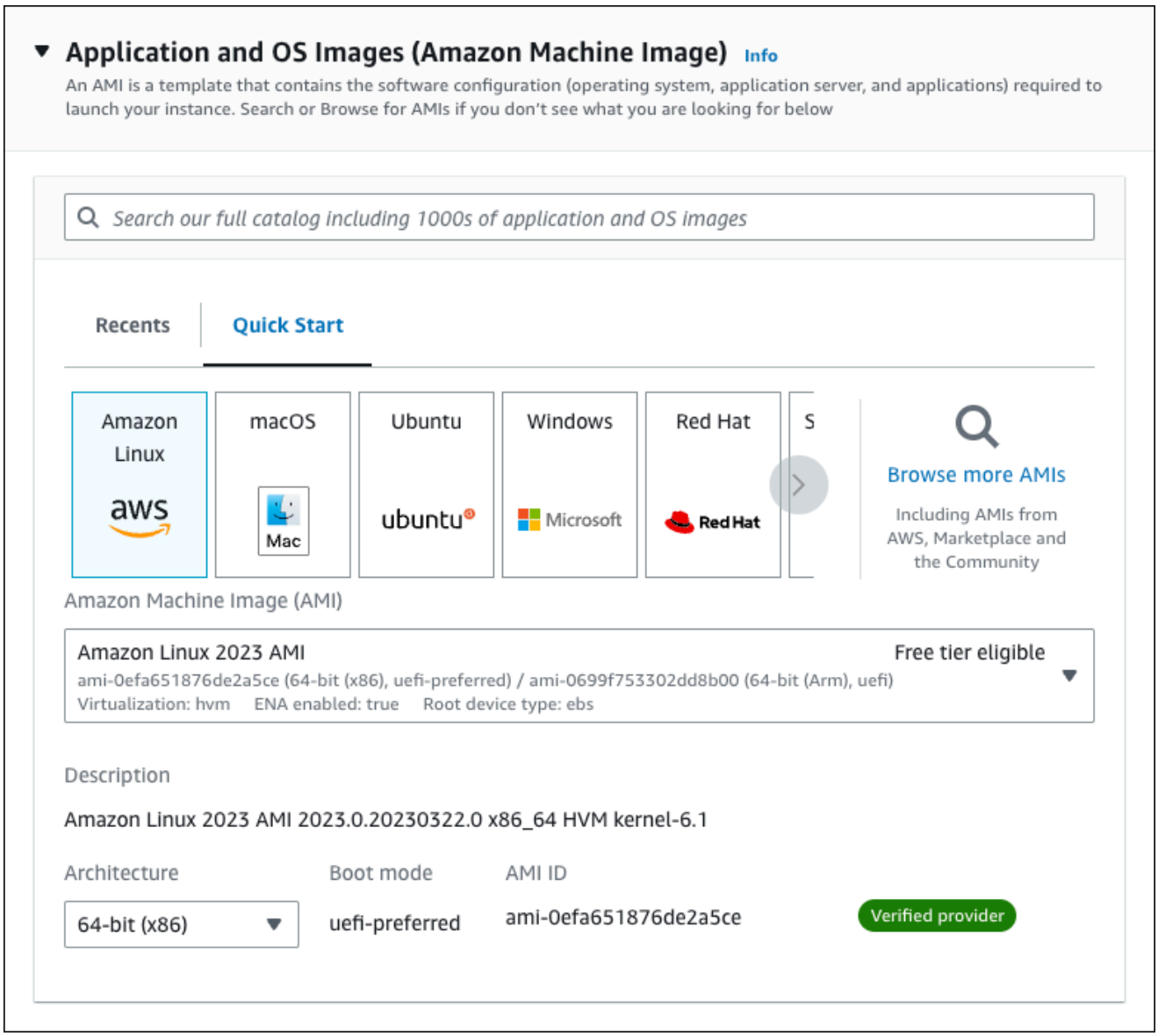


[インスタンスを起動] ページが開きます。

4. [インスタンスを起動] ページで次の設定を選択します。

   a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`ec2-database-connect`** と入力します。

   b. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[Amazon Linux] を選択し、[Amazon Linux 2023 AMI] を選択します。他の選択肢は、デフォルトの選択のままにします。



c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するには、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

   新しいキーペアの作成については、「Amazon EC2 ユーザーガイド」の「キーペアの作成」を参照してください。

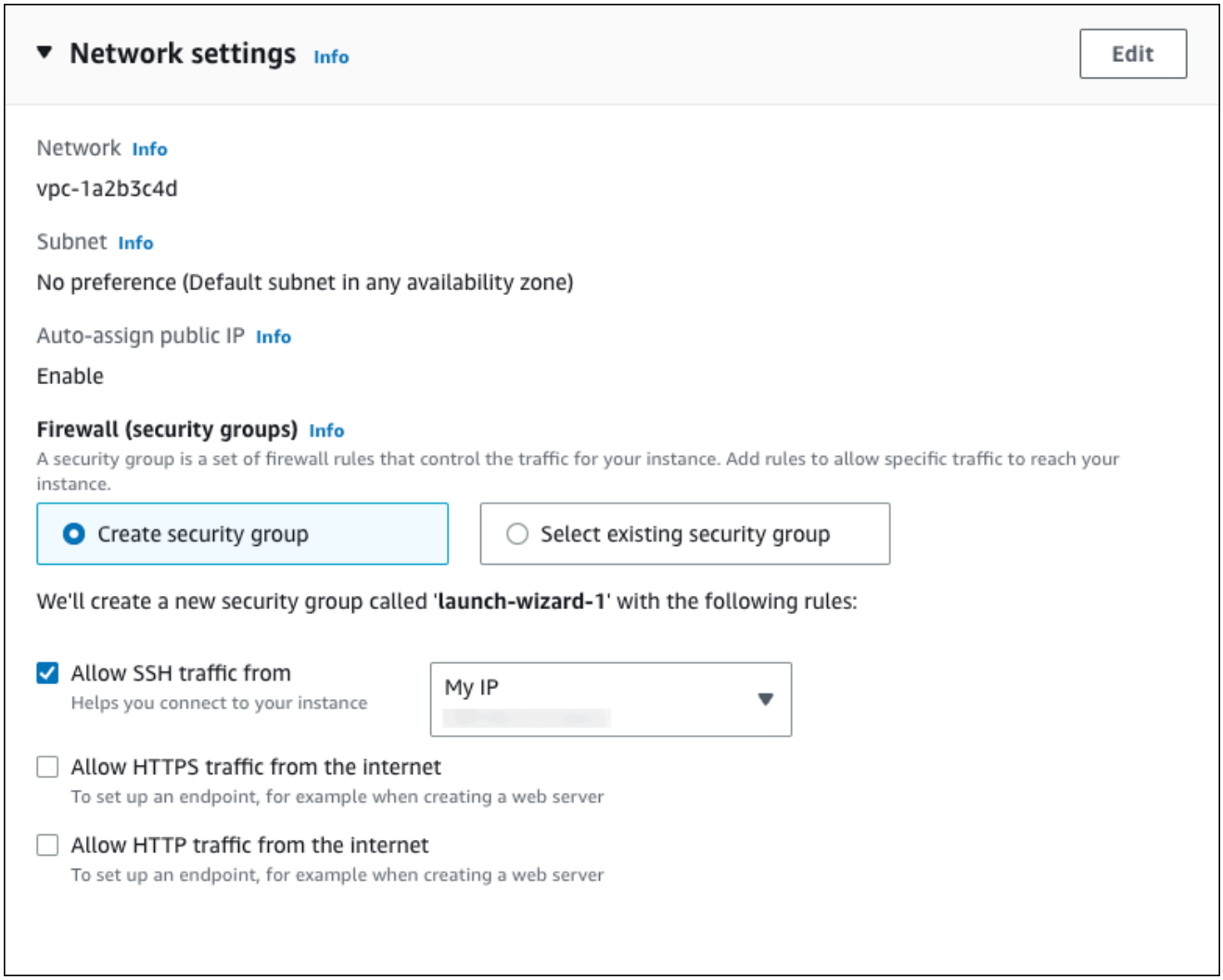
e. ネットワーク設定の [SSH トラフィックを許可] で、EC2 インスタンスへの SSH 接続のソースを選択します。

表示された IP アドレスが SSH 接続に適している場合は、[My IP] (マイ IP)を選択できます。それ以外の場合は、Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は 192.0.2.1/32 です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

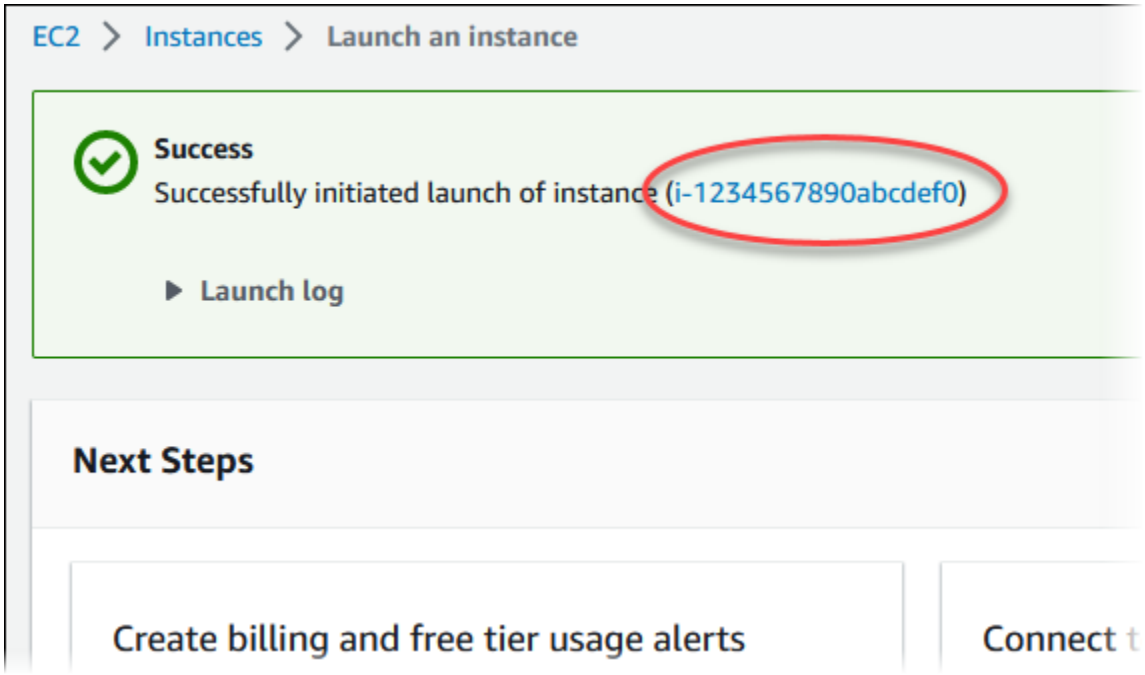
以下のイメージは、[ネットワーク設定] セクションの例を示しています。



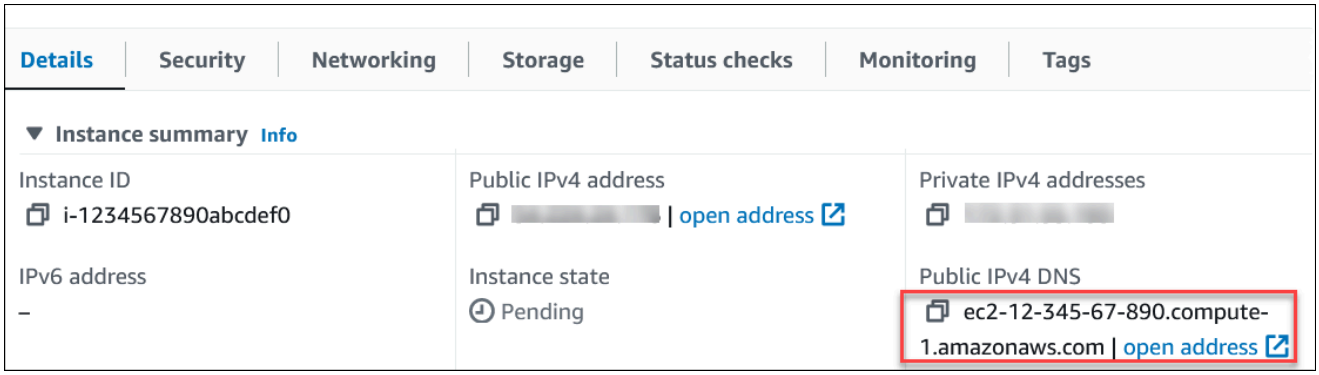
f. 残りのセクションをデフォルト値のままにします。

g. [概要] パネルで、EC2 インスタンス設定の概要を確認し、準備ができたら、[インスタンスの起動] を選択します。

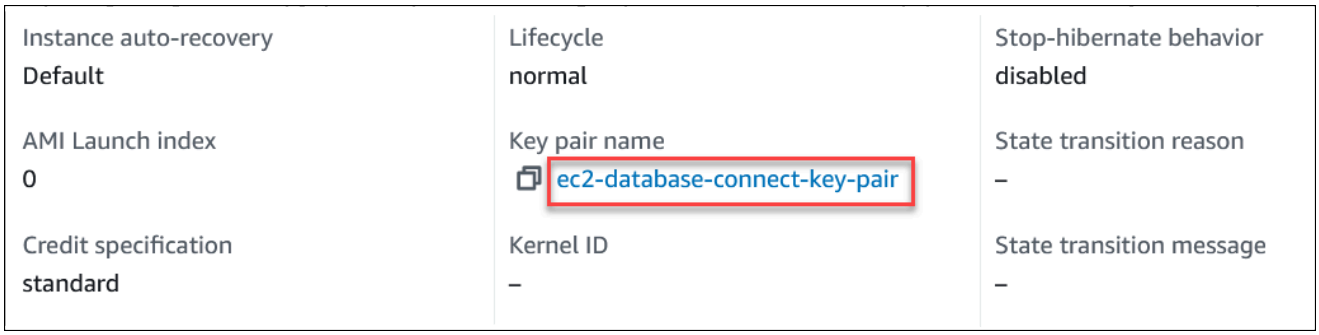
5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開き、EC2 インスタンスを選択します。
7. [詳細] タブで、SSH を使用して接続するときに必要な次の値を書き留めます。
   a. [インスタンスの概要] で、[パブリック IPv4 DNS] の値を書き留めます。



   b. [インスタンスの詳細] で、[キーペア名] の値を書き留めます。



8. EC2 インスタンスの [インスタンス状態] が [実行中] になるまで待ってから、続行します。

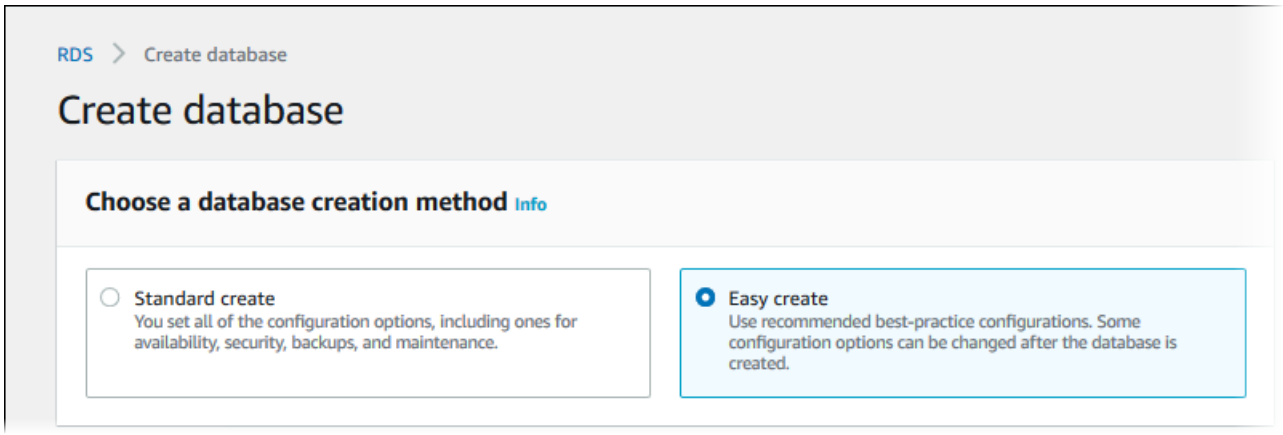
PostgreSQL DB インスタンスを作成する

Amazon RDS の基本的な構成要素は DB インスタンスです。これは、PostgreSQL データベースを実行する環境です。

この例では、[簡単作成] を使用して、DB インスタンスクラスが db.t3.micro の PostgreSQL データベースエンジンを実行する DB インスタンスを作成します。

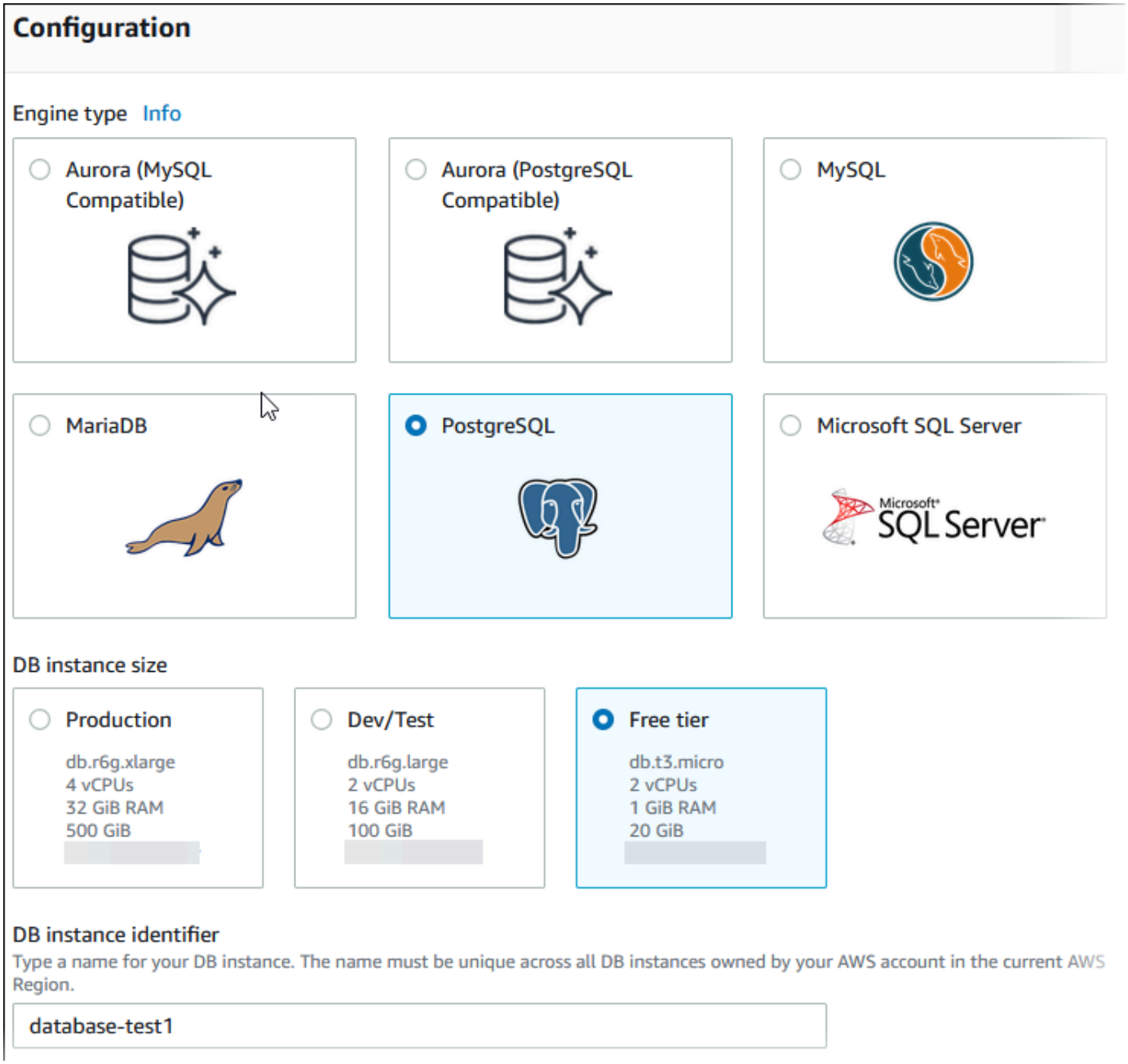
簡易作成で PostgreSQL DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [Create database (データベースの作成)] を選択し、[Easy create (簡易作成)] が選択されていることを確認します。



5. [設定] で、[PostgreSQL] を選択します。
6. [DB インスタンスサイズ] で、[無料利用枠] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。
7. [DB instance identifier] (DB インスタンス識別子) に**`database-test1`** と入力します。
8. [マスターユーザー名] に、マスターユーザーの名前を入力するか、デフォルト名 (**`postgres`**) のままにします。

[データベースの作成] ページは、次のイメージのようになります。無料プランアカウントの場合、[無料利用枠] が表示されます。有料プランアカウントの場合、[サンドボックス] が表示されます。

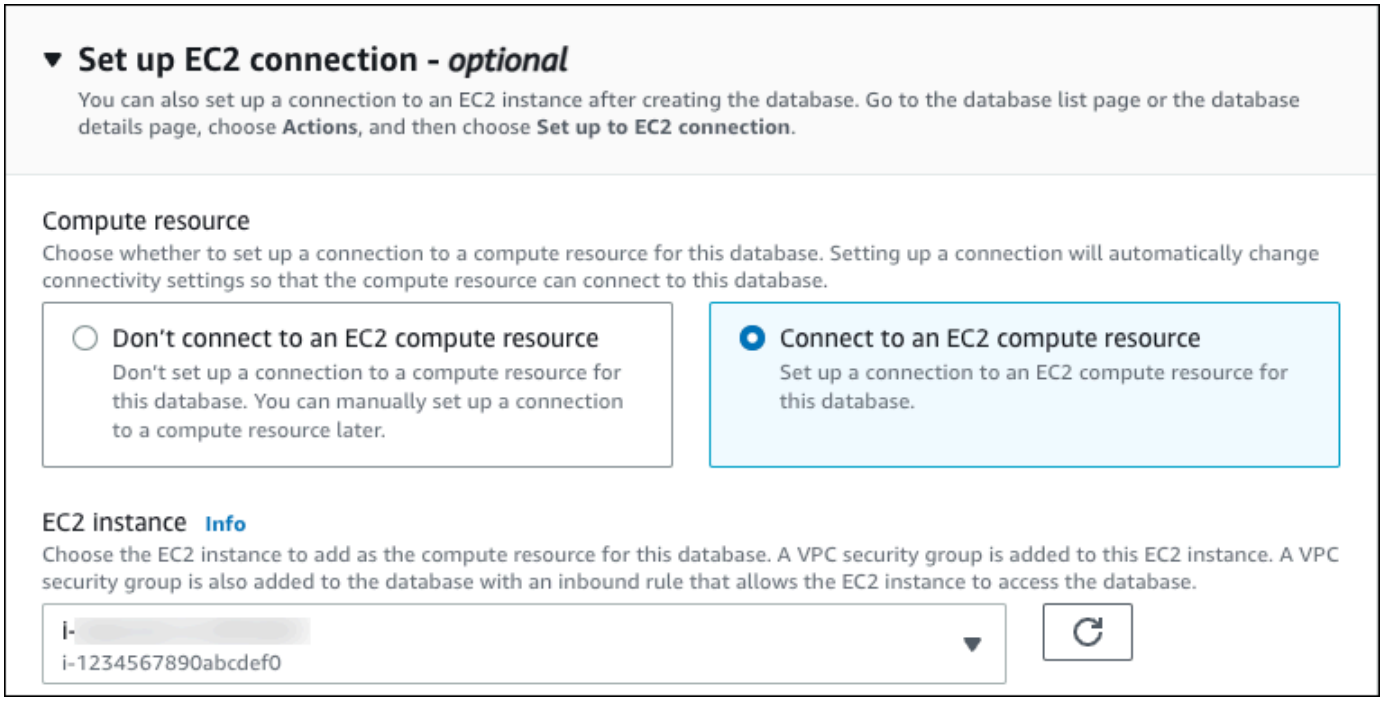


9. DB インスタンス用に自動生成されたマスターパスワードを使用するには、[パスワードの自動生成] を選択します。

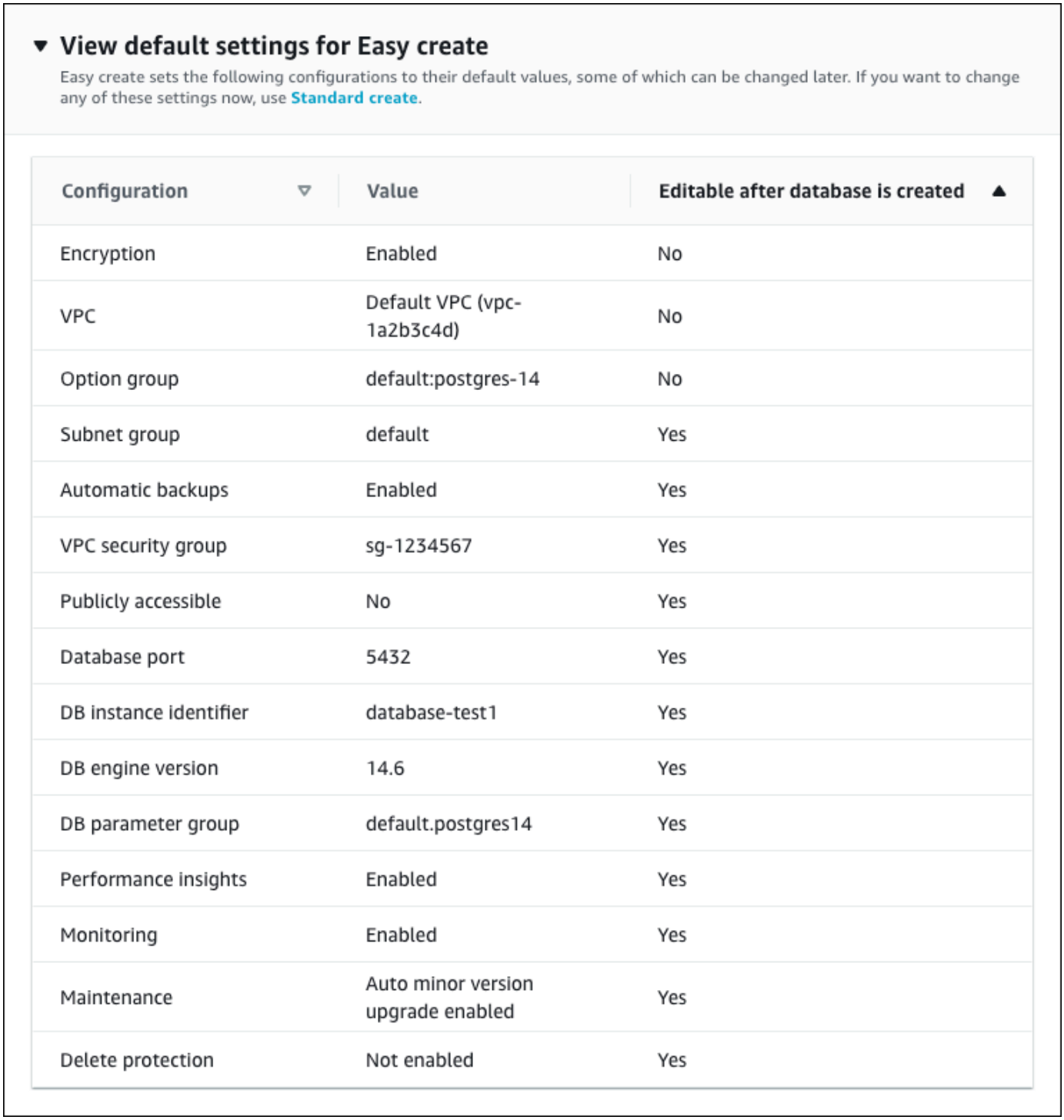
   マスターパスワードを入力するには、[パスワードの自動生成] チェックボックスをオフにして、[マスターパスワード] と [パスワードの確認] に同じパスワードを入力します。

10. 以前に作成した EC2 インスタンスとの接続をセットアップするには、[EC2 接続のセットアップ - オプション] を開きます。

    [EC2 コンピューティングリソースに接続] を選択します。以前に作成した EC2 インスタンスを選択します。



11. [簡易作成のデフォルト設定を表示] を開きます。



[Easy Create (簡易作成)] で使用されるデフォルト設定を調べることができます。[データベース作成後に編集可能] 列には、データベース作成後に変更できるオプションが表示されます。

- その列の設定に [いいえ] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成できます。

- その列の設定に [はい] があり、別の設定が必要な場合は、[標準作成] を使用して DB インスタンスを作成するか、DB インスタンスの作成後に設定を変更できます。

12. [データベースの作成] を選択します。

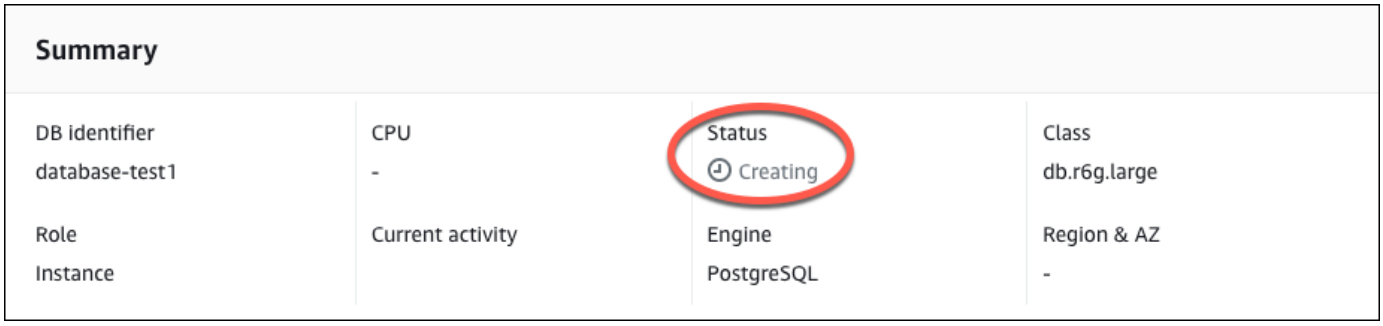
    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    表示されるユーザー名とパスワードを使用して、マスターユーザーとして DB インスタンスに接続できます。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。
    > DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンスを変更することができます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

13. [データベース] リストで、新しい PostgreSQL DB インスタンスの名前を選択し、詳細を表示します。

    DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは [作成中] のままです。



    ステータスが [Available] (利用可能) に変わったら、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しいインスタンスを使用できるようになるまで最長 20 分かかることがあります。

(オプション) CloudFormation を使用して VPC、EC2 インスタンス、PostgreSQL インスタンスを作成する

コンソールを使用して VPC、EC2 インスタンス、PostgreSQL DB インスタンスを作成する代わりに、CloudFormation を使用して Infrastructure as Code として取り扱うことで、AWS リソースをプロビジョンできます。AWS リソースをより小さく管理しやすい単位に整理するには、CloudFormation のネストされたスタック機能を使用できます。詳細については、「CloudFormation コンソールでのスタックの作成」と「ネストされたスタックの操作」を参照してください。

> **Important**
>
> CloudFormation は無料ですが、CloudFormation が作成するリソースは実動のものです。これらのリソースを終了するまで、標準使用料が発生します。詳細については、「RDS for PostgreSQL の料金」を参照してください。

CloudFormation コンソールを使用してリソースを作成するには、以下のステップを実行します。

- CloudFormation テンプレートをダウンロードする
- CloudFormation を使用してリソースを設定する

CloudFormation テンプレートをダウンロードする

CloudFormation テンプレートは、スタックに作成するリソースに関する設定情報が含まれる JSON または YAML のテキストファイルです。このテンプレートは、RDS インスタンスとともに VPC と踏み台ホストも作成します。

テンプレートファイルをダウンロードするには、次のリンク、PostgreSQL CloudFormation template を開きます。

この Github ページで、[Download raw file] ボタンをクリックしてテンプレートの YAML ファイルを保存します。

CloudFormation を使用してリソースを設定する

> **Note**
>
> このプロセスを開始する前に、AWS アカウント に EC2 インスタンスのキーペアがあることを確認してください。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

CloudFormation テンプレートを使用する場合は、適切なパラメータを選択して、リソースが正しく作成されていることを確認する必要があります。以下のステップを実行します。

1. AWS マネジメントコンソール にサインインし、CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [Create Stack] (スタックの作成) を選択します。
3. [テンプレートの指定] セクションで、[コンピュータからテンプレートファイルをアップロード] を選択し、[次へ] をクリックします。
4. [スタックの詳細を指定] ページで、次のパラメータを設定します。
   a. [スタック名] は [PostgreSQLTestStack] に設定します。
   b. [パラメータ] で、3 つのアベイラビリティーゾーンを選択して、[アベイラビリティーゾーン] を設定します。
   c. [Linux 踏み台ホスト設定] で、[キー名] に EC2 インスタンスにログインするキーペアを選択します。
   d. [Linux 踏み台ホスト設定] で、[許可された IP 範囲] を IP アドレスに設定します。Secure Shell (SSH) を使用して VPC 内の EC2 インスタンスに接続するには、https://checkip.amazonaws.com のサービスを使用してパブリック IP アドレスを確認します。IP アドレスの例は 192.0.2.1/32 です。

> **Warning**
>
> SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリック EC2 インスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使った EC2 インスタンスへのアクセスを承認します。

e. [Database General configuration] で、[データベースインスタンスクラス] を [db.t3.micro] に設定します。

f. [データベース名] を **database-test1** に設定します。

g. [データベースマスターユーザー名] には、PDB のマスターユーザー名を入力します。

h. このチュートリアルでは、[Secrets Manager で DB マスターユーザーパスワードを管理] を `false` に設定します。

i. [データベースパスワード] には、任意のパスワードを設定します。このパスワードは、チュートリアルの他の手順のために覚えておいてください。

j. [Database Storage configuration] で、[Database storage type] を [gp2] に設定します。

k. [Database Monitoring configuration] で、[Enable RDS Performance Insights] を false に設定します。

l. その他の設定はすべてデフォルトの値のままにします。[次へ] をクリックして続行します。

5. [スタックオプションの設定] ページでは、すべてのデフォルトオプションをそのまま使用します。[次へ] をクリックして続行します。

6. [スタックの確認] ページで、データベースと Linux 踏み台ホストのオプションを確認した後、[送信] をクリックします。

スタックの作成プロセスが完了したら、BastionStack と RDSNS という名前のスタックを確認して、データベースへの接続に必要な情報をメモします。詳細については、「AWS マネジメントコンソール での CloudFormation スタックデータとリソースの表示」を参照してください。

PostgreSQL DB インスタンスへの接続

pgadmin または psql を使用して DB インスタンスに接続できます。この例では、psql コマンドラインクライアントを使用して PostgreSQL DB インスタンスに接続する方法について説明します。

psql を使用して PostgreSQL DB インスタンスに接続するには

1. DB インスタンスのエンドポイント (DNS 名) とポート番号を見つけます。

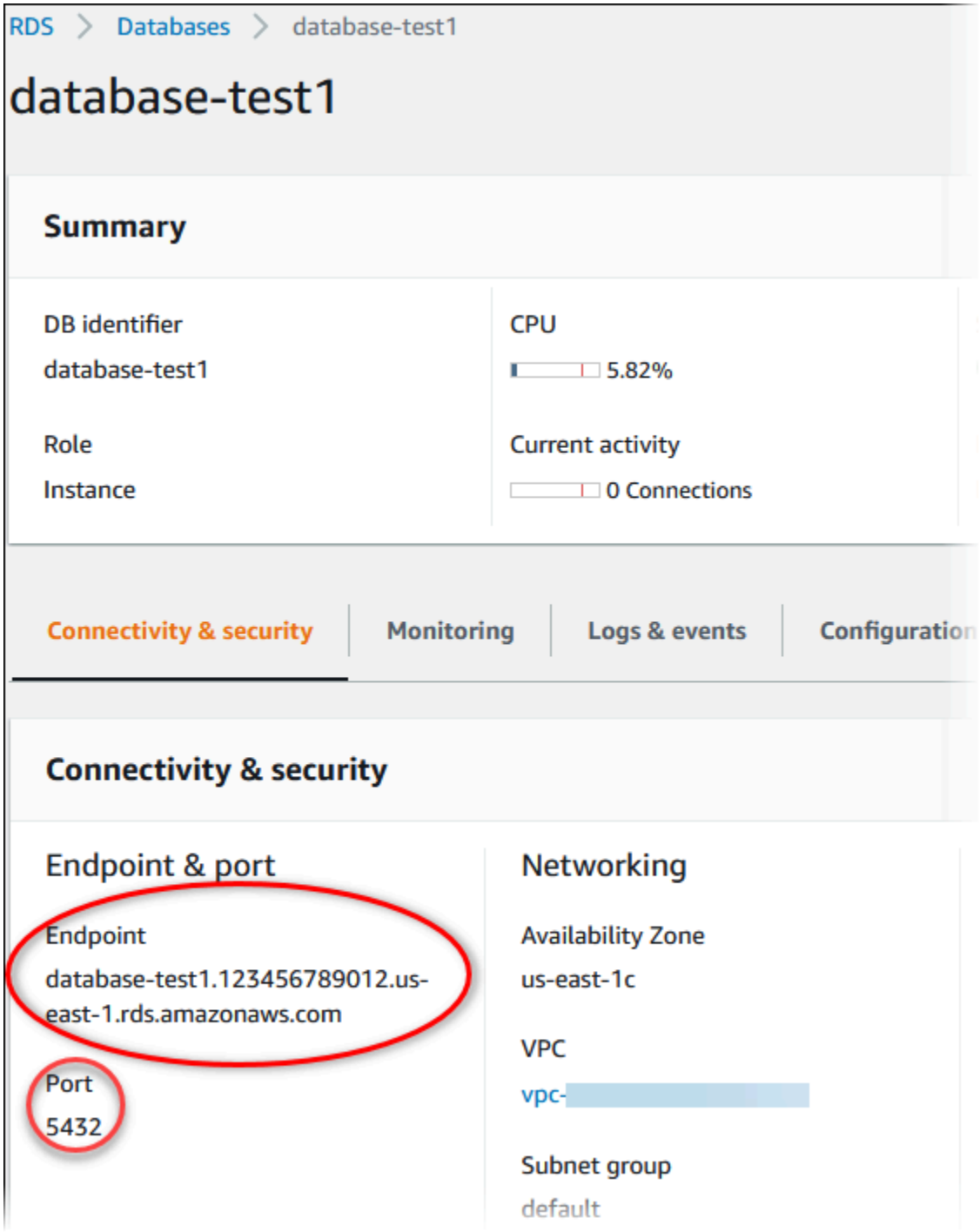
   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[Databases] (データベース) を選択します。

   d. PstgreSQL DB インスタンス名を選択して、詳細を表示します。

e. 接続とセキュリティ タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



2. 「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。

SSH を使用して EC2 インスタンスに接続することをお勧めします。SSH クライアントユーティリティが Windows、Linux、または Mac にインストールされている場合は、次のコマンド形式でインスタンスに接続できます。

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

例えば、ec2-database-connect-key-pair.pem が Linux の /dir1 に保存されていて、EC2 インスタンスのパブリック IPv4 DNS が ec2-12-345-678-90.compute-1.amazonaws.com であるとします。SSH コマンドは次のようになります。

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. EC2 インスタンスのソフトウェアを更新して、最新のバグ修正とセキュリティ更新を入手します。これを行うには、次のコマンドを使用します。

> **Note**
>
> -y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を確認するには、このオプションを省略します。

```
sudo dnf update -y
```

4. PostgreSQL の psql コマンドラインクライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install postgresql15
```

5. PostgreSQL DB インスタンスに接続します。例えば、クライアントコンピュータのコマンドプロンプトで、次のコマンドを入力します。このアクションにより、psql クライアントを使用して、PostgreSQL DB インスタンスに接続できます。

   endpoint の DB インスタンスエンドポイント (DNS 名) に置き換え、*postgres* のために接続するデータベース名 --dbname を、*postgres* に使用したマスターユーザー名に置き換えます。パスワードの入力を求められたときに使用したマスターパスワードを入力します。

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

ユーザーのパスワードを入力すると、次のような出力が表示されます。

```
psql (14.3, server 14.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

PostgreSQL DB インスタンスへの接続の詳細については、「PostgreSQL データベースエンジンを実行する DB インスタンスへの接続」を参照してください。DB インスタンスに接続できない場合は、「RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング」を参照してください。

セキュリティのためには、暗号化された接続を使用することがベストプラクティスです。クライアントとサーバーが同じ VPC にあり、ネットワークが信頼できる場合に限り、暗号化されていない PostgreSQL 接続を使用します。暗号化された接続の使用については、「SSL 経由での PostgreSQL DB インスタンスへの接続」を参照してください。

6. SQL コマンドを実行する。

   例えば、次の SQL コマンドは、現在の日付と時刻を表示します。

```
SELECT CURRENT_TIMESTAMP;
```

EC2 インスタンスと DB インスタンスを削除する

作成したサンプル EC2 インスタンスと DB インスタンスに接続して、探索したら、料金がこれ以上発生しないように、それらを削除します。

CloudFormation を使用してリソースを作成した場合は、このステップをスキップして次のステップに進みます。

EC2 インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択してください。
3. EC2 インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Terminate (終了)] を選択してください。

EC2 インスタンスの削除の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。

最終的な DB スナップショットを作成しないで DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除する DB インスタンスを選択します。
4. [アクション] で、[削除] を選択します。
5. [最終スナップショットを作成] と [自動バックアップの保持] をクリアします。
6. 確認を完了し、[削除] を選択します。

(オプション) CloudFormation で作成された EC2 インスタンスと DB インスタンスを削除する

CloudFormation を使用してリソースを作成した場合は、サンプル EC2 インスタンスと DB インスタンスに接続して確認を済ませた後、それ以上料金が発生しないように、CloudFormation スタックを削除します。

CloudFormation リソースを削除するには

1. CloudFormation コンソールを開きます。
2. CloudFormation コンソールの [スタック] ページで、ルートスタック (VPCStack、BastionStack、または RDSNS という名前がついていないスタック) を選択します。
3. [削除] を選択します。

4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudFormation でのスタックの削除方法の詳細については、「AWS CloudFormation ユーザーガイド」の「CloudFormation コンソールでのスタックの削除」を参照してください。

(オプション) DB インスタンスを Lambda 関数に接続する

RDS for PostgreSQL DB インスタンスを Lambda サーバーレスコンピューティングリソースに接続することもできます。Lambda 関数を使用すると、インフラストラクチャをプロビジョニングしたり管理したりせずにコードを実行できます。Lambda 関数を使用すると、1 日に 数十件のイベントから 1 秒間に数百件のイベントまで、あらゆる規模のコード実行リクエストに自動的に応答することもできます。詳細については、「Lambda 関数と DB インスタンスを自動的に接続する」を参照してください。

チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する

このチュートリアルでは、PHP を使用して Apache ウェブサーバーをインストールする方法と MariaDB、MySQL、または PostgreSQL データベースを作成する方法を示します。ウェブサーバーは Amazon Linux 2023 を使用して Amazon EC2 インスタンスで実行し、MySQL DB インスタンスと PostgreSQL DB インスタンスのいずれかを選択できます。Amazon EC2 インスタンスと DB インスタンスはいずれも、Amazon VPC サービスに基づき、仮想プライベートクラウド (VPC) で実行されます。

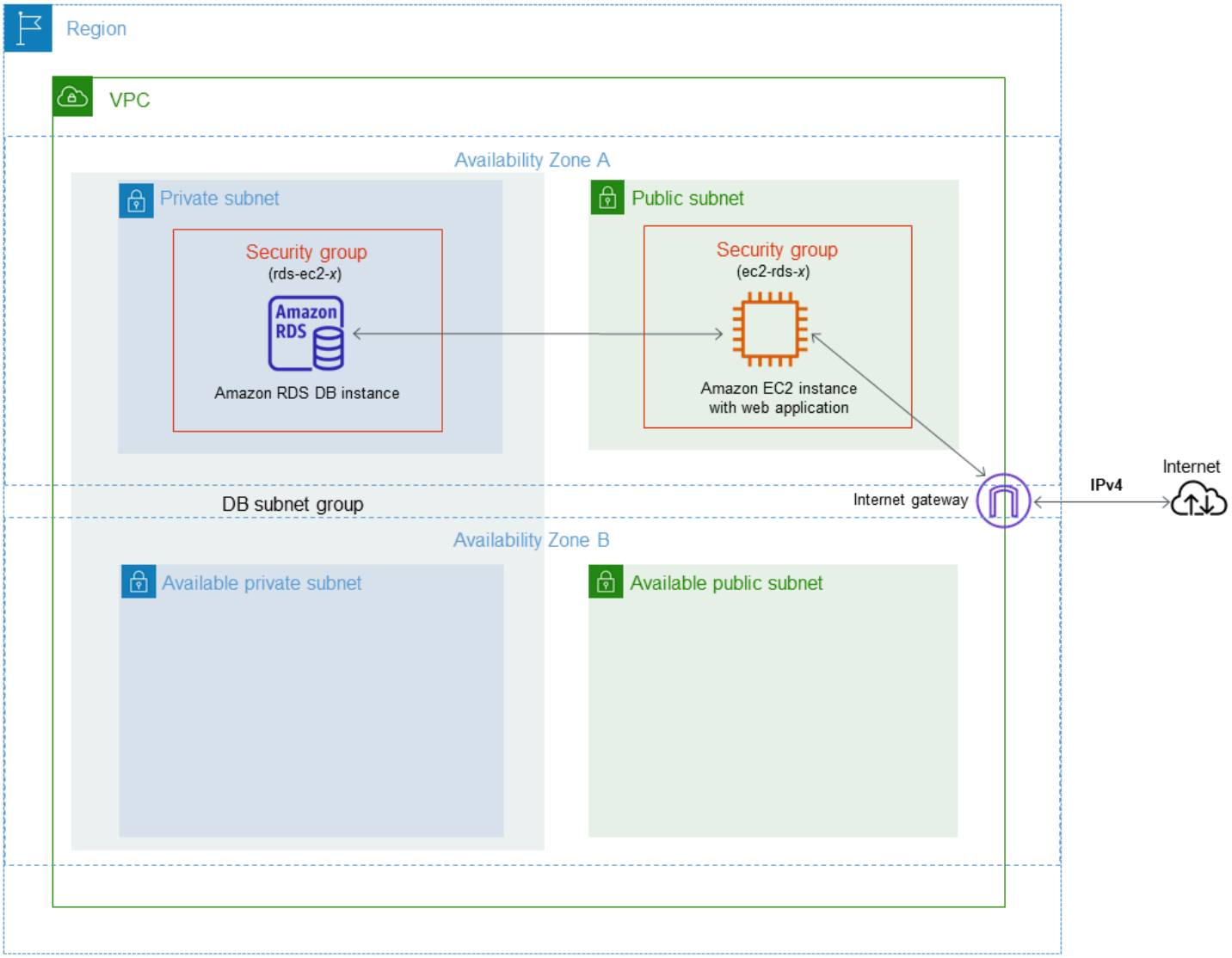
> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

> **Note**
>
> このチュートリアルは Amazon Linux 2023 で機能します。他の Linux のバージョンでは機能しない場合があります。

次のチュートリアルでは、AWS アカウント のデフォルトの VPC、サブネット、およびセキュリティグループを使用する EC2 インスタンスを作成します。このチュートリアルでは、DB インスタンスを作成し、作成した EC2 インスタンスとの接続を自動的にセットアップする方法を示します。次に、EC2 インスタンスにウェブサーバーをインストールする方法を示します。DB インスタンスのエンドポイントを使用して、ウェブサーバーを VPC の DB インスタンスに接続します。

1. EC2 インスタンスを起動して DB インスタンスに接続する
2. Amazon RDS DB インスタンスを作成する
3. EC2 インスタンスにウェブサーバーをインストールします

次の図は、チュートリアルが完了した時点の設定を示しています。



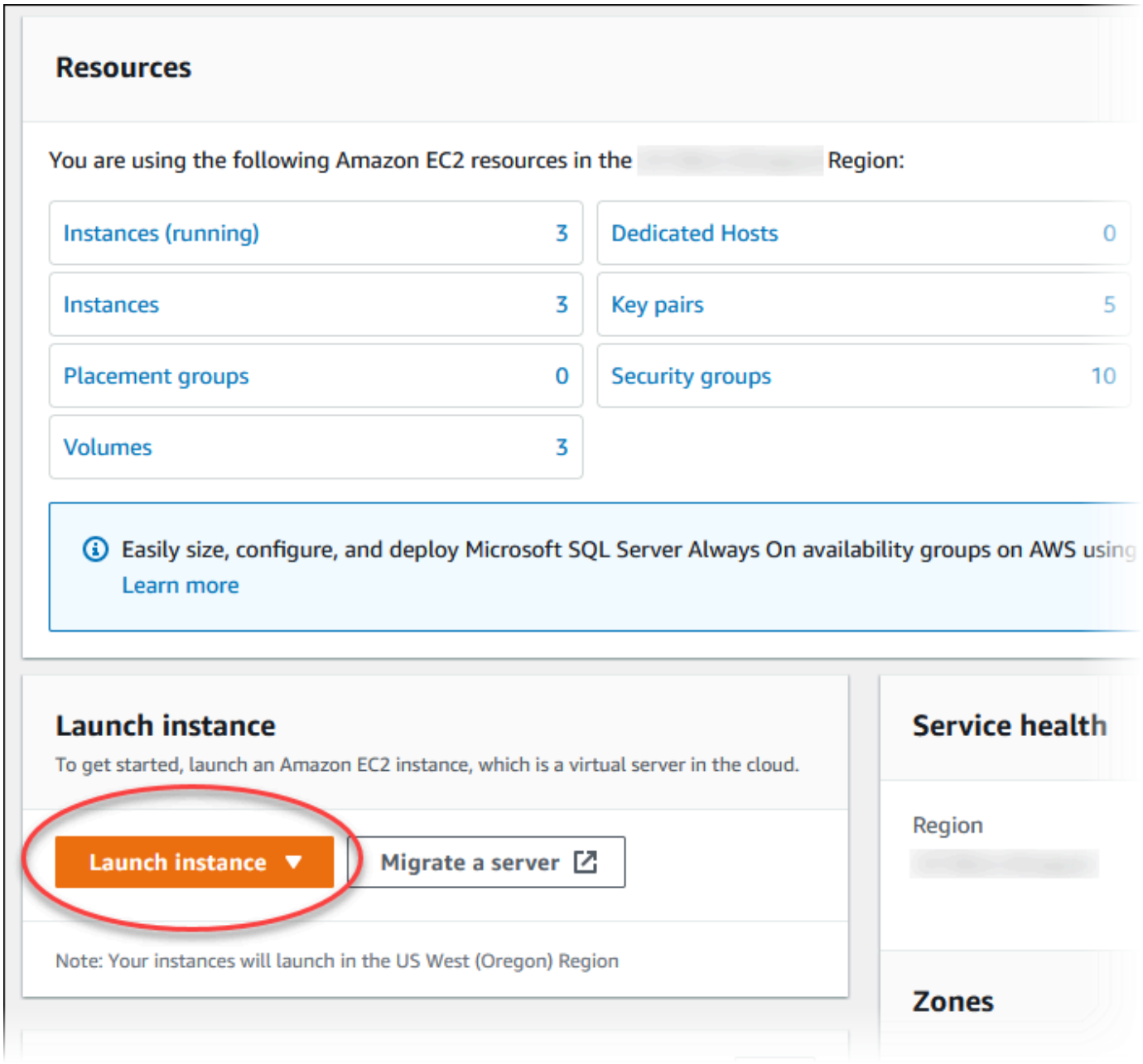
> **Note**
>
> チュートリアルを完了すると、VPC 内の各アベイラビリティーゾーンにパブリックサブネットとプライベートサブネットができます。このチュートリアルでは、AWS アカウント にデフォルトの VPC を使用し、EC2 インスタンスと DB インスタンス間の接続を自動的に設定します。このシナリオで代わりに新しい VPC を設定する場合は、チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用) のタスクを完了してください。

EC2 インスタンスを起動して DB インスタンスに接続する

VPC のパブリックサブネットで Amazon EC2 インスタンスを作成します。

EC2 インスタンスを起動するには

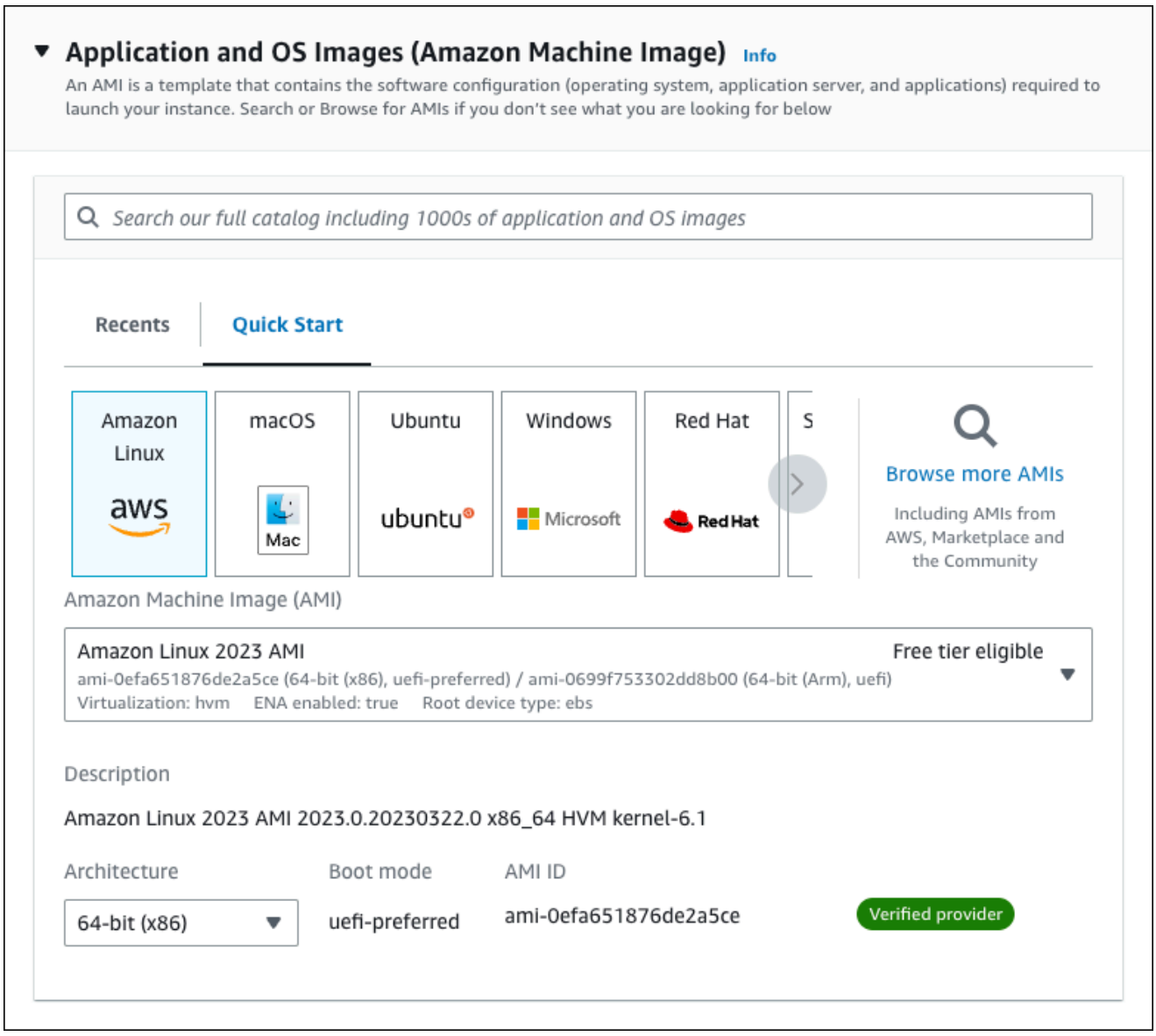
1. AWS マネジメントコンソール にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. AWS マネジメントコンソールで、EC2 インスタンスを作成する AWS リージョンを選択します。
3. 次に示すように、[EC2 ダッシュボード] を選択し、次に [インスタンスの作成] を選択します。



4. [インスタンスを起動] ページで次の設定を選択します。

a. [Name and tags] (名前とタグ) の、[Name] (名前) で、**`tutorial-ec2-instance-web-server`** と入力します。

b. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[Amazon Linux] を選択し、[Amazon Linux 2023 AMI] を選択します。他の選択肢をデフォルトのままにします。



c. [Instance type] (インスタンスタイプ) で [t2.micro] を選択します。

d. [Key pair (login)] (キーペア (ログイン)) で、[Key pair name] (キーペア名) を選択して、既存のキーペアを使用します。Amazon EC2 インスタンスの新しい key pair を作成するには、[Create new key pair] (新しい key pair を作成する) を選択し、[Create key pair] (キーペアを作成する) ウィンドウを使用して作成します。

新しいキーペアの作成については、「Amazon EC2 ユーザーガイド」の「キーペアの作成」を参照してください。

e. [Network settings] (ネットワーク設定) で、次の値を設定し、他の値はデフォルトのままにします。

- [Allow SSH traffic from] (SSH トラフィックを許可) で、EC2 インスタンスへの SSH 接続のソースを選択します。

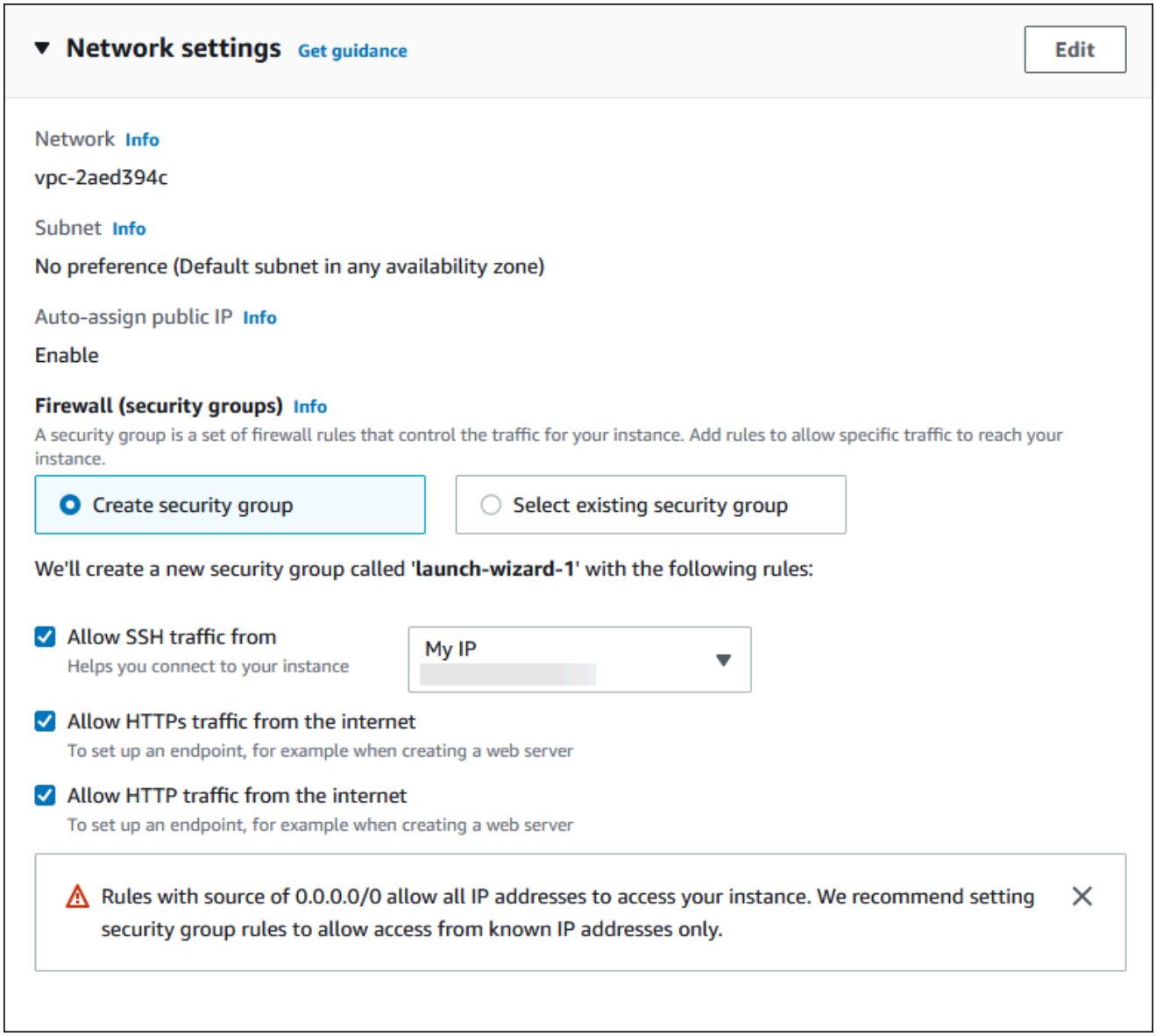
  表示された IP アドレスが SSH 接続に適している場合は、[My IP] (マイ IP)を選択できます。

  それ以外の場合は、Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は `203.0.113.25/32` です。

  多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。その場合、クライアントコンピュータが使用する IP アドレスの範囲を確認してください。

  > **Warning**
  >
  > SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリックインスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使ったインスタンスへのアクセスを限定します。

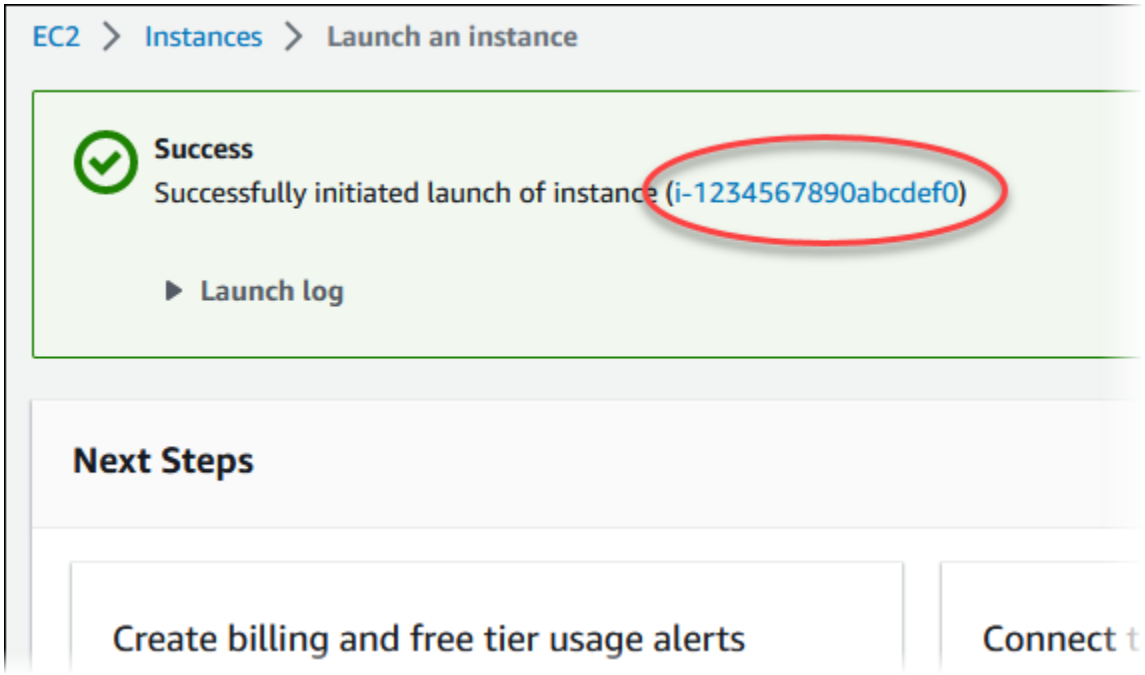
- [Allow HTTPs traffic from the internet] (インターネットからの HTTPs トラフィックを許可する) をオンにします。
- [Allow HTTP traffic from the internet] (インターネットからの HTTP トラフィックを許可する) をオンにします。



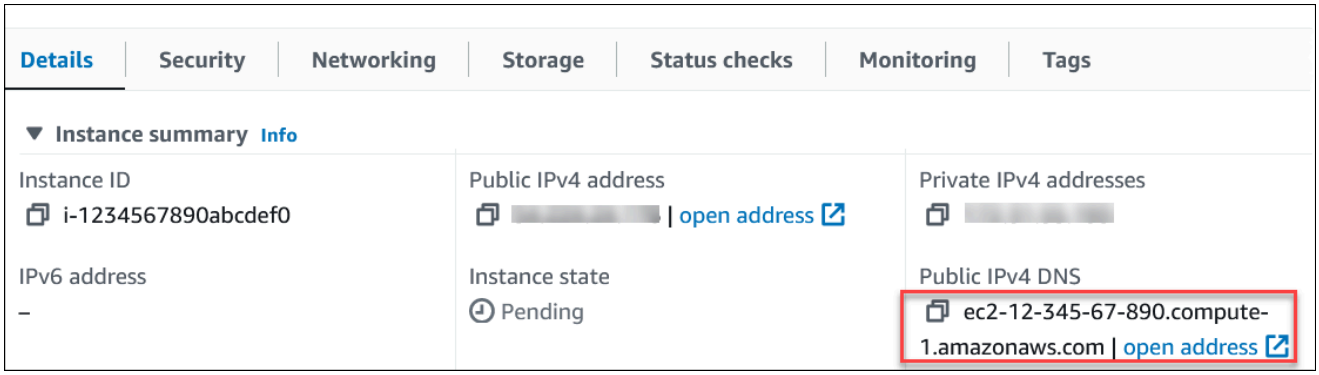
f. 残りのセクションをデフォルト値のままにします。

g. Summary (概要) パネルでインスタンス設定の要約を確認します。準備が完了したら、[Launch instance] (インスタンスを起動) を選択してください。

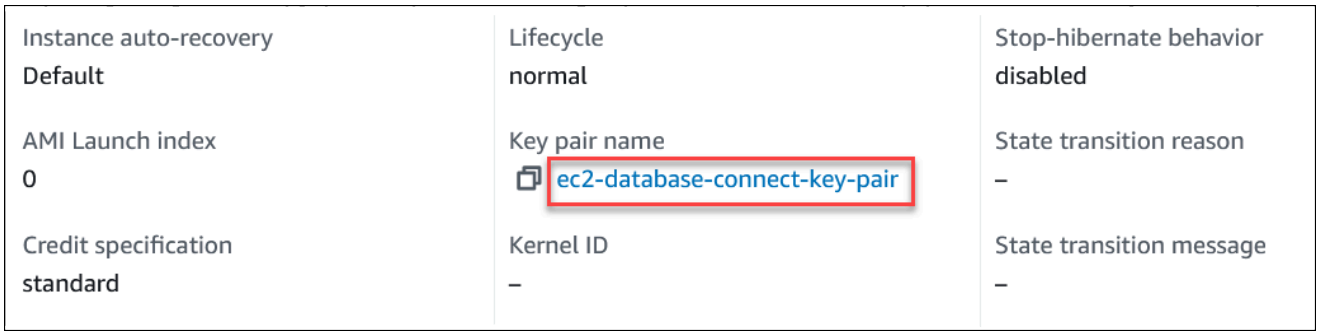
5. [起動ステータス] ページで、新しい EC2 インスタンスの ID (例: `i-1234567890abcdef0`) をメモします。



6. EC2 インスタンス ID を選択して、EC2 インスタンスのリストを開き、EC2 インスタンスを選択します。
7. [詳細] タブで、SSH を使用して接続するときに必要な次の値を書き留めます。
   a. [インスタンスの概要] で、[パブリック IPv4 DNS] の値を書き留めます。



   b. [インスタンスの詳細] で、[キーペア名] の値を書き留めます。



8. インスタンスの [Instance Status] (インスタンスのステータス) が [Running] (実行中) になるまで続行せずに待ちます。
9. Amazon RDS DB インスタンスを作成する を完了します。

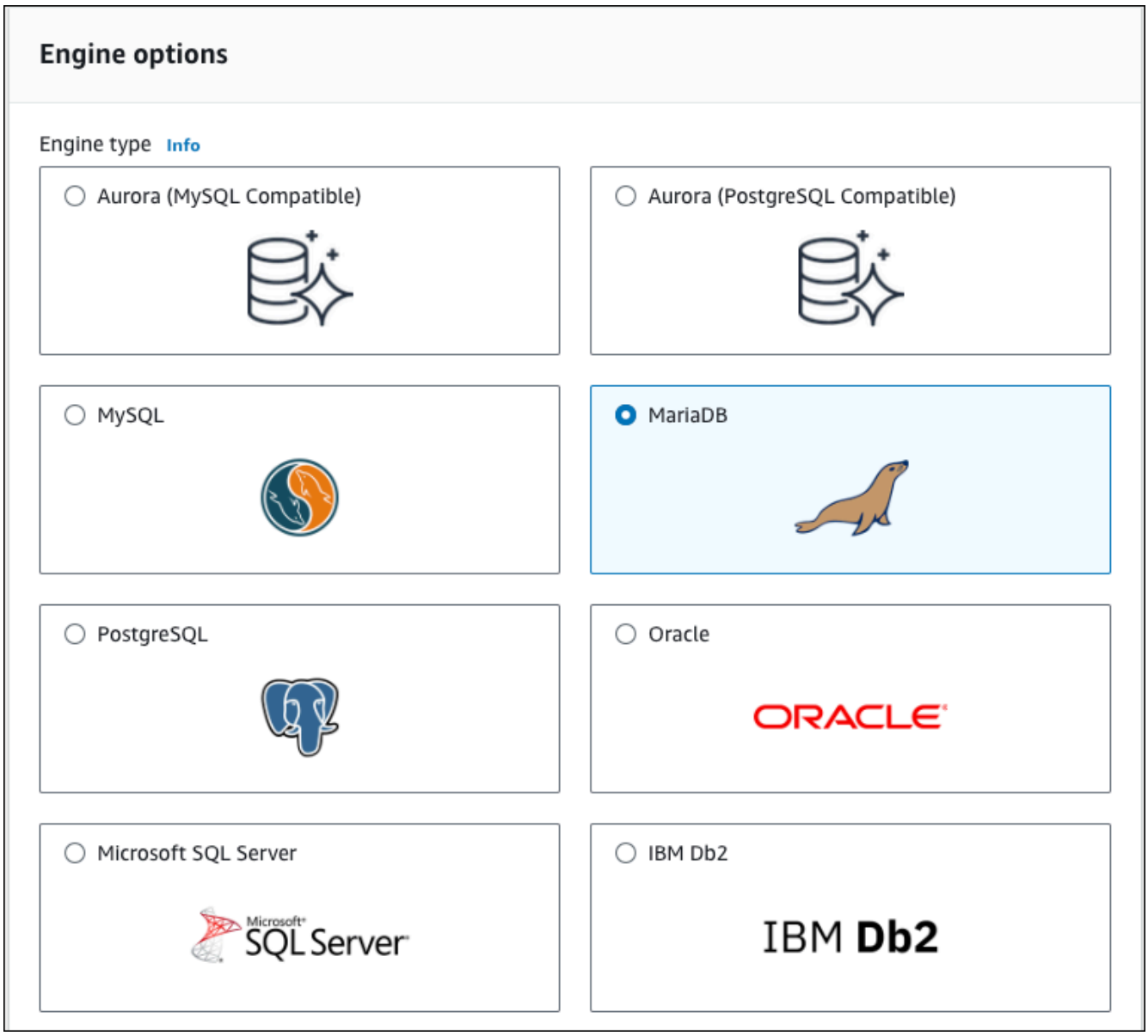
Amazon RDS DB インスタンスを作成する

ウェブアプリケーションで使用するデータを維持する RDS for MariaDB、RDS for MySQL、または RDS for PostgreSQL DB インスタンスを作成します。

RDS for MariaDB

MariaDB インスタンスを作成するには

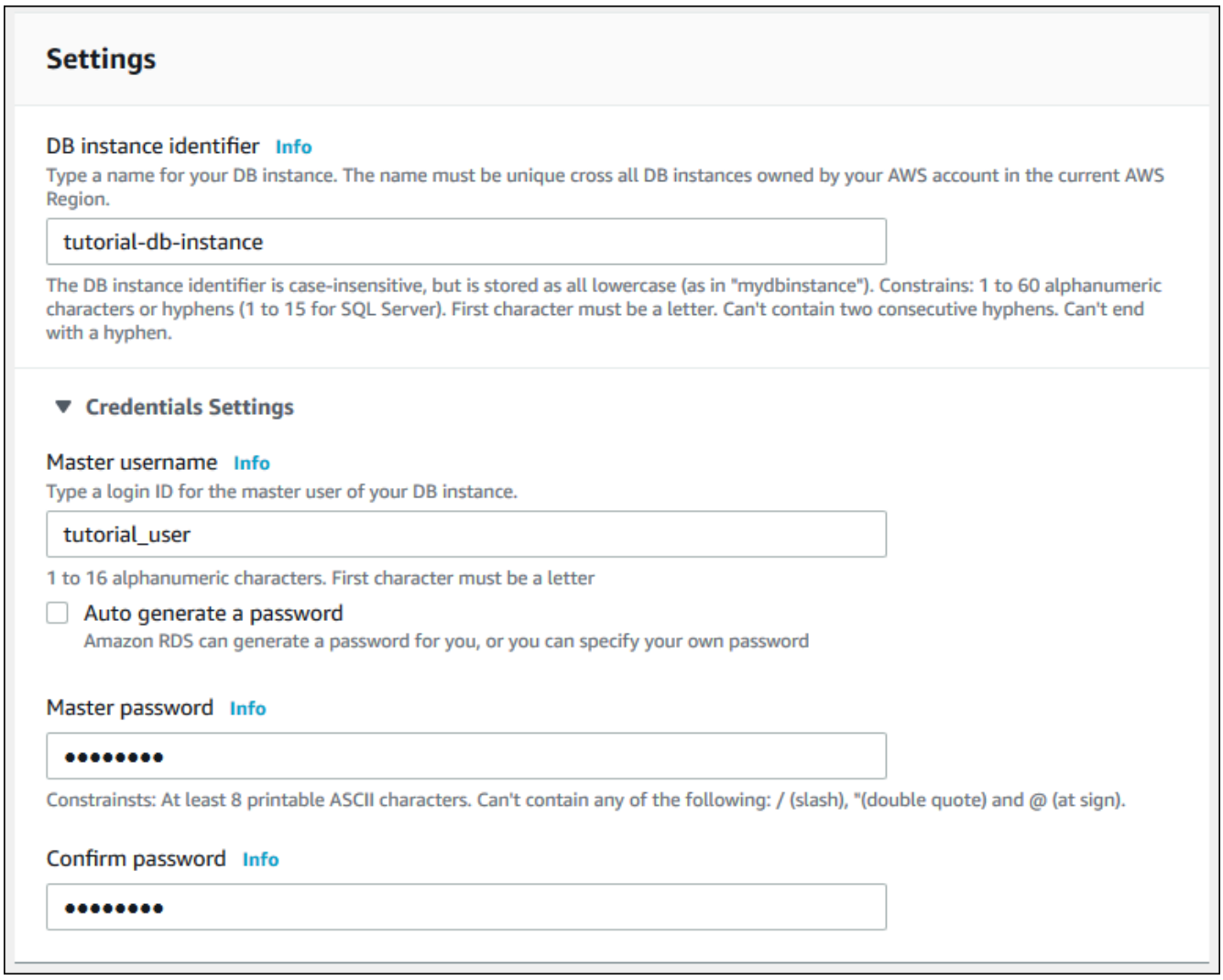
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. AWS マネジメントコンソールで、AWS リージョンを確認します。これは EC2 インスタンスを作成した場所と同じである必要があります。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。
5. [データベースの作成] ページで、[標準作成] を選択します。
6. [エンジンのオプション] で [MariaDB] を選択します。



7. [テンプレート] で、[無料利用枠] または [サンドボックス] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。

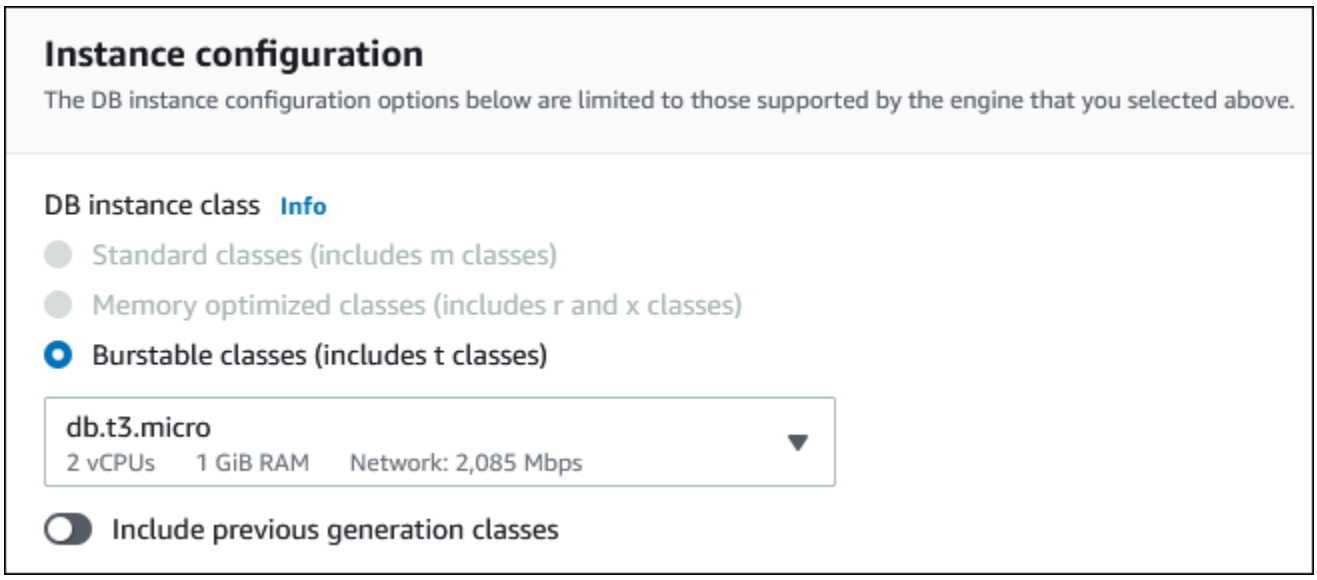
8. [可用性と耐久性] セクションで、デフォルト値を使用します。
9. [設定] セクションで、これらの値を設定します。

   - [DB instance identifier] (DB インスタンス識別子) - **`tutorial-db-instance`** を入力します。
   - [Master username] (マスターユーザー名) — **`tutorial_user`** を入力します。
   - [Auto generate a password] (パスワードの自動生成) — オプションはオフのままにします。
   - [Master password] (マスターパスワード): パスワードを入力します。
   - [パスワードの確認] - パスワードを再入力します。

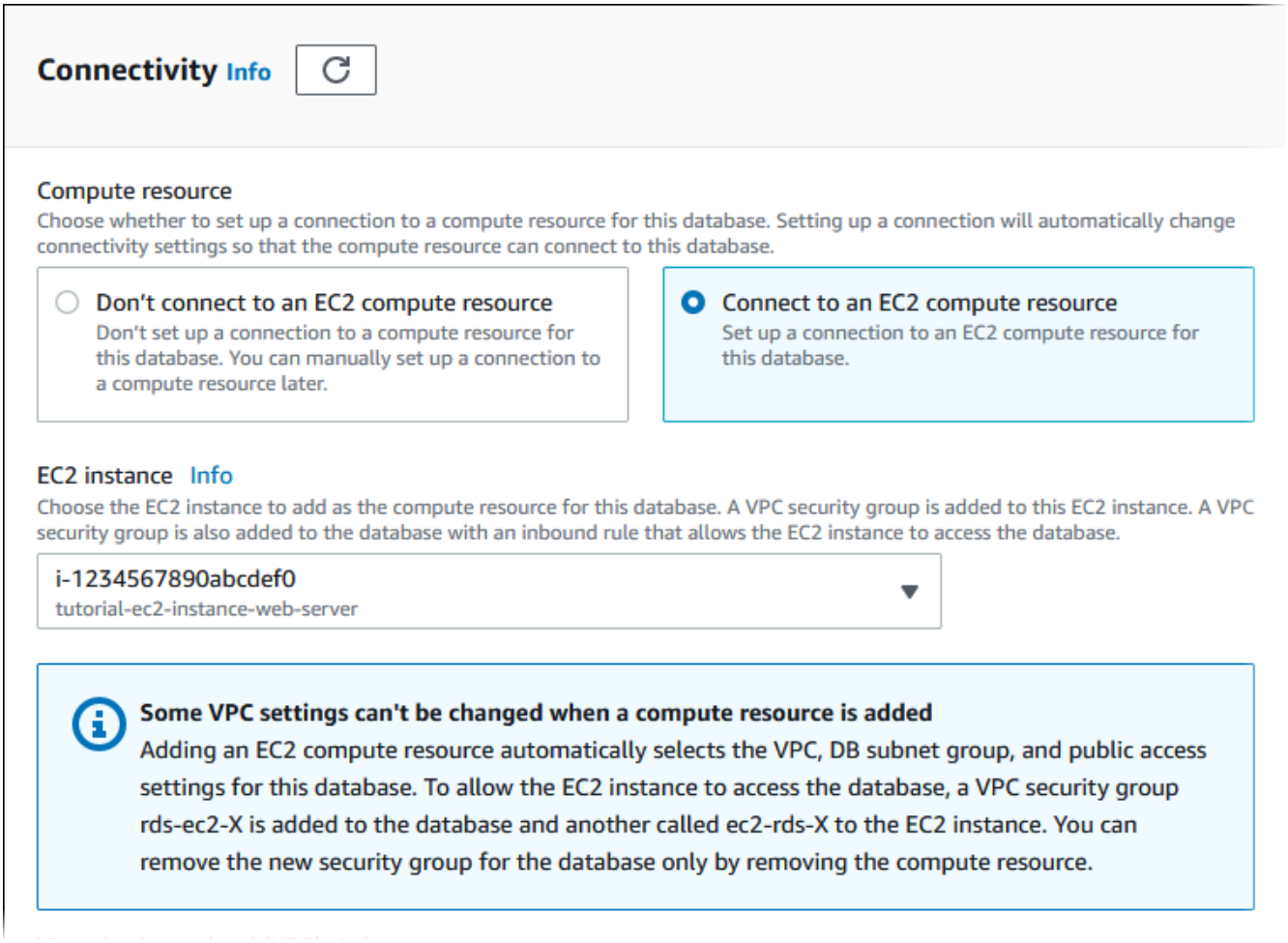


10. [Instance configuration] (インスタンス設定) セクションで、次の値を設定します。

    - バースト可能クラス (t クラスを含む)
    - db.t3.micro

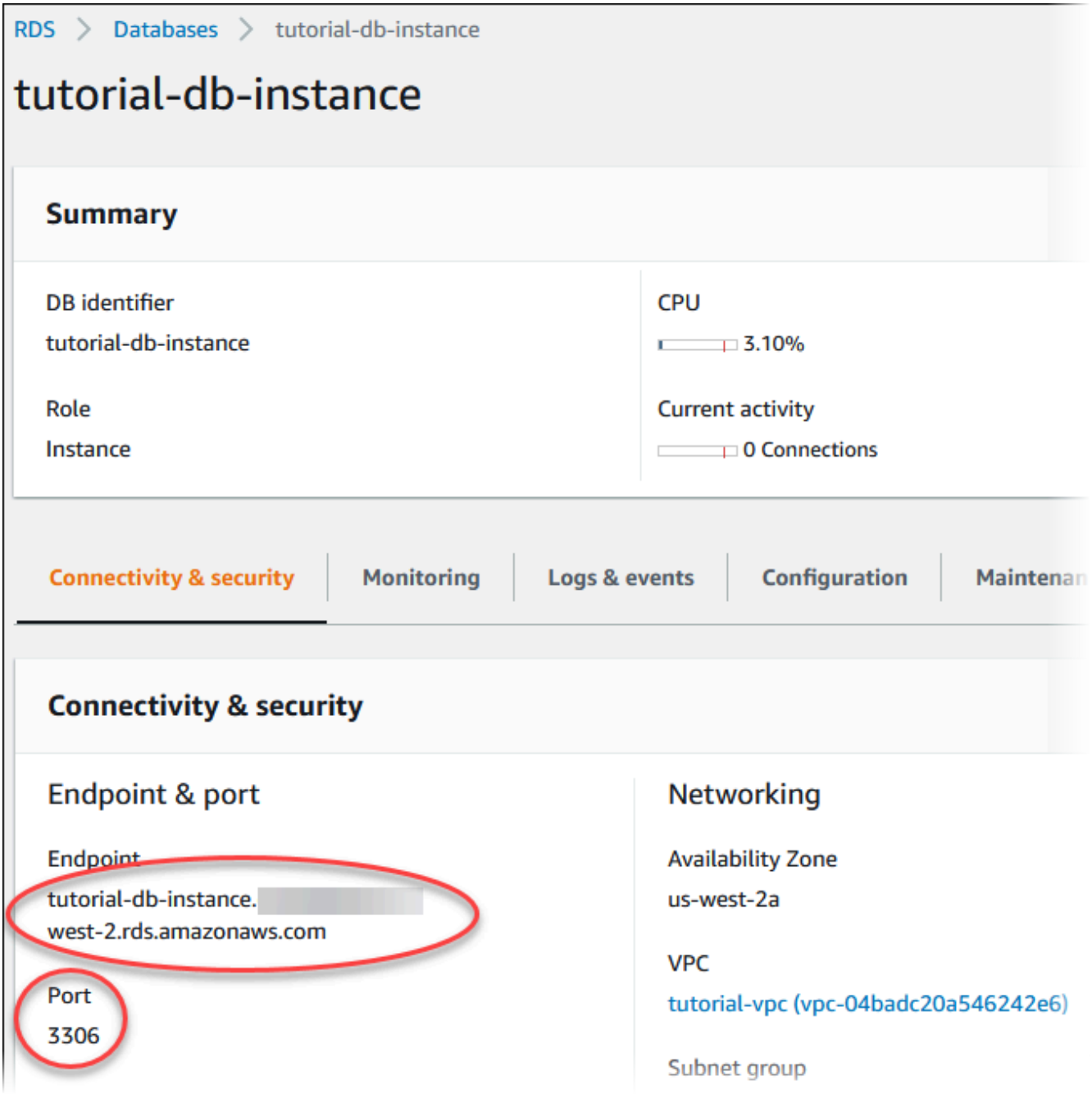


11. [Storage] (ストレージ) セクションでは、デフォルトのままにします。
12. [Connectivity] (接続) セクションで、次の値を設定し、他の値はデフォルトのままにします。
    - [Compute resource] (コンピューティングリソース) で、選択してください[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択します。
    - [EC2 instance] (EC2 インスタンス) で、以前に作成した [tutorial-ec2-instance-web-server] などの EC2 インスタンスを選択します。



13. [データベース認証] セクションで、[パスワード認証] が選択されていることを確認します。
14. [Additional configuration (追加設定)] セクションを開き、[Initial database name (初期データベース名)] に **`sample`** と入力します。その他のオプションについては、デフォルト設定のままにしておきます。
15. MariaDB インスタンスを作成するには、[データベースを作成] を選択します。

    新しい DB インスタンスは、[作成中] ステータスとして [データベース] リストに表示されます。
16. 新しい DB インスタンスの [ステータス] が [Available] と表示されるまで待ちます。詳細を表示する DB インスタンスの名前を選択します。
17. [接続とセキュリティ] セクションで、DB インスタンスの [エンドポイント] および [ポート] を表示します。



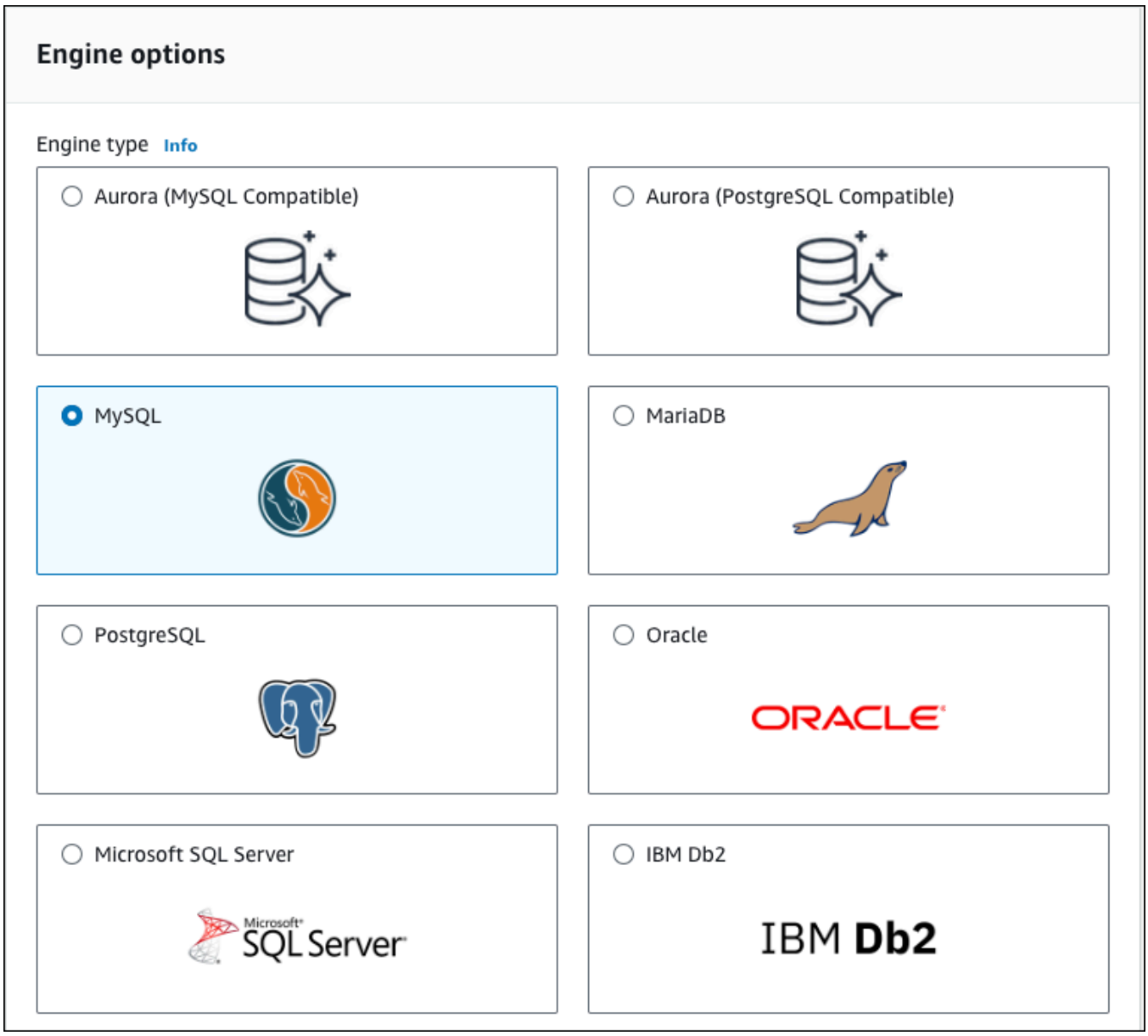
DB インスタンスのエンドポイントとポートを書き留めます。この情報を使用して、ウェブサーバーを DB インスタンスに接続します。

18. EC2 インスタンスにウェブサーバーをインストールします を完了します。

RDS for MySQL

MySQL DB インスタンスを作成するには

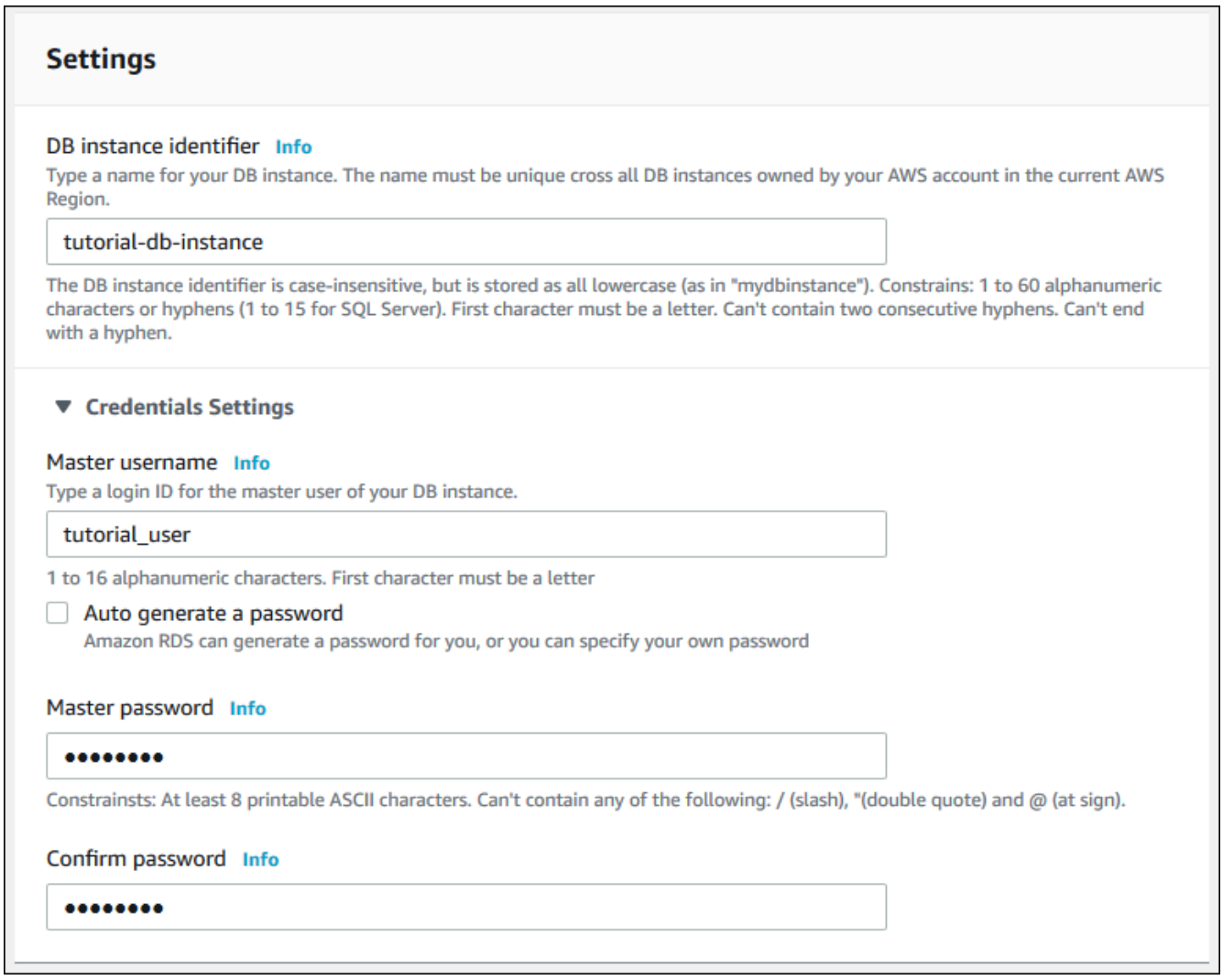
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. AWS マネジメントコンソールで、AWS リージョンを確認します。これは EC2 インスタンスを作成した場所と同じである必要があります。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。
5. [データベースの作成] ページで、[標準作成] を選択します。
6. [エンジンのオプション] として、[MySQL] を選択します。



7. [テンプレート] で、[無料利用枠] または [サンドボックス] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。

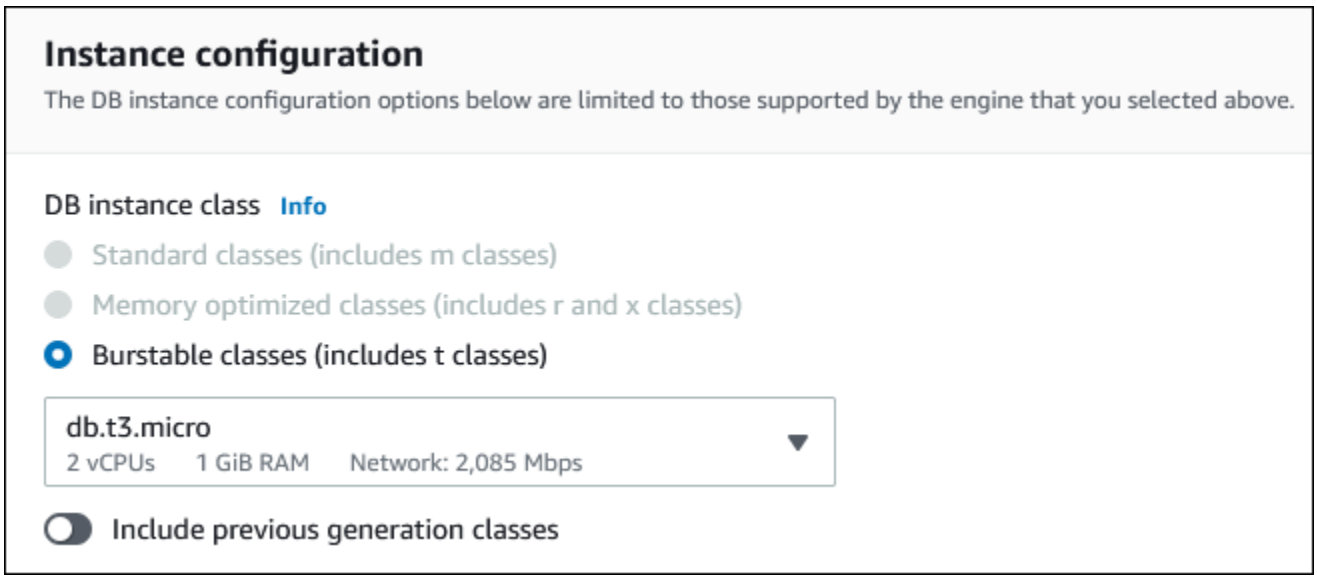
8. [可用性と耐久性] セクションで、デフォルト値を使用します。
9. [設定] セクションで、これらの値を設定します。

   - [DB instance identifier] (DB インスタンス識別子) - **`tutorial-db-instance`** を入力します。
   - [Master username] (マスターユーザー名) — **`tutorial_user`** を入力します。
   - [Auto generate a password] (パスワードの自動生成) — オプションはオフのままにします。
   - [Master password] (マスターパスワード): パスワードを入力します。
   - [パスワードの確認] - パスワードを再入力します。

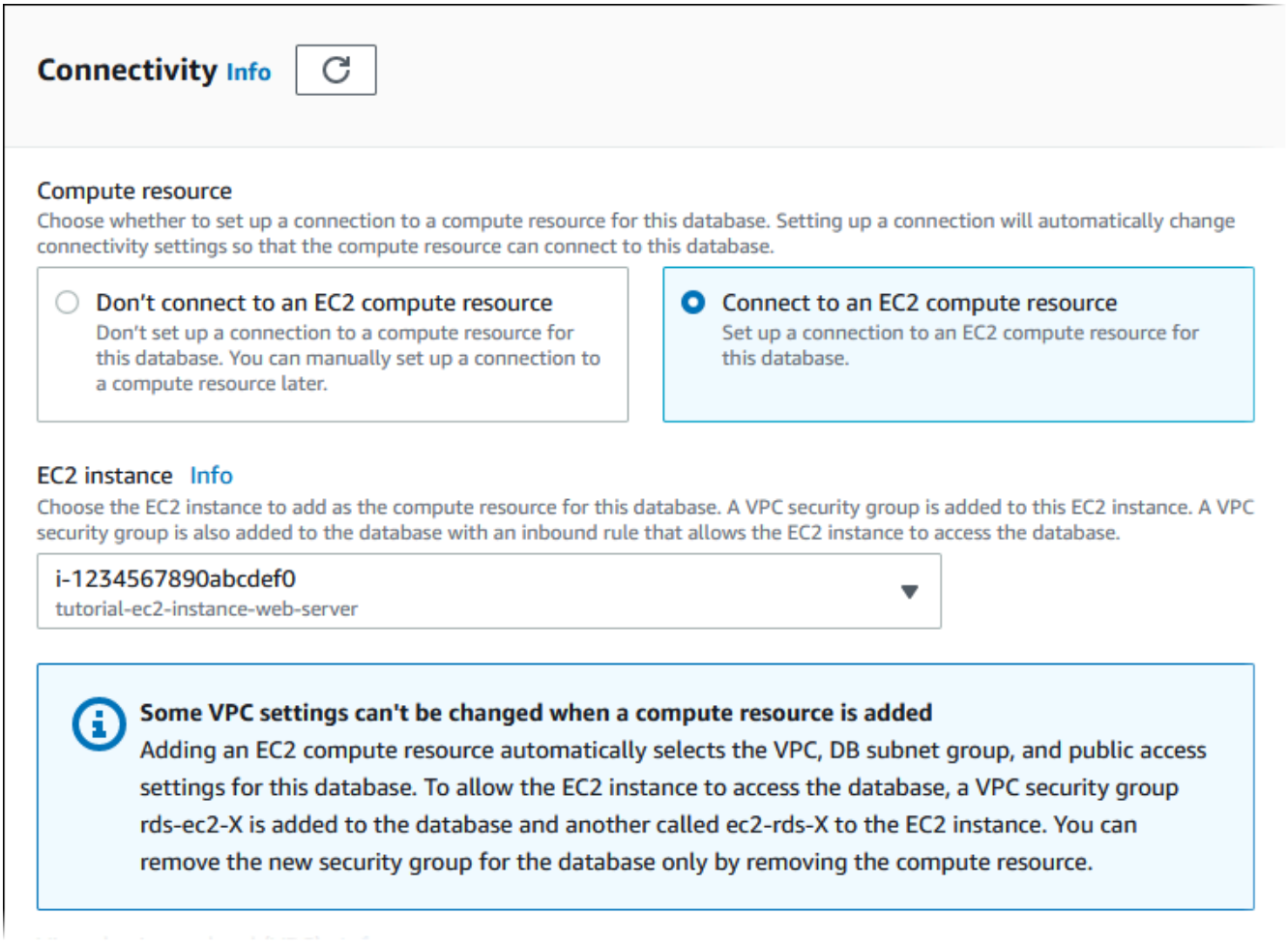


10. [Instance configuration] (インスタンス設定) セクションで、次の値を設定します。

    - バースト可能クラス (t クラスを含む)
    - db.t3.micro

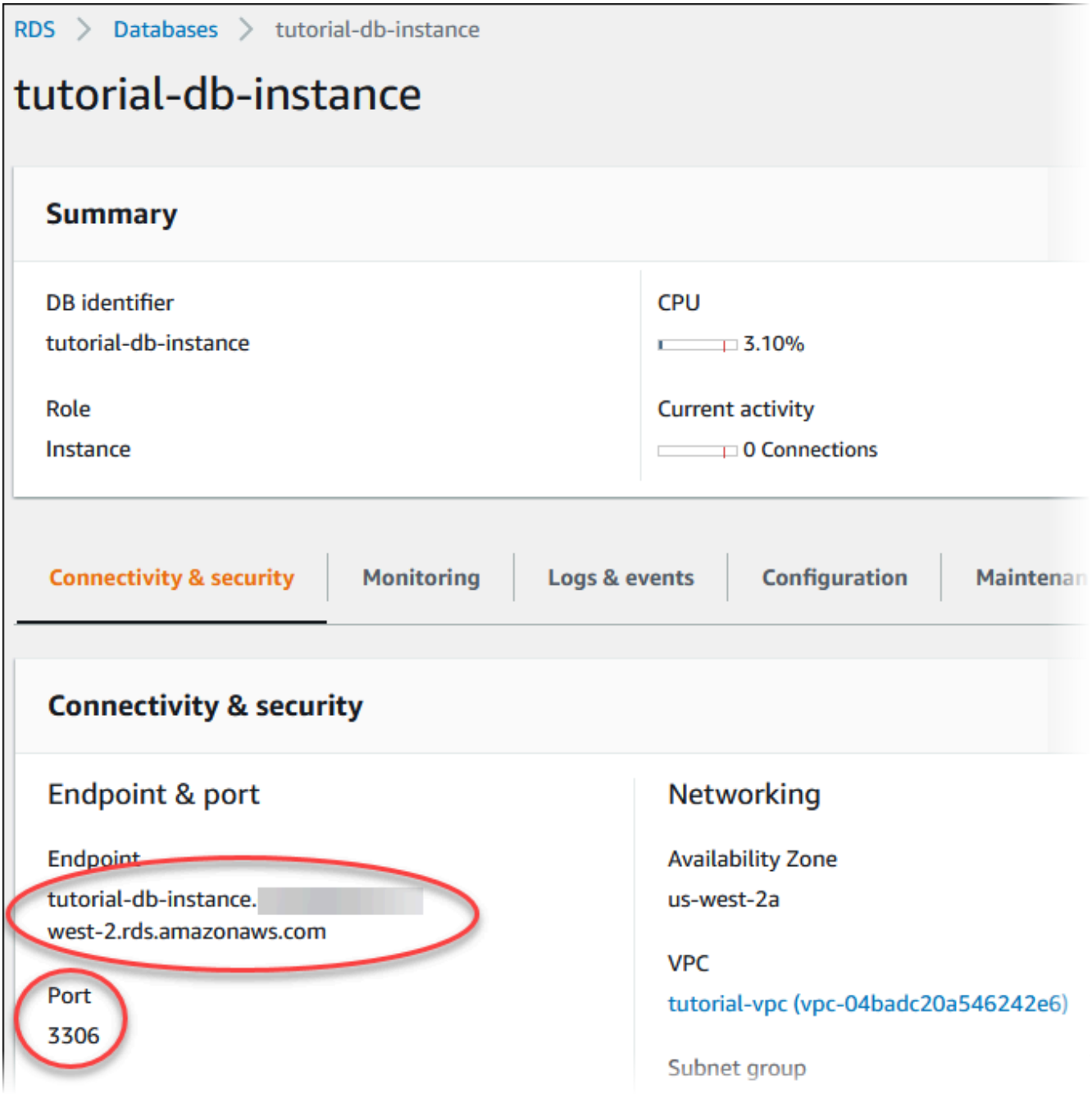


11. [Storage] (ストレージ) セクションでは、デフォルトのままにします。
12. [Connectivity] (接続) セクションで、次の値を設定し、他の値はデフォルトのままにします。
    - [Compute resource] (コンピューティングリソース) で、選択してください[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択します。
    - [EC2 instance] (EC2 インスタンス) で、以前に作成した [tutorial-ec2-instance-web-server] などの EC2 インスタンスを選択します。



13. [データベース認証] セクションで、[パスワード認証] が選択されていることを確認します。
14. [Additional configuration (追加設定)] セクションを開き、[Initial database name (初期データベース名)] に **`sample`** と入力します。その他のオプションについては、デフォルト設定のままにしておきます。
15. MySQL DB インスタンスを作成するには、[Create database (データベースの作成)] を選択します。

    新しい DB インスタンスは、[作成中] ステータスとして [データベース] リストに表示されます。
16. 新しい DB インスタンスの [ステータス] が [Available] と表示されるまで待ちます。詳細を表示する DB インスタンスの名前を選択します。
17. [接続とセキュリティ] セクションで、DB インスタンスの [エンドポイント] および [ポート] を表示します。



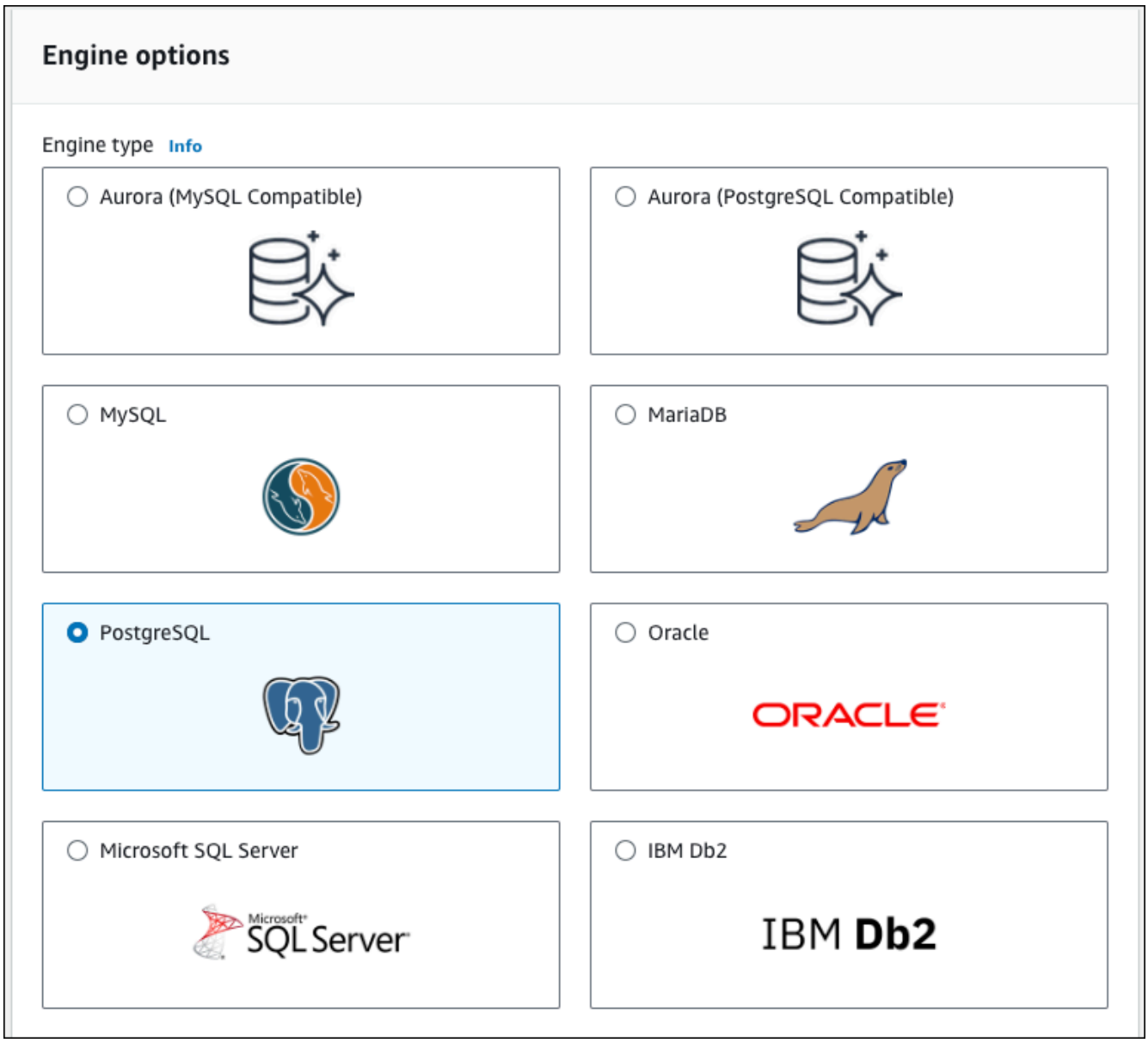
DB インスタンスのエンドポイントとポートを書き留めます。この情報を使用して、ウェブサーバーを DB インスタンスに接続します。

18. EC2 インスタンスにウェブサーバーをインストールします を完了します。

RDS for PostgreSQL

PostgreSQL DB インスタンスを作成するには

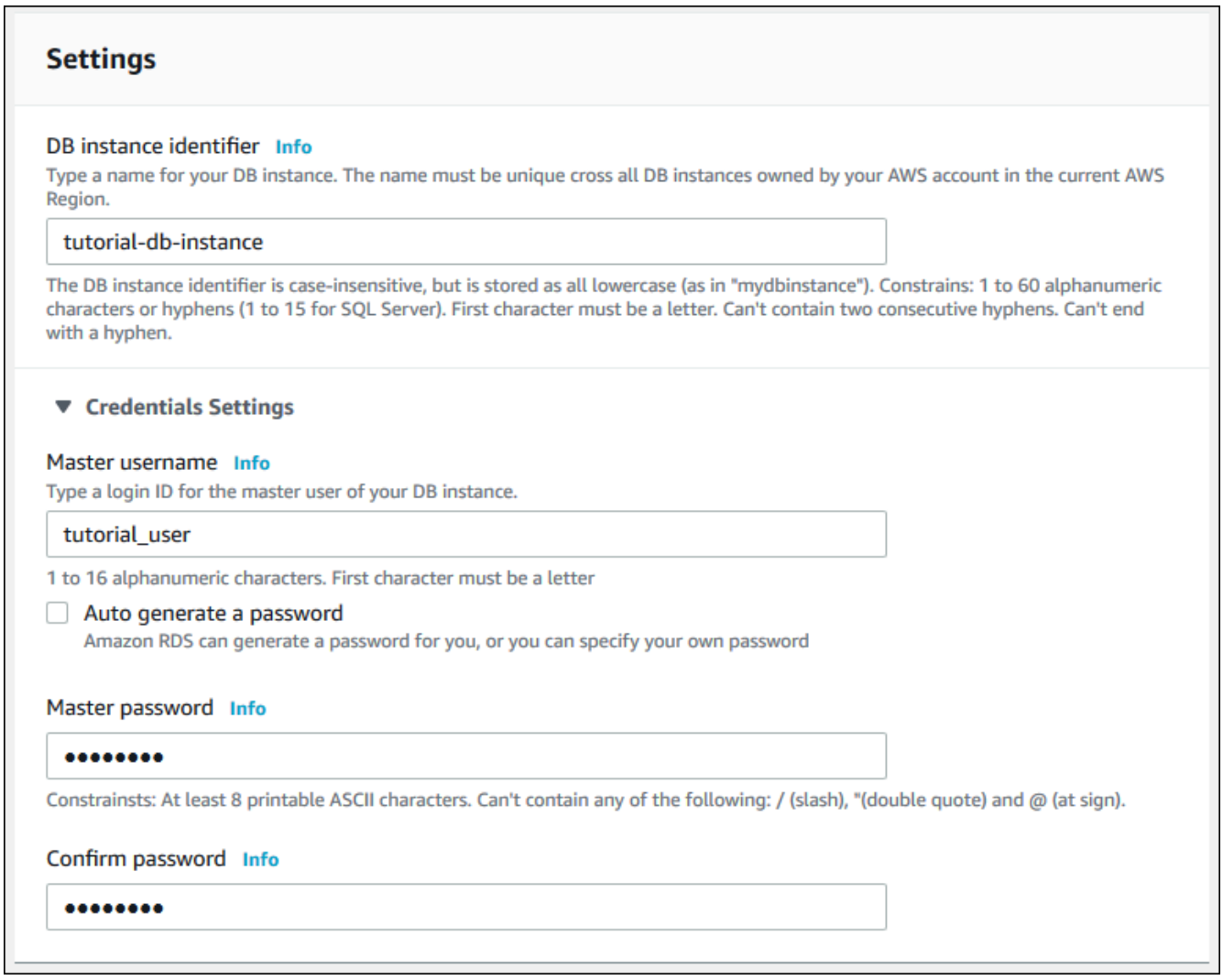
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. AWS マネジメントコンソールで、AWS リージョンを確認します。これは EC2 インスタンスを作成した場所と同じである必要があります。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。
5. [データベースの作成] ページで、[標準作成] を選択します。
6. [Engine options] (エンジンオプション) で PostgreSQL を選択します。



7. [テンプレート] で、[無料利用枠] または [サンドボックス] を選択します。[無料利用枠] は無料プランアカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。

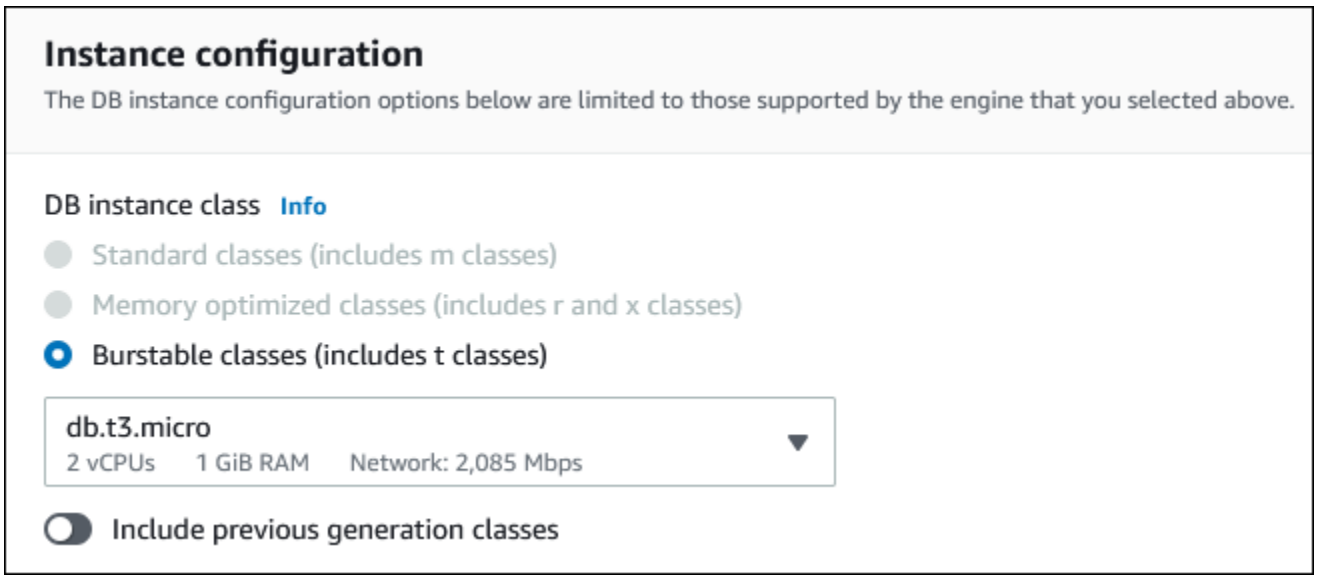
8. [可用性と耐久性] セクションで、デフォルト値を使用します。
9. [設定] セクションで、これらの値を設定します。

   - [DB instance identifier] (DB インスタンス識別子) - **`tutorial-db-instance`** を入力します。
   - [Master username] (マスターユーザー名) — **`tutorial_user`** を入力します。
   - [Auto generate a password] (パスワードの自動生成) — オプションはオフのままにします。
   - [Master password] (マスターパスワード): パスワードを入力します。
   - [パスワードの確認] - パスワードを再入力します。

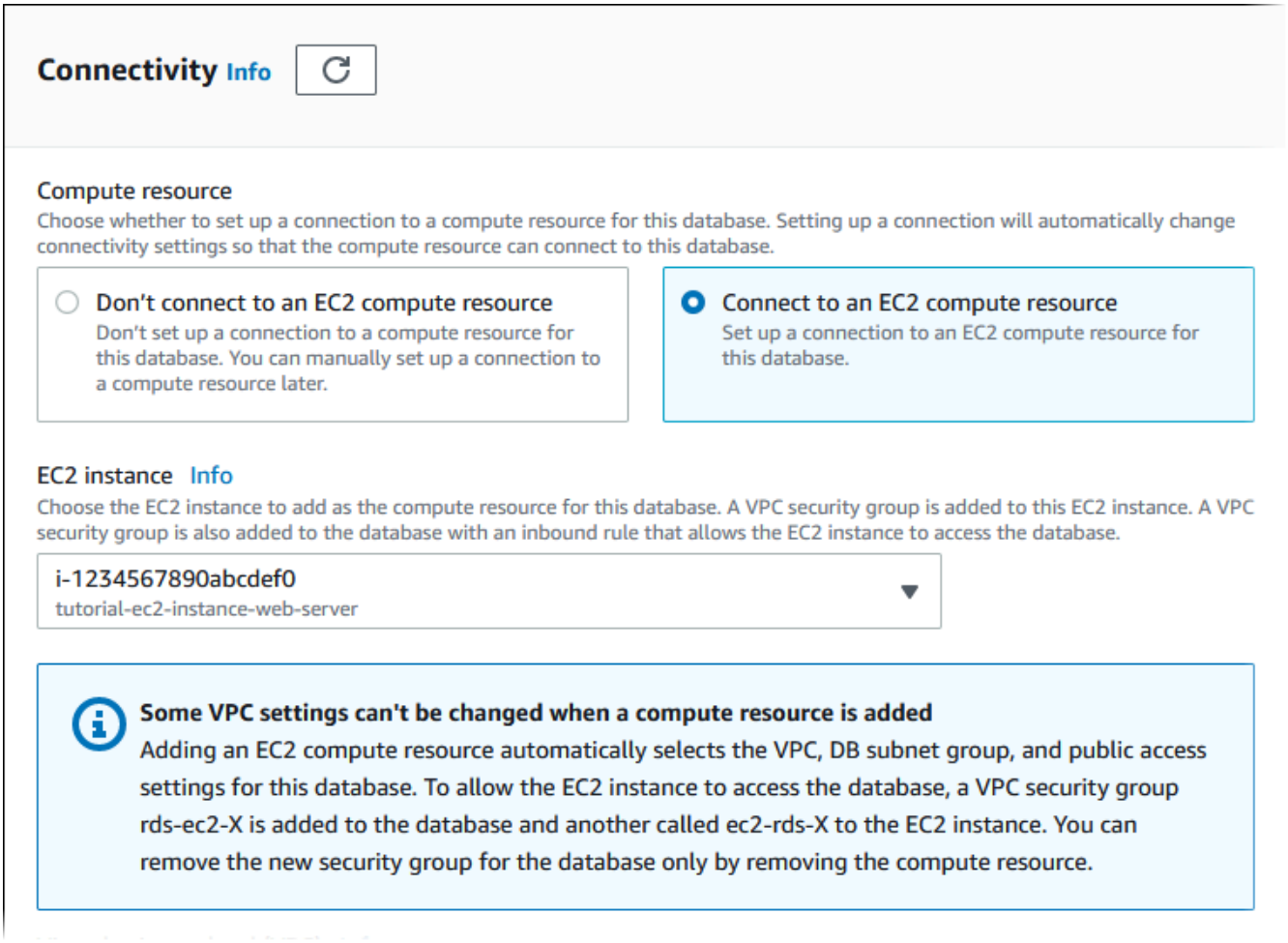


10. [Instance configuration] (インスタンス設定) セクションで、次の値を設定します。

    - バースト可能クラス (t クラスを含む)
    - db.t3.micro

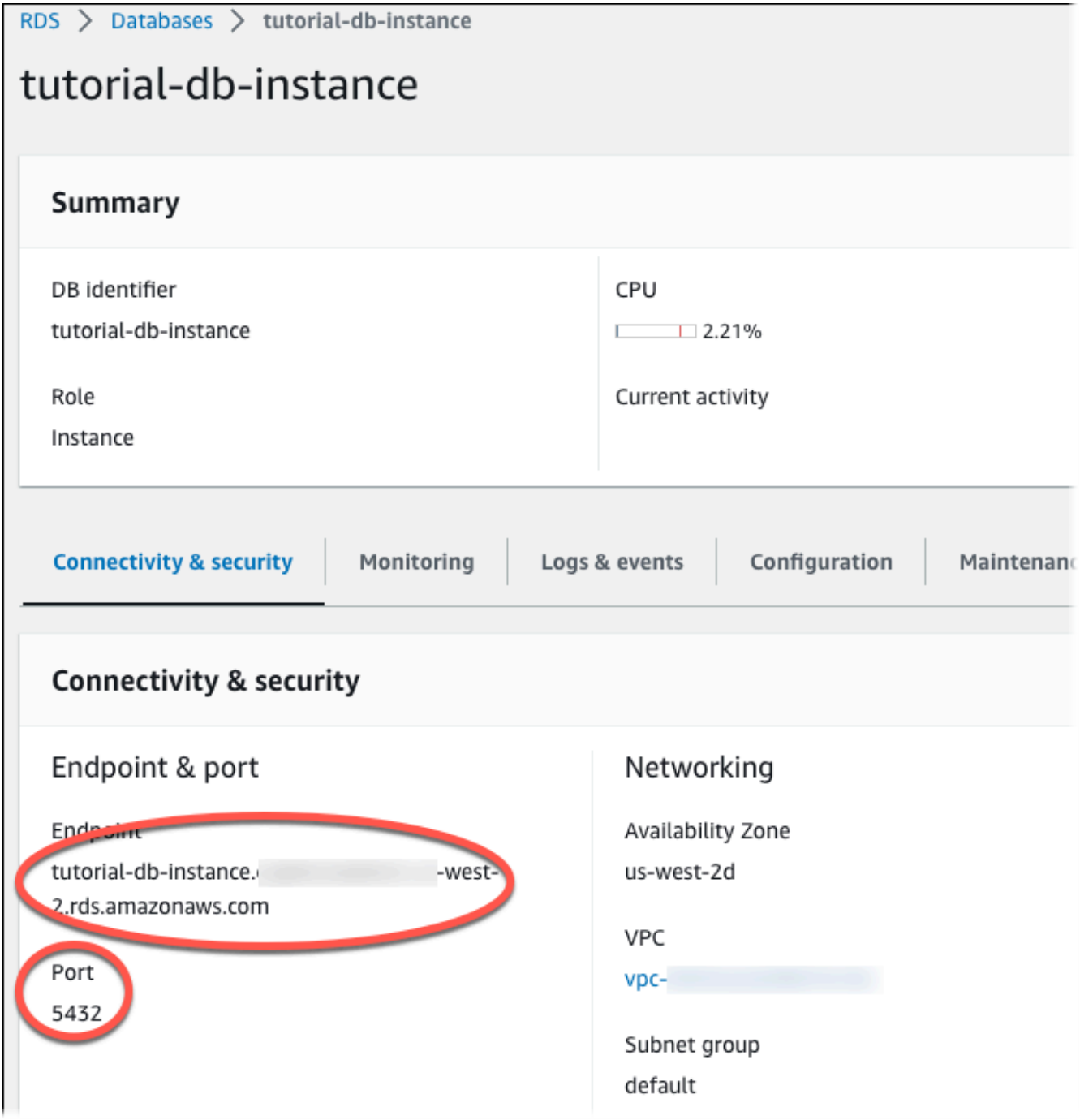


11. [Storage] (ストレージ) セクションでは、デフォルトのままにします。
12. [Connectivity] (接続) セクションで、次の値を設定し、他の値はデフォルトのままにします。
    - [Compute resource] (コンピューティングリソース) で、選択してください[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択します。
    - [EC2 instance] (EC2 インスタンス) で、以前に作成した [tutorial-ec2-instance-web-server] などの EC2 インスタンスを選択します。



13. [データベース認証] セクションで、[パスワード認証] が選択されていることを確認します。
14. [Additional configuration (追加設定)] セクションを開き、[Initial database name (初期データベース名)] に **`sample`** と入力します。その他のオプションについては、デフォルト設定のままにしておきます。
15. PostgreSQL DB インスタンスを作成するには、[データベースの作成] を選択します。

    新しい DB インスタンスは、[作成中] ステータスとして [データベース] リストに表示されます。
16. 新しい DB インスタンスの [ステータス] が [Available] と表示されるまで待ちます。詳細を表示する DB インスタンスの名前を選択します。
17. [接続とセキュリティ] セクションで、DB インスタンスの [エンドポイント] および [ポート] を表示します。



DB インスタンスのエンドポイントとポートを書き留めます。この情報を使用して、ウェブサーバーを DB インスタンスに接続します。

18. EC2 インスタンスにウェブサーバーをインストールします を完了します。

EC2 インスタンスにウェブサーバーをインストールします

EC2 インスタンスを起動して DB インスタンスに接続する で作成した EC2 インスタンスにウェブサーバーをインストールします。ウェブサーバーは、Amazon RDS DB インスタンスを作成する で作成した Amazon RDS DB インスタンスに接続します。

PHP と MariaDB を使用する Apache ウェブサーバーのインストール

EC2 インスタンスに接続し、ウェブサーバーをインストールします。

EC2 インスタンスに接続し、PHP を使用して Apache ウェブサーバーをインストールするには

1. 「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」のステップに従って、先ほど作成した EC2 インスタンスに接続します。

   SSH を使用して EC2 インスタンスに接続することをお勧めします。SSH クライアントユーティリティが Windows、Linux、または Mac にインストールされている場合は、次のコマンド形式でインスタンスに接続できます。

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   例えば、`ec2-database-connect-key-pair.pem` が Linux の `/dir1` に保存されていて、EC2 インスタンスのパブリック IPv4 DNS が `ec2-12-345-678-90.compute-1.amazonaws.com` であるとします。SSH コマンドは次のようになります。

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

2. EC2 インスタンスのソフトウェアを更新して、最新のバグ修正とセキュリティ更新を入手します。これを行うには、次のコマンドを使用します。

   > **Note**
   >
   > -y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を確認するには、このオプションを省略します。

   ```
   sudo dnf update -y
   ```

3. 更新が完了した後、次のコマンドを使用して、Apache ウェブサーバー、PHP、および MariaDB または PostgreSQL ソフトウェアをインストールします。このコマンドは、複数のソフトウェアパッケージおよび関連する依存関係を同時にインストールします。

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

   エラーが表示された場合、インスタンスはおそらく Amazon Linux 2023 AMI で起動していません。代わりに Amazon Linux 2 AMI を使用している可能性があります。次のコマンドを使用して、Amazon Linux のバージョンを表示できます。

   ```
   cat /etc/system-release
   ```

   詳細については、「インスタンスソフトウェアの更新」を参照してください。

4. 次に示すコマンドを使用してウェブサーバーを起動します。

   ```
   sudo systemctl start httpd
   ```

   ウェブサーバーが正しくインストールされ、起動されているかどうかをテストできます。これを行うには、ウェブブラウザのアドレスバーに EC2 インスタンスのパブリックドメインネームシステム (DNS) 名を入力します (例: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`)。ウェブサーバーが実行中の場合、Apache テストページが表示されます。

   Apache テストページが表示されない場合は、チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用) で作成した VPC セキュリティグループのインバウンドルールを確認してください。インバウンドルールに、ウェブサーバーへの接続に使用する IP アドレスに対する HTTP (ポート 80) アクセスを許可するルールが含まれていることを確認します。

> **Note**
>
> Apache テストページは、ドキュメントのルートディレクトリ /var/www/html にコンテンツがないときのみ表示されます。ドキュメントルートディレクトリにコンテンツを追加すると、そのコンテンツが EC2 インスタンスのパブリック DNS アドレスに表示されます。これより前では、Apache のテストページに表示されます。

5. systemctl コマンドを使用して、システムがブートするたびにウェブサーバーが起動するように設定します。

```
sudo systemctl enable httpd
```

ec2-user が Apache ウェブサーバーのデフォルトルートディレクトリにあるファイルを管理できるようにするには、/var/www ディレクトリの所有権とアクセス許可を変更します。このタスクを行うには、複数の方法があります。このチュートリアルでは、ec2-user を apache グループに追加し、apache ディレクトリの所有権を /var/www グループに付与し、グループへの書き込み権限を割り当てます。

Apache ウェブサーバーのファイルアクセス許可を設定するには

1. ec2-user ユーザーを apache グループに追加します。

```
sudo usermod -a -G apache ec2-user
```

2. ログアウトしてアクセス許可を更新して新しい apache グループを含めます。

```
exit
```

3. 再度ログインし、apache コマンドを使用して groups グループが存在していることを確認します。

```
groups
```

出力は次のようになります。

```
ec2-user adm wheel apache systemd-journal
```

4. /var/www ディレクトリとそのコンテンツのグループ所有権を apache グループに変更します。

```
sudo chown -R ec2-user:apache /var/www
```

5. /var/www およびそのサブディレクトリのディレクトリアクセス許可を変更して、グループの書き込みアクセス許可を追加し、後で作成するサブディレクトリのグループ ID を設定します。

```
sudo chmod 2775 /var/www
find /var/www -type d -exec sudo chmod 2775 {} \;
```

6. /var/www ディレクトリおよびそのサブディレクトリのファイルごとにアクセス許可を繰り返し変更し、グループの書き込みアクセス許可を追加します。

```
find /var/www -type f -exec sudo chmod 0664 {} \;
```

これで、ec2-user (および apache グループの将来のメンバー) は、Apache ドキュメントルートにファイルを追加、削除、編集できるようになります。これにより、静的ウェブサイトや PHP アプリケーションなどのコンテンツを追加できます。

> **Note**
>
> HTTP プロトコルを実行するウェブサーバーは、送受信したデータのトランスポートセキュリティを提供しません。ウェブブラウザを使用して HTTP サーバーに接続すると、ネットワーク経路上のどこからでも、多くの情報が誰でも閲覧できるようになります。この情報には、アクセスした URL、受信したウェブページのコンテンツ、HTML フォームの内容 (パスワードなど) が含まれます。
> ウェブサーバーを保護するためのベストプラクティスとして、HTTPS (HTTP Secure) のサポートをインストールしてください。このプロトコルでは、SSL/TLS を使用してデータを保護します。詳細については、Amazon EC2 ユーザーガイドの「チュートリアル: Amazon Linux AMI を使用した SSL/TLS の設定」を参照してください。

Apache ウェブサーバーを DB インスタンスに接続する

次に、Amazon RDS DB インスタンスに接続する Apache ウェブサーバーへのコンテンツを追加します。

DB インスタンスに接続する Apache ウェブサーバーにコンテンツを追加するには

1. EC2 インスタンスに接続したまま、ディレクトリを /var/www に変更し、inc という名前の新しいサブディレクトリを作成します。

```
cd /var/www
mkdir inc
cd inc
```

2. inc という dbinfo.inc ディレクトリに新しいファイルを作成し、nano を呼び出して (または任意のテキストエディタで) ファイルを編集します。

```
>dbinfo.inc
nano dbinfo.inc
```

3. dbinfo.inc ファイルに次のコンテンツを追加します。ここで、*db_instance_endpoint* は、DB インスタンス の、ポートのない、DB インスタンスエンドポイントです。

> **Note**
>
> ユーザー名とパスワード情報は、ウェブサーバーのドキュメントルートに含まれていないフォルダに配置することをお勧めします。これにより、セキュリティ情報が公開される可能性が低くなります。
> アプリケーションで master password を適切なパスワードに変更してください。

```
<?php

define('DB_SERVER', 'db_instance_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

4. dbinfo.inc ファイルを保存して閉じます。nano を使用している場合は、Ctrl+S と Ctrl+X を使用してファイルを保存して閉じます。

5. ディレクトリを /var/www/html に変更します。

```
cd /var/www/html
```

6. html という SamplePage.php ディレクトリに新しいファイルを作成し、nano を呼び出して (または任意のテキストエディタで) ファイルを編集します。

```
>SamplePage.php
nano SamplePage.php
```

7. SamplePage.php ファイルに次のコンテンツを追加します。

   MariaDB & MySQL

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
```

```
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
```

```
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");
```

```
  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
<?php

/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
```

```
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
```

```
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;
```

```
}
?>
```

8. SamplePage.php ファイルを保存して閉じます。
9. ウェブブラウザを開いて http://*EC2 instance endpoint*/SamplePage.php (例: http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php) を参照することで、ウェブサーバーが正常に DB インスタンスに接続していることを確認します。

SamplePage.php を使用して、DB インスタンスにデータを追加できます。これで、追加したデータがこのページに表示されます。データがテーブルに挿入されたことを確認するには、Amazon EC2 インスタンスに MySQL をインストールします。その後、DB インスタンスに接続し、テーブルにクエリを実行します。

MySQL クライアントをインストールして DB インスタンスに接続する方法については、MySQL DB インスタンスへの接続 を参照してください。

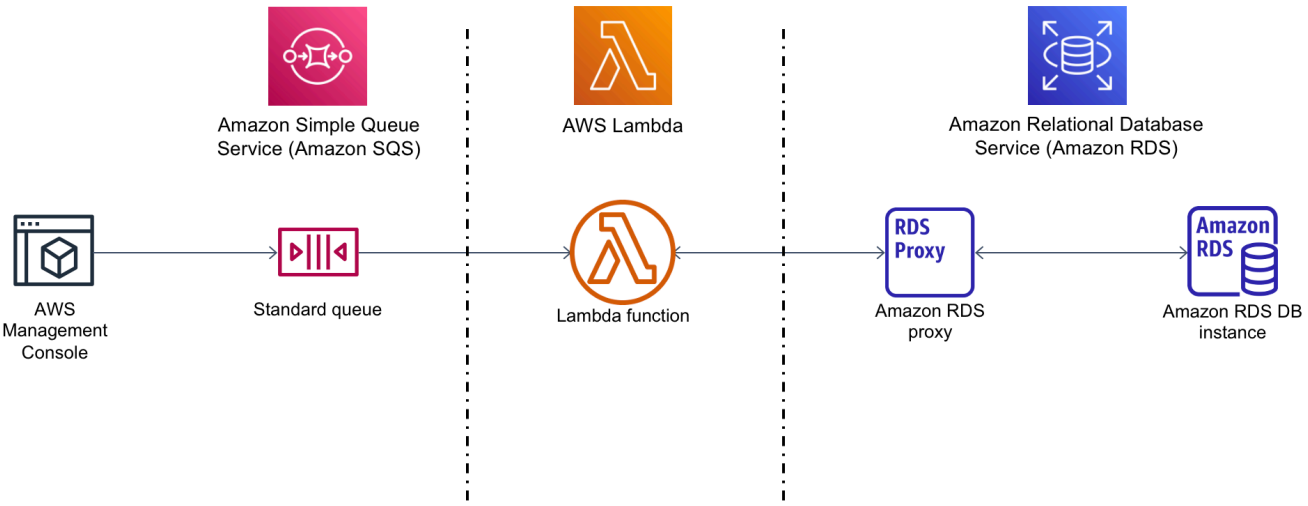
DB インスタンス の安全性を高めるために、VPC の外部にある出典が DB インスタンスに接続できないことを確認します。

ウェブサーバーとデータベースのテストが完了したら、DB インスタンスと Amazon EC2 インスタンスを削除する必要があります。

- DBインスタンスを削除するには、DB インスタンスを削除するの手順に従います。最終的なスナップショットを作成する必要はありません。
- Amazon EC2 インスタンスを終了するには、Amazon EC2 ユーザーガイドの「インスタンスの終了」の手順に従います。

チュートリアル: Lambda 関数を使用して Amazon RDS にアクセスする

このチュートリアルでは、Lambda 関数を使用して、Amazon Relational Database Service (Amazon RDS) データベースに RDS プロキシ経由でデータを書き込みます。Lambda 関数は、メッセージが追加されるたびに Amazon Simple Queue Service (Amazon SQS) キューからレコードを読み取り、データベース内のテーブルに新しい項目を書き込みます。この例では、AWS マネジメントコンソール を使用してキューにメッセージを手動で追加します。次の図は、チュートリアルを完了するために使用する AWS リソースを示しています。



Amazon RDS では、Microsoft SQL Server、MariaDB、MySQL、Oracle Database、PostgreSQL などの一般的なデータベース製品を使用して、マネージドリレーショナルデータベースをクラウドで実行できます。Lambda を使用してデータベースにアクセスすると、ウェブサイトに新規顧客を登録するなどのイベントに応じてデータを読み書きできます。関数、データベースインスタンス、およびプロキシは、需要の高い時期に合わせて自動的にスケーリングされます。

このチュートリアルを完了するには、次のタスクを実行します。

1. RDS for MySQL データベースインスタンスとプロキシを AWS アカウント のデフォルト VPC で起動します。
2. データベースに新しいテーブルを作成して、データを書き込む Lambda 関数を作成してテストします。

3. Amazon SQS キューを作成して、新しいメッセージが追加されるたびに Lambda 関数を呼び出すように設定します。
4. AWS マネジメントコンソール を使用してキューにメッセージを追加し、CloudWatch ログで結果をモニタリングすることによって、セットアップ全体をテストします。

これらの手順を実行することで、次のことが理解できます。

- Amazon RDS を使用してデータベースインスタンスとプロキシを作成し、Lambda 関数をプロキシに接続する方法。
- Lambda を使用して Amazon RDS データベースでの作成および読み取りオペレーションを実行する方法。
- Amazon SQS を使用して Lambda 関数を呼び出す方法。

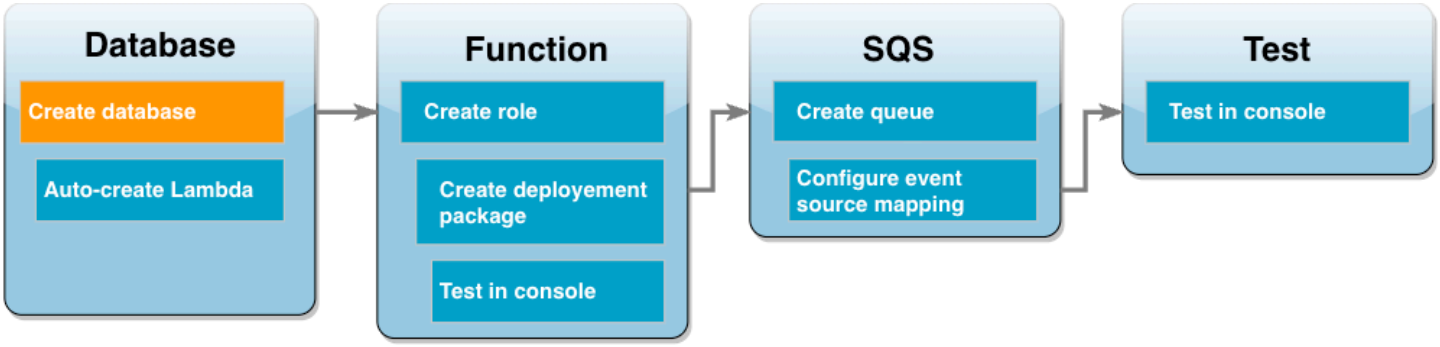
AWS マネジメントコンソール または AWS Command Line Interface(AWS CLI) を使って、このチュートリアルを完了できます。

前提条件

開始する前に、以下のセクションのステップを完了してください。

- AWS アカウント へのサインアップ

Amazon RDS DB インスタンスを作成する



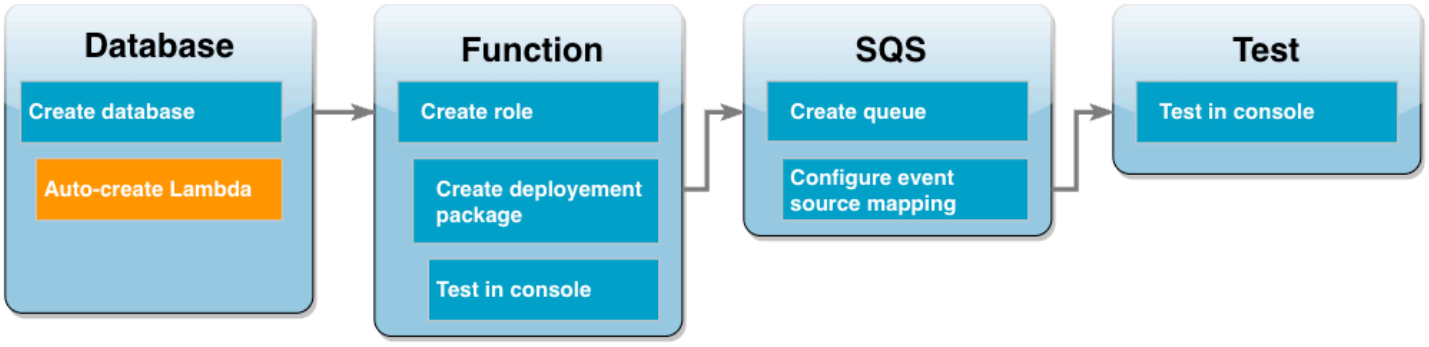
Amazon RDS DB インスタンスは、AWS クラウド で実行される分離されたデータベース環境です。インスタンスには、ユーザーが作成した 1 つ以上のデータベースを含めることができます。特に指定しない限り、Amazon RDS は、AWS アカウント に含まれるデフォルトの VPC に新しいデータベースインスタンスを作成します。Amazon VPC の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」を参照してください。

このチュートリアルでは、AWS アカウントのデフォルト VPC に新しいインスタンスを作成し、そのインスタンスに ExampleDB という名前のデータベースを作成します。AWS マネジメントコンソール または AWS CLI のいずれかを使用して、DB インスタンスとデータベースを作成できます。

データベースインスタンスを作成するには

1. Amazon RDS コンソールを開き、[データベースの作成] を選択します。
2. [標準作成] オプションを選択したままにし、[エンジンのオプション] で [MySQL] を選択します。
3. [テンプレート] で、[無料利用枠] または [サンドボックス] を選択します。[無料利用枠] は無料利用枠アカウントに表示されます。[サンドボックス] は有料プランアカウントに表示されます。
4. [設定] で、[DB インスタンス識別子] に **MySQLForLambda** を入力します。
5. 以下を実行して、ユーザー名とパスワードを設定します。
   a. [認証情報の設定] で、[マスターユーザー名] の設定を admin のままにします。
   b. [マスターパスワード] には、データベースにアクセスするためのパスワードを入力して確認します。
6. 次の操作を実行して、データベース名を指定します。
   - 残りのデフォルトオプションはすべて選択したままにして、[追加設定] セクションまで下にスクロールします。
   - このセクションを展開し、[最初のデータベース名] として **ExampleDB** を入力します。
7. 残りのデフォルトオプションはすべて選択したままにして、[データベースの作成] を選択します。

Lambda 関数とプロキシを作成する



RDS コンソールを使用して、データベースと同じ VPC に Lambda 関数とプロキシを作成できます。

> **Note**
>
> これらの関連リソースは、データベースの作成が完了し、[使用可能] ステータスのときにのみ作成できます。

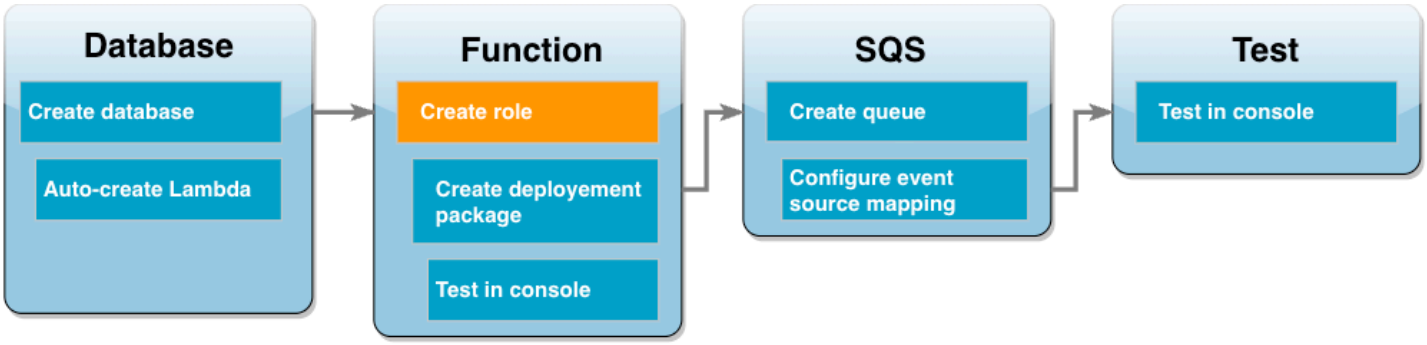
関連する関数とプロキシを作成するには

1. [データベース] ページから、データベースが [使用可能] ステータスかどうかを確認します。その場合は、次のステップに進みます。そうでない場合は、データベースが使用可能になるまで待ちます。
2. データベースを選択し、[アクション] から [Lambda 接続をセットアップする] を選択します。
3. [Lambda 接続をセットアップする] ページで、[新しい関数の作成] を選択します。

   [新しい Lambda 関数名] を **`LambdaFunctionWithRDS`** に設定します。
4. [RDS プロキシ] セクションで、[RDS プロキシを使用して接続] オプションを選択します。さらに、[新しいプロキシの作成] を選択します。
   - [データベース認証情報] として、[データベースのユーザー名とパスワード] を選択します。
   - [ユーザー名] として、`admin` と指定します。
   - [パスワード] として、データベースインスタンスのために作成したパスワードを入力します。
5. [セットアップ] を選択して、プロキシと Lambda 関数の作成を完了します。

ウィザードがセットアップを完了し、新しい関数を確認するための Lambda コンソールへのリンクが表示されます。Lambda コンソールに切り替える前に、プロキシエンドポイントをメモしておきます。

関数実行ロールを作成するには



Lambda 関数を作成する前に、関数に必要なアクセス権限を与える実行ロールを作成します。このチュートリアルでは、Lambda 二は、データベースインスタンスを含んでいる VPC へのネットワーク接続を管理し、Amazon SQS キューからメッセージをポーリングするためのアクセス許可が必要です。

Lambda 関数に必要なアクセス権限を付与するために、このチュートリアルでは IAM 管理ポリシーを使用します。これらは多くの一般的なユースケースに許可を付与するポリシーであり、AWS アカウントで利用できます。管理ポリシーの使用の詳細については、「ポリシーのベストプラクティス」を参照してください。

Lambda 実行ロールを作成するには

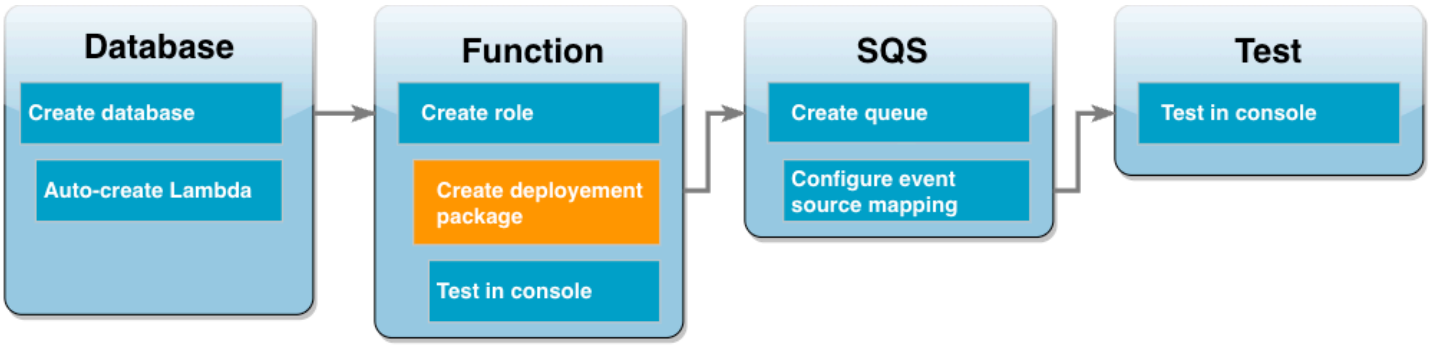
1. IAM コンソールの [ロール] ページを開いて、[ロールの作成] を選択します。
2. [信頼されるエンティティタイプ] として、[AWS サービス] を選択し、[ユースケース] として [Lambda] を選択します。
3. [次へ] を選択します。
4. 次の手順を実行して IAM 管理ポリシーを追加します。
    a. 検索ボックスで **AWSLambdaSQSQueueExecutionRole** を検索します。
    b. 結果リストで、ロールの横にあるチェックボックスを選択し、[フィルターをクリア] を選択します。
    c. 検索ボックスで **AWSLambdaVPCAccessExecutionRole** を検索します。
    d. 結果リストで、ロールの横にあるチェックボックスをオンにし、[次へ] を選択します。
5. [ロール名] には、**lambda-vpc-sqs-role** を入力して [ロールの作成] を選択します。

このチュートリアルの後半で、先ほど作成した実行ロールの Amazon リソースネーム (ARN) が必要になります。

実行ロール ARN を確認するには

1. IAM コンソールの [ロール] ページを開き、ロール (`lambda-vpc-sqs-role`) を選択します。
2. [概要] セクションに表示されている [ARN] をコピーします。

Lambda デプロイパッケージを作成する



次の Python コードの例では、PyMySQL パッケージを使用してデータベースへの接続を開始します。関数を初めて呼び出すと、`Customer` という新しいテーブルも作成されます。このテーブルは次のスキーマを使用しており、ここで `CustID` はプライマリキーです。

```
Customer(CustID, Name)
```

また、この関数は PyMySQL を使用してこのテーブルにレコードを追加します。この関数は、Amazon SQS キューに追加するメッセージで指定された顧客 ID と名前を使用してレコードを追加します。

このコードは、ハンドラー関数の外部でデータベースへの接続を作成します。初期化コードで接続を作成すると、その後の関数の呼び出しで接続を再利用できるようになり、パフォーマンスが向上します。本番アプリケーションでは、プロビジョニングされた同時実行性を使用して、要求された数のデータベース接続を初期化することもできます。関数が呼び出されると、これらの接続はすぐに利用できます。

```
import sys
import logging
import pymysql
import json
import os

# rds settings
user_name = os.environ['USER_NAME']
```

```
password = os.environ['PASSWORD']
rds_proxy_host = os.environ['RDS_PROXY_HOST']
db_name = os.environ['DB_NAME']

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# create the database connection outside of the handler to allow connections to be
# re-used by subsequent function invocations.
try:
        conn = pymysql.connect(host=rds_proxy_host, user=user_name, passwd=password,
 db=db_name, connect_timeout=5)
except pymysql.MySQLError as e:
    logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
    logger.error(e)
    sys.exit(1)

logger.info("SUCCESS: Connection to RDS for MySQL instance succeeded")

def lambda_handler(event, context):
    """
    This function creates a new RDS database table and writes records to it
    """
    message = event['Records'][0]['body']
    data = json.loads(message)
    CustID = data['CustID']
    Name = data['Name']

    item_count = 0
    sql_string = f"insert into Customer (CustID, Name) values(%s, %s)"

    with conn.cursor() as cur:
        cur.execute("create table if not exists Customer ( CustID  int NOT NULL, Name
 varchar(255) NOT NULL, PRIMARY KEY (CustID))")
        cur.execute(sql_string, (CustID, Name))
        conn.commit()
        cur.execute("select * from Customer")
        logger.info("The following items have been added to the database:")
        for row in cur:
            item_count += 1
            logger.info(row)
    conn.commit()

    return "Added %d items to RDS for MySQL table" %(item_count)
```

> **Note**
>
> この例では、データベースアクセス認証情報は環境変数として保存されます。本番アプリケーションでは、より安全なオプションとして AWS Secrets Manager を使用することをお勧めします。Lambda 関数が VPC にある場合、Secrets Manager に接続するには VPC エンドポイントを作成する必要があります。詳細については、「仮想プライベートクラウド内で Secrets Manager サービスに接続する方法」を参照してください。

PyMySQL の依存関係を関数コードに含めるには、.zip デプロイパッケージを作成します。次のコマンドは、Linux、macOS、または Unix で動作します。

.zip デプロイパッケージを作成するには

1. サンプルコードをファイル名 `lambda_function.py` で保存します。
2. `lambda_function.py` ファイルを作成したのと同じディレクトリに、`package` という名前の新しいディレクトリを作成し、PyMySQL ライブラリをインストールします。

   ```
   mkdir package
   pip install --target package pymysql
   ```

3. アプリケーションコードと PyMySQL ライブラリを含む zip ファイルを作成します。Linux または MacOS では、次の CLI コマンドを実行します。Windows では、任意の zip ツールを使用して、`lambda_function.zip` ファイルを作成します。`lambda_function.py` ソースコードファイルと依存関係を含むフォルダは、.zip ファイルのルートにインストールする必要があります。

   ```
   cd package
   zip -r ../lambda_function.zip .
   cd ..
   zip lambda_function.zip lambda_function.py
   ```

   また、Python 仮想環境を使用してデプロイパッケージを作成することもできます。「.zip ファイルアーカイブで Python Lambda 関数をデプロイする」を参照してください。

Lambda 関数を更新する

作成した.zip パッケージを使用して、Lambda コンソールで Lambda 関数を更新します。関数がデータベースにアクセスできるようにするには、アクセス認証情報を使用して環境変数を設定する必要もあります。

Lambda 関数を更新するには

1. Lambda コンソールの [関数] ページを開き、関数 `LambdaFunctionWithRDS` を選択します。
2. [ランタイム設定] タブで、[編集] を選択し、関数の [ランタイム] を [Python 3.10] に変更します。
3. [ハンドラー] を `lambda_function.lambda_handler` に変更します。
4. [コード] タブで、[アップロード元]、[.zip ファイル] の順に選択します。
5. 前の段階で作成した `lambda_function.zip` ファイルを選択してから、[保存] を選択します。

ここで、以前に作成した実行ロールで関数を設定します。これにより、データベースインスタンスにアクセスして Amazon SQS キューをポーリングするために必要なアクセス権限が関数に付与されます。

関数の実行ロールを設定するには

1. Lambda コンソールの [関数] ページで、[設定]、[アクセス権限] の順に選択します。
2. [実行ロール] で、[編集] を選択します。
3. [既存のロール] で、実行ロール (`lambda-vpc-sqs-role`) を選択します。
4. [保存] を選択します。

関数の環境変数を設定するには

1. Lambda コンソールの「関数」ページで、[Configuration] タブ、[Environment variables] の順に選択します。
2. [編集] を選択します。
3. データベースアクセス認証情報を追加するには、以下を実行してください。

    a. [Add environment variable] を選択し、[Key] には **`USER_NAME`**、[Value] には **`admin`** を入力します。

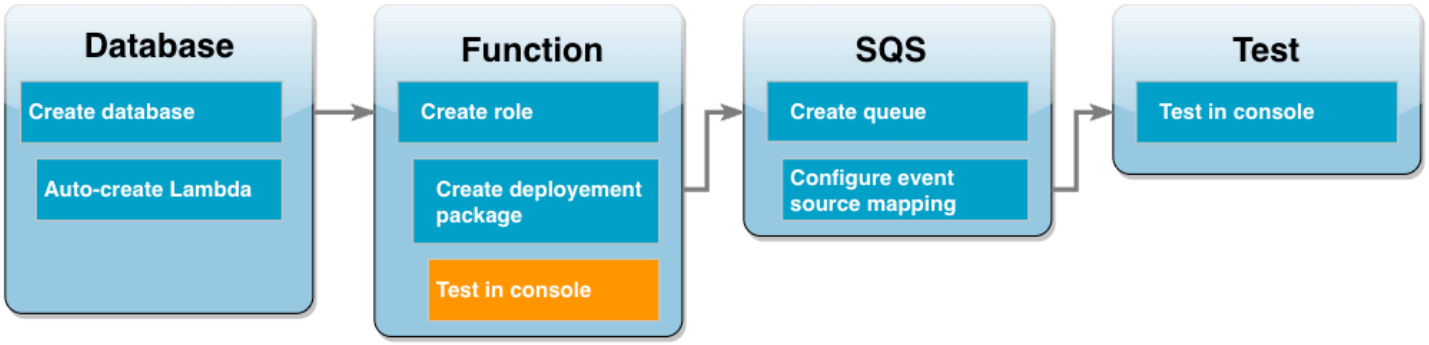
b. [Add environment variable] を選択し、[Key] には **DB_NAME**、[Value] には **ExampleDB** を入力します。

c. [環境変数の追加] を選択し、[キー] には **PASSWORD**、[値] にはデータベースの作成時に選択したパスワードを入力します。

d. [環境変数の追加] を選択し、[キー] として **RDS_PROXY_HOST** を入力し、[値] として、先程メモした RDS プロキシのエンドポイントを入力します。

e. [保存] を選択します。

コンソールで Lambda 関数をテストする



Lambda コンソールを使用して関数をテストできるようになりました。チュートリアルの最終段階で Amazon SQS を使用して関数を呼び出したときに、関数が受け取るデータを模倣するテストイベントを作成します。テストイベントには、関数が作成する `Customer` テーブルに追加する顧客 ID と顧客名を指定する JSON オブジェクトが含まれています。

Lambda 関数をテストする

1. Lambda コンソールの [関数] ページを開き、関数を選択します。
2. [テスト] セクションを選択します。
3. [新しいイベントを作成] を選択し、イベント名として **myTestEvent** と入力します。
4. 次のコードを [イベント JSON] にコピーし、[保存] を選択します。

```
{
  "Records": [
    {
      "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
      "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
      "body": "{\n     \"CustID\": 1021,\n     \"Name\": \"Martha Rivera\"\n}",
      "attributes": {
```

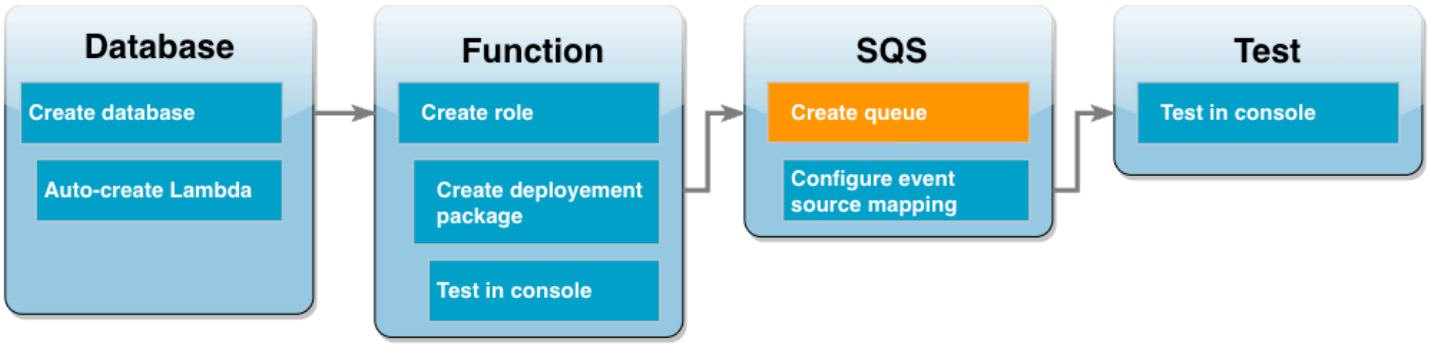
```
        "ApproximateReceiveCount": "1",
        "SentTimestamp": "1545082649183",
        "SenderId": "AIDAIENQZJOLO23YVJ4VO",
        "ApproximateFirstReceiveTimestamp": "1545082649185"
      },
      "messageAttributes": {},
      "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
      "eventSource": "aws:sqs",
      "eventSourceARN": "arn:aws:sqs:us-west-2:123456789012:my-queue",
      "awsRegion": "us-west-2"
    }
  ]
}
```

5. [テスト] を選択します。

[実行結果] タブに、次の [関数ログ] に表示されているのと同様の結果が表示されます。

```
[INFO] 2023-02-14T19:31:35.149Z bdd06682-00c7-4d6f-9abb-89f4bbb4a27f The following
 items have been added to the database:
[INFO] 2023-02-14T19:31:35.149Z bdd06682-00c7-4d6f-9abb-89f4bbb4a27f (1021, 'Martha
 Rivera')
```

Amazon SQS キュー を作成する



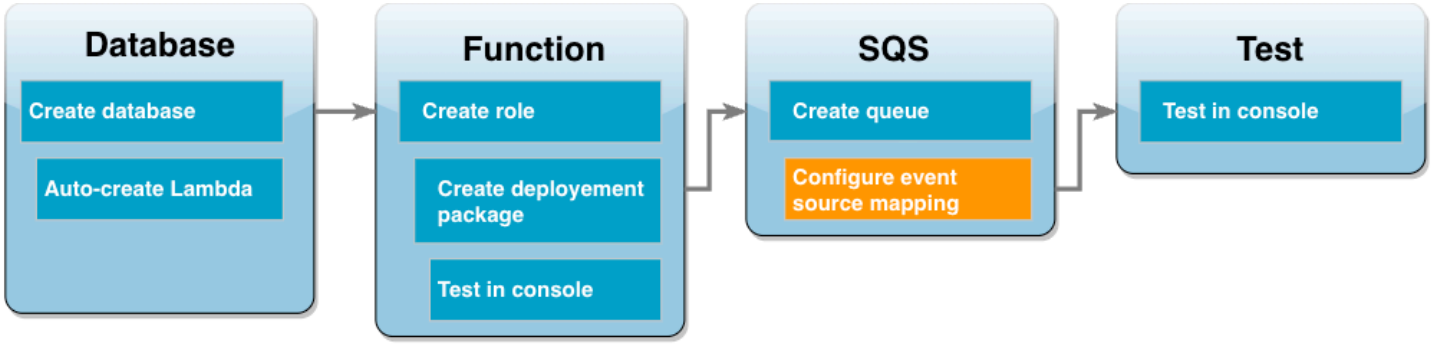
Lambda 関数と Amazon RDS データベースインスタンスの統合が正常にテストされました。ここで、チュートリアルの最終段階で Lambda 関数を呼び出すために使用する Amazon SQS キューを作成します。

Amazon SQS を作成するには (コンソール)

1. Amazon SQS コンソールの [キュー] ページを開き、[キューの作成] を選択します。

2. [タイプ] は [標準] のままにし、キューの名前に **LambdaRDSQueue** と入力します。
3. デフォルトオプションはすべて選択したままにして、[キューの作成] を選択します。

Lambda 関数を呼び出すためのイベントソースマッピングを作成する



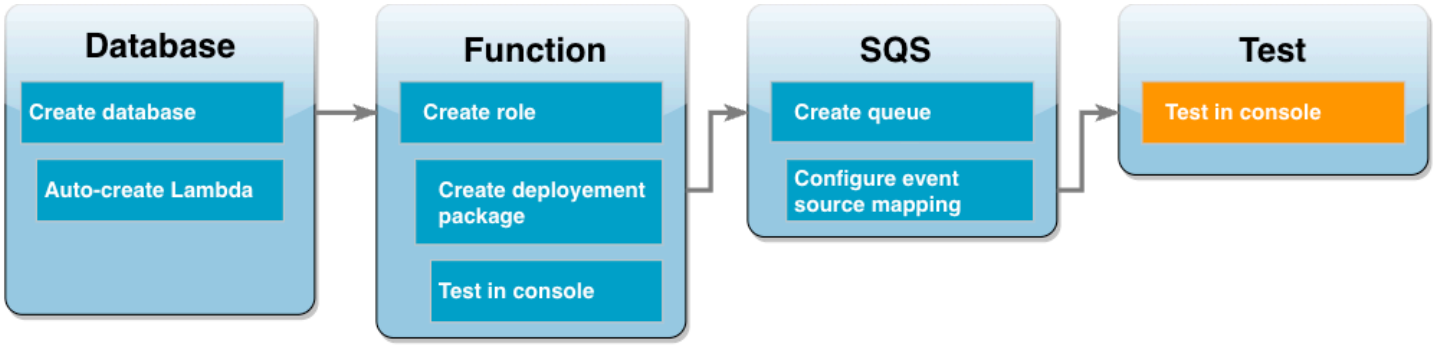
イベントソースマッピングは、ストリームまたはキューからアイテムを読み取り、Lambda 関数を呼び出す Lambda リソースです。イベントソースマッピングを設定するときに、ストリームまたはキューからのレコードが 1 つのペイロードにまとめられるようにバッチサイズを指定できます。この例では、キューにメッセージを送信するたびに Lambda 関数が呼び出されるように、バッチサイズを 1 に設定します。イベントソースマッピングは、AWS CLI または Lambda コンソールを使用して設定できます。

イベントソースマッピングを作成するには (コンソール)

1. Lambda コンソールの [関数] ページを開き、(`LambdaFunctionWithRDS`) 関数を選択します。
2. [関数の概要] セクションで、[トリガーを追加] を選択します。
3. ソースとして [Amazon SQS] を選択し、キューの名前 (`LambdaRDSQueue`) を選択します。
4. [バッチサイズ] に、**1** を入力します。
5. 他のオプションはすべてデフォルト値のままにして、[追加] を選択します。

これで Amazon SQS キューにメッセージを追加して、セットアップ全体をテストする準備ができました。

セットアップのテストとモニタリング



セットアップ全体をテストするには、コンソールを使用して Amazon SQS キューにメッセージを追加します。次に、CloudWatch Logs を使用して、Lambda 関数が予想どおりにレコードをデータベースに書き込んでいることを確認します。

セットアップをテストおよびモニタリングするには

1. Amazon SQS コンソールの [キュー] ページを開き、キュー (LambdaRDSQueue) を選択します。
2. [メッセージを送受信] を選択し、[メッセージの送信] セクションの [メッセージ本文] に次の JSON を貼り付けます。

   ```
   {
       "CustID": 1054,
       "Name": "Richard Roe"
   }
   ```

3. [メッセージの送信] を選択します。

   メッセージをキューに送信すると、Lambda はイベントソースマッピングを通じて関数を呼び出します。Lambda が予想どおりに関数を呼び出したことを確認するには、CloudWatch Logs を使用して、関数が顧客名と ID をデータベーステーブルに書き込んだことを確認します。

4. CloudWatch コンソールの [ロググループ] ページを開き、関数のロググループ (/aws/lambda/LambdaFunctionWithRDS) を選択します。
5. [ログストリーム] セクションで、最新のログストリームを選択します。

   テーブルには、関数の呼び出しごとに 1 つずつ、合計 2 つの顧客レコードが含まれている必要があります。ログストリームには、次に似たメッセージが表示されます。

```
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 The following
 items have been added to the database:
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 (1021, 'Martha
 Rivera')
[INFO] 2023-02-14T19:06:43.873Z 45368126-3eee-47f7-88ca-3086ae6d3a77 (1054,
 'Richard Roe')
```

リソースのクリーンアップ

このチュートリアル用に作成したリソースは、保持しない場合は削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

Lambda 関数を削除するには

1. Lambda コンソールの [関数] ページを開きます。
2. 作成した関数を選択します。
3. [Actions] で、[Delete] を選択します。
4. [削除] を選択します。

実行ロールを削除するには

1. IAM コンソールの [ロール] ページを開きます。
2. 作成した実行ロールを選択します。
3. [ロールの削除] を選択します。
4. [はい、削除します] を選択します。

MySQL DB インスタンスを削除するには

1. Amazon RDS コンソールの [Databases] (データベース) ページを開きます。
2. 作成したデータベースを選択します。
3. [Actions] で、[Delete] を選択します。
4. [Create final snapshot] (最終スナップショットの作成) のチェックボックスをオフにします。
5. テキストボックスに「**`delete me`**」と入力します。

6. [削除] を選択します。

Amazon SQS キューを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon SQS コンソール (https://console.aws.amazon.com/sqs/) を開きます。
2. 作成したキューを選択します。
3. [削除] を選択します。
4. テキストボックスに「**delete**」と入力します。
5. [Delete] (削除) をクリックします。

Amazon RDS チュートリアルおよびサンプルコード

AWS のドキュメントには、一般的な Amazon RDS のユースケースをガイドするチュートリアルがいくつか含まれています。これらのチュートリアルの多くは、他の AWS のサービスとともに Amazon RDS を使用する方法を説明しています。加えて、サンプルコードにアクセスすることもできます。

> **Note**
>
> その他のチュートリアルについては、AWS データベースブログをご覧ください。トレーニングの詳細については、AWS トレーニングと認定を参照してください。

トピック

- このガイドのチュートリアル
- 他の AWS ガイドのチュートリアル
- GitHub のチュートリアルとサンプルコード
- AWS データベースクックブック
- Amazon RDS PostgreSQL の AWS ワークショップおよびラボコンテンツポータル
- Amazon RDS MySQL の AWS ワークショップおよびラボコンテンツポータル
- このサービスを AWS SDK で使用する

このガイドのチュートリアル

このガイドの次のチュートリアルは、Amazon RDS を使用して一般的なタスクを実行する方法を示しています。

- チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)

  Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) に、DB インスタンスを含める方法について説明します。この場合、VPC は同じ VPC 内の Amazon EC2 インスタンスで実行しているウェブサーバーとデータを共有します。
- チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード)

Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) に、DB インスタンスを含める方法について説明します。この場合、VPC は同じ VPC 内の Amazon EC2 インスタンスとデータを共有します。このチュートリアルでは、デュアルスタックモードで実行されているデータベースで動作する VPC を、このシナリオで作成します。

- チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する

  PHP を使用する Apache ウェブサーバーのインストールと、MySQL データベースの作成を説明します。ウェブサーバーは Amazon Linux を使用して Amazon EC2 インスタンスで実行され、MySQL データベースは MySQL DB インスタンスと です。Amazon EC2 インスタンスと DB インスタンスの両方が Amazon VPC で実行されます。

- チュートリアル: DB スナップショットからの Amazon RDS DB インスタンスの復元

  DB スナップショットから DB インスタンスを復元する方法を説明します。

- チュートリアル: Lambda 関数を使用して Amazon RDS にアクセスする

  プロキシ経由でデータベースにアクセスし、テーブルを作成し、少数のレコードを追加し、テーブルからレコードを取得する Lambda 関数を RDS コンソールから作成する方法を説明します。また、Lambda 関数を呼び出してクエリ結果を確認する方法についても説明します。

- チュートリアル: タグを使用して、停止する DB インスタンスを指定する

  タグを使用して停止する DB インスタンスを指定する方法を説明します。

- チュートリアル: Amazon EventBridge を使用して DB インスタンスの状態変化を口グに記録する

  Amazon EventBridge および AWS Lambda を使用して DB インスタンスの状態変更を口グに記録する方法を説明します。

- チュートリアル: Amazon RDS 向けマルチ AZ DB クラスターレプリカラグ用の Amazon CloudWatch アラームの作成

  マルチ AZ DB クラスターのレプリカラグがしきい値を超えたときに Amazon SNS メッセージを送信する CloudWatch アラームを作成する方法について説明します。1 つのアラームで、指定した期間中、`ReplicaLag` メトリクスを監視します。アクションは、Amazon SNS トピックまたは Amazon EC2 Auto Scaling ポリシーに送信される通知です。

他の AWS ガイドのチュートリアル

他の AWS ガイドの次のチュートリアルは、Amazon RDS を使用して一般的なタスクを実行する方法を説明しています。

- AWS Secrets Manager ユーザーガイドの「チュートリアル: AWS データベース用のシークレットをローテーションする」

  AWS データベースのシークレットを作成し、スケジュールに従ってローテーションするよう設定します。1 つのローテーションを手動でトリガーし、新しいバージョンのシークレットが引き続きアクセスを提供していることを確認します。
- AWS Elastic Beanstalk デベロッパーガイドの「チュートリアルとサンプル」

  Amazon RDS データベースと AWS Elastic Beanstalk を使用するアプリケーションをデプロイする方法を説明します。
- Amazon Machine Learning デベロッパーガイド の Amazon RDS データベースのデータを使用して Amazon ML データソースを作成する

  MySQL DB インスタンスに格納されているデータから Amazon Machine Learning (Amazon ML) データソースオブジェクトを作成する方法を説明します。
- 「Amazon Quick ユーザーガイド」の「手動で VPC での Amazon RDS インスタンスへのアクセスを有効にする」

  Quick が VPC 内の Amazon RDS DB インスタンスにアクセスできるようにする方法を説明します。

GitHub のチュートリアルとサンプルコード

GitHub の次のチュートリアルとサンプルコードは、Amazon RDS を使用して一般的なタスクを実行する方法を示しています。

- Amazon Relational Database Service アイテムトラッカーの作成

  作業項目を追跡し、レポートするアプリケーションを作成する方法について説明します。このアプリケーションは、Amazon RDS、Amazon Simple Email Service、Elastic Beanstalk、SDK for Java 2.x を使用します。

AWS データベースクックブック

AWS DB クックブックは、AWS で高性能で費用対効果の高いデータベースソリューションを構築、デプロイ、管理する方法を学習する包括的なデータベースガイドです。ステップバイステップのチュートリアルでは、本番環境に対応したアプリケーションを作成し、CloudFormation テンプレートを使用してアプリケーションをデプロイする方法について説明します。インフラストラクチャの構築、ネットワーキングの実装、サーバーレスアーキテクチャの開発、データベースの管理、生成 AI の統合など、重要な AWS のサービスについて説明します。コストを最適化しながら、安全でスケーラブルなソリューションを作成するのに役立つ AWS のベストプラクティスについて説明します。AWS 初心者の方でも経験豊富なプロフェッショナルの方でも、AWS DB クックブックは、データベースに関する一般的な課題を解決し、エンタープライズ対応ソリューションを実装するためのスキルを習得するのに役立ちます。クックブックには以下のセクションがあります。

- DB アプリケーション向け AWS の開始方法 – アカウントや Jupyter Notebook 環境を設定する方法など、AWS の基本について説明します。
- データベースの基礎 – 重要なデータベースの概念を調べ、AWS のデータベースサービスを比較して、ワークロードに適したソリューションを選択します。
- Amazon Aurora を使用したサーバーレスウェブアプリケーション – インベントリ、注文、顧客データを処理する Amazon Aurora PostgreSQL を使用してエンドツーエンドの小売アプリケーションを構築します。
- モニタリングとオブザーバビリティ – パフォーマンス追跡を設定し、アラートを設定して、アプリケーションに影響を与える前に潜在的なデータベースの問題を特定します。
- Amazon Aurora によるスケーリング – Aurora DSQL を使用して回復力のあるマルチリージョンデプロイを構築する方法、データベースをスケールアップして処理能力を高める方法や、複数のインスタンスにまたがってスケールアウトして容量を増やす方法について説明します。
- 最適化のパフォーマンスとコスト – 実証済みの調整戦略により、データベースのパフォーマンスを最適化し、コストを削減します。
- AWS 目的別データベースへのジャーニー – 生成 AI ソリューションとデータ駆動型アプリケーションをプロトタイプからエンタープライズデプロイまでスケールする、安全で信頼性の高いインフラストラクチャを構築します。
- RAG を使用した GenAI アプリケーション – 検索拡張生成 (RAG) を使用して、コンテキストに応じた正確な結果を提供する保険および医療文書用のインテリジェント検索システムを構築します。

Amazon RDS PostgreSQL の AWS ワークショップおよびラボコンテンツポータル

以下のワークショップやその他のハンズオンコンテンツのコレクションは、Amazon RDS PostgreSQL の機能と能力を理解するのに役立ちます。

- DB インスタンスの作成

  DB インスタンスの作成方法について説明します。

- RDS ツールによるパフォーマンスモニタリング

  AWS と SQL ツール (Cloudwatch、拡張モニタリング、スロークエリログ、Performance Insights、PostgreSQL カタログビュー) を使用して、パフォーマンスの問題を理解し、データベースのパフォーマンスを向上させる方法を特定する方法について説明します。

Amazon RDS MySQL の AWS ワークショップおよびラボコンテンツポータル

以下のワークショップやその他のハンズオンコンテンツのコレクションは、Amazon RDS MySQL の機能と能力を理解するのに役立ちます。

- DB インスタンスの作成

  DB インスタンスの作成方法について説明します。

- Performance Insights の使用

  Performance Insights を使用して DB インスタンスを監視および調整する方法について説明します。

このサービスを AWS SDK で使用する

AWS Software Development Kit (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようになる API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コードの例
AWS CLI	AWS CLI コードの例
AWS SDK for Go	AWS SDK for Go コードの例
AWS SDK for Java	AWS SDK for Java コードの例
AWS SDK for JavaScript	AWS SDK for JavaScript コードの例
AWS SDK for Kotlin	AWS SDK for Kotlin コードの例
AWS SDK for .NET	AWS SDK for .NET コードの例
AWS SDK for PHP	AWS SDK for PHP コードの例
AWS Tools for PowerShell	AWS Tools for PowerShell コードの例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コードの例
AWS SDK for Ruby	AWS SDK for Ruby コードの例
AWS SDK for Rust	AWS SDK for Rust コードの例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コードの例
AWS SDK for Swift	AWS SDK for Swift コードの例

このサービスに固有の例については、「AWS SDK を使用した Amazon RDS のコード例」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの下側にある [Provide feedback] リンクから、コードの例をリクエストしてください。

Amazon RDS のベストプラクティス

Amazon RDS を使用するためのベストプラクティスについて説明します。新しいベストプラクティスが確認されると、このセクションは更新されます。

トピック

- Amazon RDS の基本的な操作のガイドライン
- DB インスタンスの RAM の推奨事項
- データベースエンジンのバージョンを最新の状態に保つ
- AWS データベースドライバー
- 拡張モニタリングを使用したオペレーティングシステムの問題の特定
- メトリクスを使用したパフォーマンスの問題の特定
- クエリのチューニング
- MySQL の使用のベストプラクティス
- MariaDB の使用のベストプラクティス
- Oracle を使用するためのベストプラクティス
- PostgreSQL を使用するためのベストプラクティス
- SQL Server を使用するためのベストプラクティス
- DB パラメータグループを使用する
- DB インスタンスの作成を自動化するためのベストプラクティス
- Amazon RDS の新機能の動画

> **Note**
>
> Amazon RDS の一般的な推奨事項については、Amazon RDS の推奨事項 を参照してください。

Amazon RDS の基本的な操作のガイドライン

以下に示しているのは、基本的な運用についてのガイドラインであり、Amazon RDS の使用時にすべてのユーザーが従う必要があります。Amazon RDS のサービスレベルアグリーメントでは、次のガイドラインに従う必要があります。

- メトリクスを使用して、メモリ、CPU、レプリカの遅延、およびストレージの使用状況をモニタリングします。Amazon CloudWatch は、使用パターンが変更されたり、デプロイメントの最大容量に近づいたりすると、通知するように設定できます。このようにして、システムのパフォーマンスと可用性を維持できます。
- 最大ストレージ容量に近づいたら、DB インスタンスをスケールアップする。アプリケーションからのリクエストの予期しない増加に対応するために、ストレージとメモリにいくらかのバッファがあることが必要です。
- 自動バックアップを有効にし、1 日のうちで書き込み IOPS が低くなる時間帯にバックアップが実行されるようにバックアップウィンドウを設定する。この場合、バックアップによるデータベース使用量は最も少なくなります。
- データベースのワークロードでプロビジョニングした I/O がより多く必要になると、フェイルオーバーやデータベース障害後の復旧が遅くなります。DB インスタンスの I/O 処理能力を高めるには、以下のいずれかまたはすべての操作を実行します。
  - I/O 処理能力の高い別の DB インスタンスクラスに移行します。
  - 必要とする処理能力の向上に応じて、汎用 SSD からプロビジョンド IOPS SSD ストレージに変換します。利用可能なストレージの種類の詳細については、「Amazon RDS ストレージタイプ」を参照してください。

    プロビジョンド IOPS ストレージに変換する場合には、プロビジョンド IOPS に合わせて最適化された DB インスタンスクラスも使用してください。プロビジョンド IOPS の詳細については、「プロビジョンド IOPS SSD ストレージ」を参照してください。
  - プロビジョンド IOPS ストレージをすでに使用している場合は、追加の処理能力をプロビジョニングする。
- クライアントアプリケーションが DB インスタンスのドメインネームサービス (DNS) のデータをキャッシュしている場合には、有効期限 (TTL) の値を 30 秒未満に設定します。DB インスタンスの基になる IP アドレスは、フェイルオーバー後に変更されている可能性があります。そのため、DNS データを長時間キャッシュすると、接続障害が発生する可能性があります。アプリケーションが、使用されなくなった IP アドレスに接続しようとする可能性があります。
- DB インスタンスのフェイルオーバーをテストすることで、そのプロセスで特定のユースケースにかかる時間を把握します。また、フェイルオーバーが発生した後、DB インスタンスにアクセスするアプリケーションが自動的に新しい DB インスタンスに接続できることを確認するために、フェイルオーバーをテストします。

DB インスタンスの RAM の推奨事項

Amazon RDS のパフォーマンスのベストプラクティスは、作業セットをほぼ完全にメモリ内に保持できるように十分な RAM を割り当てることです。作業セットは、インスタンスで頻繁に使用されるデータとインデックスです。DB インスタンスを使用するほど、作業セットが増大します。

作業セットがほぼすべてメモリ内にあるかどうかを調べるには、DB インスタンスに負荷がかかっている状態で、(Amazon CloudWatchを使用して) ReadIOPS メトリクスを確認します。ReadIOPS の値は小さく、安定している必要があります。場合によっては、DB インスタンスクラスを、RAM の容量が大きいクラスにスケールアップして、ReadIOPS が劇的に低下することがあります。このような場合、ワーキングセットはほぼ完全にメモリ内にあったわけではありません。スケーリングオペレーションを実行しても ReadIOPS が急激に低下しなくなるか、ReadIOPS が非常に小さい値になるまで、スケールアップを継続します。DB インスタンスのメトリクスのモニタリングについては、「Amazon RDS コンソールでのメトリクスの表示」を参照してください。

データベースエンジンのバージョンを最新の状態に保つ

セキュリティ、パフォーマンス、コンプライアンスを維持するために、データベースエンジンのバージョンを定期的にアップグレードします。Amazon RDS は、セキュリティパッチ、パフォーマンスの強化、新機能を含む新しいマイナーバージョンとメジャーバージョンをリリースします。古いデータベースエンジンを実行すると、ワークロードが既知の脆弱性、互換性の問題にさらされ、AWS およびデータベースベンダーからのサポートが低下する可能性があります。

中断を最小限に抑えるには、アップグレードを計画するときに次の点を考慮してください。

- ステージング環境でテストする – 本番データベースをアップグレードする前に、ワークロードに対して新しいバージョンを検証します。
- Amazon RDS マネージドアップグレードの使用 – マイナーバージョンの自動アップグレードを有効にすると、パッチ適用が容易になります。
- メジャーバージョンアップグレードのスケジュール – リリースノートを確認し、アプリケーションの互換性をテストし、制御されたアップグレードウィンドウを計画します。

定期的なアップグレードにより、データベースの安全性、最適化、AWS ベストプラクティスとの整合性が維持されます。

AWS データベースドライバー

アプリケーション接続には、AWS のドライバースイートを推奨します。ドライバーは、スイッチオーバーとフェイルオーバーの時間の短縮のほか、AWS Secrets Manager、AWS Identity and Access Management (IAM)、フェデレーティッド ID での認証をサポートするように設計されています。AWS ドライバーは、DB インスタンスステータスをモニタリングし、インスタントポロジを認識して新しいライターを決定することを前提としています。このアプローチにより、スイッチオーバーとフェイルオーバーの時間が 1 桁秒に短縮されます (オープンソースドライバーの場合は数十秒)。

新しいサービス機能が導入されるにあたって、こうしたサービス機能を標準でサポートすることが AWS のドライバースイートの目標です。

詳細については、「AWS ドライバーを使用した DB インスタンスへの接続」を参照してください。

拡張モニタリングを使用したオペレーティングシステムの問題の特定

拡張モニタリングが有効になっている場合、Amazon RDS は、DB インスタンスが実行されるオペレーティングシステム (OS) のメトリクスをリアルタイムで提供します。コンソールを使用して DB インスタンスのメトリクスを表示できます。また、選択したモニタリングシステムで Amazon CloudWatch Logs からの拡張モニタリング JSON 出力を使用できます。拡張モニタリングの詳細については、拡張モニタリングを使用した OS メトリクスのモニタリング を参照してください。

メトリクスを使用したパフォーマンスの問題の特定

リソース不足やその他の一般的なボトルネックによるパフォーマンスの問題を特定するには、Amazon RDS DB インスタンスに適用されるメトリクスを監視できます。

パフォーマンスメトリクスの表示

さまざまな時間範囲の平均値、最大値、最小値を表示するには、パフォーマンスメトリクスを定期的に監視する必要があります。そのように監視している場合、いつパフォーマンスが低下しているかを特定できます。また、特定のメトリクスしきい値に対して Amazon CloudWatch アラームを設定することにより、しきい値に達した場合に警告されるようにすることもできます。

パフォーマンスの問題を解決するために重要なのは、システムのベースラインパフォーマンスを理解することです。DB インスタンスをセットアップして一般的なワークロードで実行する場合は、すべ

てのパフォーマンスメトリクスの平均値、最大値、最小値をキャプチャします。さまざまな間隔 (例えば、1 時間、24 時間、1 週間、2 週間) で行います。これにより、正常な状態を把握することができます。それにより、オペレーションのピークおよびオフピークの時間帯を比較して、得られた情報から、いつパフォーマンスが標準レベルを下回っているかを特定できます。

マルチ AZ の DB クラスターを使用している場合、ライター DB インスタンスの最新のトランザクションと、リーダー DB インスタンスで最後に適用されたトランザクションとの時間の差をモニタリングできます。この差は、レプリカ遅延と呼ばれます。詳細については、「レプリカの遅延とマルチ AZ DB クラスター」を参照してください。

Performance Insights と CloudWatch メトリクスの組み合わせをPerformance Insights ダッシュボードで表示し、DB インスタンスをモニタリングできます。このモニタリングビューを使用するには、DB インスタンスの Performance Insights がオンになっている必要があります。このモニタリングビューの詳細については、「Performance Insights ダッシュボードを使用して組み合わせたメトリクスを表示する」を参照してください。

特定の期間のパフォーマンス分析レポートを作成して、特定されたインサイトと問題を解決するための推奨事項を確認できます。詳細については、「Performance Insights でのパフォーマンス分析レポートの作成」を参照してください。

パフォーマンスメトリクスを表示するには

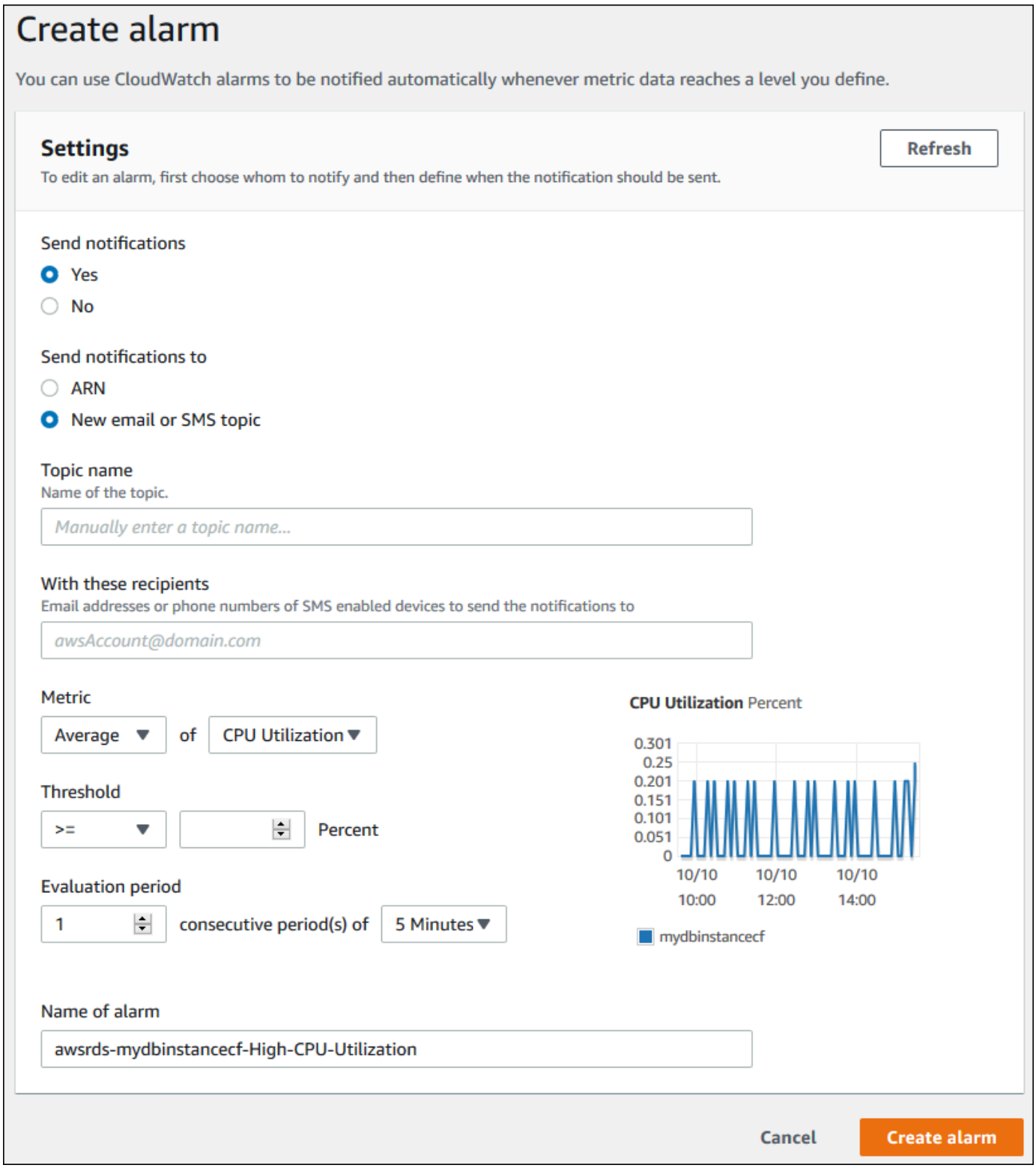
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [データベース] を選択し、DB インスタンスを選択します。
3. [モニタリング] を選択します。

   ダッシュボードにはパフォーマンスメトリクスが表示されます。メトリクスはデフォルトで、過去 3 時間の情報が表示されます。
4. ページ分割ボタンを使用して別のメトリクスを表示するか、設定を調整してさらにメトリクスを表示します。
5. 現在の日付以外のデータを表示するには、パフォーマンスメトリクスを選択して時間範囲を調整します。[Statistic]、[Time Range]、[Period] の値を変更して、表示される情報を調整できます。例えば、過去 2 週間の各日についてメトリクスの最大値を表示したい場合があります。その場合は、[Statistic] (統計) を [Maximum] (最大) に、[Time Range] (時間範囲) を [Last 2 Weeks] (過去 2 週間)、[Period] (期間) を [Day] (日) に設定します。

CLI または API を使用しても、パフォーマンスメトリクスを表示できます。詳細については、「Amazon RDS コンソールでのメトリクスの表示」を参照してください。

CloudWatch アラームを設定するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [データベース] を選択し、DB インスタンスを選択します。
3. [Logs & events] を選択します。
4. [CloudWatch アラーム] セクションで [アラームの作成] を選択します。



5. [通知の送信] で、[はい] を選択し、[通知の送信先]で、[New email or SMS topic] を選択します。
6. [Topic name (トピック名)] に通知用のトピックの名前を入力し、[With these recipients (対象の受取人)] に E メールアドレスまたは電話番号をコンマで区切って入力します。

7. [Metric (メトリクス)] で、セットするアラーム統計とメトリクスを選択します。
8. [Threshold (しきい値)] で、メトリクスがしきい値より大きい、しきい値より小さい、またはしきい値に等しいかどうかを指定し、しきい値の数値を指定します。
9. [Evaluation period] (期間) で、アラームの評価期間を選択します。[consecutive period(s) of] (度次の間隔で発生) で、アラートをトリガーするためにしきい値に達するまでの期間を選択します。
10. [アラーム名] に、アラームの名前を入力します。
11. [Create Alarm] を選択します。

[CloudWatch alarms (CloudWatch アラーム)] セクションで アラームが表示されます。

パフォーマンスメトリクスの評価

DB インスタンスには多くのカテゴリのメトリクスがあり、許容値の決め方はメトリクスによって異なります。

CPU

- CPU Utilization – 使用されているコンピュータの処理能力の割合。

メモリ

- Freeable Memory – DB インスタンスで使用可能な RAM の量 (バイト単位)。[モニタリング] タブのメトリクスの CPU、メモリ、ストレージメトリクスの 75% 地点に赤い線でマークされています。インスタンスメモリの消費量が頻繁にこの線を超える場合は、ワークロードの見直し、またはインスタンスのアップグレードが必要であることを表します。
- Swap Usage – DB インスタンスによって使用されているスワップスペースの量 (バイト単位)。

ディスク容量

- Free Storage Space – DB インスタンスによって現在使用されていないディスク領域の量 (メガバイト単位)。

入力/出力オペレーション

- Read IOPS、Write IOPS – 1 秒あたりのディスク読み取りまたは書き込みオペレーションの平均数。

- Read Latency、Write Latency – 読み取りまたは書き込みオペレーションの平均時間 (ミリ秒単位)。
- Read Throughput、Write Throughput – 1 秒あたりのディスク読み取りまたは書き込みデータの平均量 (メガバイト単位)。
- Queue Depth – ディスクに対する読み取りまたは書き込み待機中の I/O オペレーションの数。

ネットワークトラフィック

- Network Receive Throughput、Network Transmit Throughput – 1 秒あたりの DB インスタンスに対する送信または受信ネットワークトラフィックのレート (バイト単位)。

データベース接続

- DB Connections – DB インスタンスに接続されたクライアントセッションの数。

使用可能な各パフォーマンスメトリクスの詳細については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

一般的に、パフォーマンスメトリクスの許容値は、ベースラインがどのようになっているか、アプリケーションによって何が実行されているかによって異なります。ベースラインからの一貫した差異またはトレンドになっている差異を調べます。メトリクスのタイプごとのアドバイスは以下のとおりです。

- CPU または RAM の高消費量 — CPU または RAM の消費量が大きい値になっていても、それは妥当である場合があります。例えば、アプリケーションの目標 (スループット、同時実行数など) に沿った想定値であることが前提です。
- ディスクスペースの消費量 – 使用されているディスクスペースが一貫して合計ディスクスペースの 85% 以上である場合は、ディスクスペースの消費量を調べます。インスタンスからデータを削除するか、別のシステムにデータをアーカイブして、スペースを解放できるかどうかを確認します。
- ネットワークトラフィック - ネットワークトラフィックについてシステム管理者に問い合わせて、ドメインネットワークとインターネット接続に対する想定スループットを把握します。スループットが一貫して想定よりも低い場合は、ネットワークトラフィックを調べます。
- データベース接続数 – ユーザー接続数が多いことが、インスタンスのパフォーマンスが下がっていること、応答時間が長くなっていることに関連しているとわかった場合、データベース接続数を制限することを検討します。DB インスタンスの最適なユーザー接続数は、インスタンスのク

ラスと実行中のオペレーションの複雑さによって異なります。データベース接続数を確認するには、DB インスタンスをパラメータグループに関連付けます。このグループでは、ユーザー接続パラメータを 0 (無制限) 以外に設定します。既存のパラメータグループを使用するか、新しいパラメータグループを作成できます。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

- IOPS メトリクス - IOPS メトリクスの想定値はディスクの仕様とサーバーの設定によって異なるため、ベースラインを使用して一般的な値を把握します。値とベースラインとの差が一貫しているかどうかを調べます。最適な IOPS パフォーマンスを得るには、読み取りおよび書き込みオペレーションが最小限になるように、一般的な作業セットがメモリに収まることを確認してください。

パフォーマンスメトリクスの問題については、パフォーマンスを向上させるための第一歩は、最も使用頻度が高く、最も tune コストのかかるクエリをチューニングすることです。チューニングして、システムリソースへの負荷が軽減されるかどうかを確認してください。詳細については、「クエリのチューニング」を参照してください。

クエリが調整されていても問題が解決しない場合は、Amazon RDS DB インスタンスクラス のアップグレードを検討してください。問題に関連するリソース (CPU、RAM、ディスク容量、ネットワーク帯域幅、I/O 容量) が多いものにアップグレードすることができます。

クエリのチューニング

DB インスタンスのパフォーマンスを向上させるには、大量のリソースを消費する使用頻度の最も高いクエリをチューニングします。ここでは、それらを調整して実行コストを下げます。クエリの改善については、次のリソースを参照してください。

- MySQL – MySQL ドキュメントの Optimizing SELECT statements を参照してください。クエリチューニングリソースの詳細については、MySQL performance tuning and optimization resources を参照してください。
- Oracle – Oracle Database ドキュメントの Database SQL Tuning Guide を参照してください。
- SQL Server – Microsoft のドキュメントの Analyzing a query を参照してください。Microsoft ドキュメントの System Dynamic Management Views に説明されているように、実行、インデックス、I/O 関連のデータ管理ビュー (DMV) を使用して、SQL Server クエリの問題をトラブルシューティングすることもできます。

  クエリのチューニングの一般的な側面は、効果的なインデックスの作成です。DB インスタンスの潜在的なインデックスの改善については、Microsoft ドキュメントの Database Engine Tuning Advisor を参照してください。RDS for SQL Server での Tuning Advisor の使用について

は、Database Engine Tuning Advisor を使用して Amazon RDS for SQL Server DB インスタンスのデータベースワークロードを分析する を参照してください。

- PostgreSQL – クエリプランの分析方法については、PostgreSQL ドキュメントの Using EXPLAIN を参照してください。この情報を使用してクエリまたは基礎となるテーブルを変更することで、クエリのパフォーマンスを向上させることができます。

  最適なパフォーマンスを得るためにクエリで結合を指定する方法については、Controlling the planner with explicit JOIN clauses を参照してください。
- MariaDB – MariaDB ドキュメントの Query optimizations を参照してください。

MySQL の使用のベストプラクティス

MySQL データベースのテーブルサイズとテーブル数の両方が、パフォーマンスに影響を与える可能性があります。

テーブルのサイズ

通常、ファイルサイズに対するオペレーティングシステムの制約によって、MySQL データベースの有効な最大テーブルサイズが決まります。したがって、制限は通常、内部の MySQL 制約によって決定されません。

MySQL DB インスタンスでは、データベース内のテーブルが過度に大きくならないようにしてください。一般的なストレージの制限は 64 TiB ですが、プロビジョンドストレージでは、MySQL テーブルファイルの最大サイズは 16 TB に制限されています。ファイルサイズが 16 TB の制限を十分に下回るように、大きなテーブルはパーティション化します。この方法により、パフォーマンスと復旧時間も向上します。詳細については、「Amazon RDS での MySQL のファイルサイズ制限」を参照してください。

非常に大きなテーブル (100 GB を超える) は、読み取りと書き込み (DML ステートメント、特に DDL ステートメントを含む) の両方のパフォーマンスに悪影響を与える可能性があります。ラージテーブルのインデックスは、選択パフォーマンスを大幅に向上させることができますが、DML ステートメントのパフォーマンスを低下させる可能性があります。ALTER TABLE などの DDL ステートメントは、ラージテーブルでは大幅に遅くなる可能性があります。これは、これらの操作によってテーブルが完全に再構築される場合があるためです。これらの DDL ステートメントは、操作の間、テーブルをロックする場合があります。

MySQL で読み取りと書き込みに必要なメモリの量は、操作に関係するテーブルによって異なります。アクティブに使用されているテーブルのインデックスを保持するのに十分な RAM を少なくとも

確保しておくことがベストプラクティスです。データベース内の 10 個の最も大きいテーブルとインデックスを検索するには、次のクエリを使用します。

```
select table_schema, TABLE_NAME, dat, idx from
(SELECT table_schema, TABLE_NAME,
       ( data_length ) / 1024 / 1024 as dat,
       ( index_length ) / 1024 / 1024 as idx
FROM information_schema.TABLES
order by 3 desc ) a
order by 3 desc
limit 10;
```

テーブルの数

基になるファイルシステムでは、テーブルを表すファイル数が制限されている場合があります。ただし、MySQL にはテーブルの数に制限はありません。これにもかかわらず、MySQL InnoDB ストレージエンジンのテーブルの合計数は、これらのテーブルのサイズに関係なく、パフォーマンスの低下につながる可能性があります。オペレーティングシステムの影響を制限するために、同じ MySQL DB インスタンス内の複数のデータベース間でテーブルを分割できます。そうすると、ディレクトリ内のファイル数が制限される可能性がありますが、全体的な問題は解決されません。

多数のテーブル (1 万以上) のためにパフォーマンスの低下がある場合、それは MySQL がストレージファイルのオープンとクローズを含む作業を行うことによって引き起こされるものです。この問題に対処するには、table_open_cache および table_definition_cache パラメータのサイズを大きくします。ただし、これらのパラメータの値を大きくすると、MySQL が使用するメモリの量が大幅に増加し、使用可能なメモリがすべて使用される場合もあります。詳細については、MySQL ドキュメントの「MySQL でのテーブルのオープンとクローズの方法」を参照してください。

さらに、テーブルが多すぎると、MySQL のスタートアップ時間に大きな影響を与える可能性があります。特に MySQL 8.0 より前のバージョンでは、クリーンシャットダウンと再起動およびクラッシュ復旧の両方が影響を受ける可能性があります。

DB インスタンス内のすべてのデータベースで、テーブル数合計を 1 万未満にすることをお勧めします。MySQL データベース内で多数のテーブルを使用するユースケースについては、「MySQL 8.0 の 100 万テーブル」を参照してください。

ストレージエンジン

Amazon RDS for MySQL のポイントインタイム復元とスナップショット復元の機能を利用するには、クラッシュからの回復が可能なストレージエンジンが必要です。これらの機能は、InnoDB スト

レージエンジンでのみサポートされます。MySQL は様々な機能を持つ複数のストレージエンジンをサポートしていますが、それらすべてがクラッシュの回復とデータ耐久性に最適化されているわけではありません。例えば、MyISAM ストレージエンジンで信頼性の高いクラッシュ回復機能がサポートされていないと、ポイントインタイム復元機能やスナップショット復元機能が意図したとおりに動作しない場合があります。その場合、MySQL がクラッシュ後に再起動されるときに、データの損失やクラッシュにつながることがあります。

InnoDB は、Amazon RDS での MySQL DB インスタンスについて推奨およびサポートされているストレージエンジンです。InnoDB のインスタンスは、Aurora に移行することも可能です。MyISAM のインスタンスは移行できません。ただし、強力な全文検索機能が必要な場合、MyISAM の方が InnoDB よりもパフォーマンスは高くなります。Amazon RDS で MyISAM を使用する場合は、「サポートされない MySQL ストレージエンジンを使用した自動バックアップ」で説明されている手順に従ってください。スナップショット復元機能の特定のシナリオに役立つことがあります。

既存の MyISAM テーブルを InnoDB テーブルに変換する場合は、MySQL ドキュメントの「Converting Tables from MyISAM to InnoDB」で説明されているプロセスを使用できます。MyISAM と InnoDB にはそれぞれ長所と短所があるため、アプリケーションに対してこの切り替えを行う前に、影響の評価を十分に行ってください。

加えて、フェデレーティッドストレージエンジンは、現在 Amazon RDS for MySQL ではサポートされていません。

MariaDB の使用のベストプラクティス

MariaDB データベースのテーブルサイズとテーブル数の両方がパフォーマンスに影響を与える可能性があります。

テーブルのサイズ

通常、ファイルサイズに対するオペレーティングシステムの制約によって、MariaDB データベースの有効な最大テーブルサイズが決まります。したがって、制限は通常、内部 MariaDB 制約によって決定されません。

MariaDB DB インスタンスでは、データベース内のテーブルが過度に大きくならないようにしてください。一般的なストレージの制限は 64 TiB ですが、プロビジョニング済みストレージでは、MariaDB テーブルファイルの最大サイズは 16 TiB に制限されています。ファイルサイズが 16 TB の制限を十分に下回るように、大きなテーブルはパーティション化します。この方法により、パフォーマンスと復旧時間も向上します。

非常に大きなテーブル (100 GB を超える) は、読み取りと書き込み (DML ステートメント、特に DDL ステートメントを含む) の両方のパフォーマンスに悪影響を与える可能性があります。ラージテーブルのインデックスは、選択パフォーマンスを大幅に向上させることができますが、DML ステートメントのパフォーマンスを低下させる可能性があります。ALTER TABLE などの DDL ステートメントは、ラージテーブルでは大幅に遅くなる可能性があります。これは、これらの操作によってテーブルが完全に再構築される場合があるためです。これらの DDL ステートメントは、操作の間、テーブルをロックする場合があります。

MariaDB が読み取りと書き込みに必要なメモリの量は、操作に関係するテーブルによって異なります。アクティブに使用されているテーブルのインデックスを保持するのに十分な RAM を少なくとも確保しておくことがベストプラクティスです。データベース内の 10 個の最も大きいテーブルとインデックスを検索するには、次のクエリを使用します。

```
select table_schema, TABLE_NAME, dat, idx from
(SELECT table_schema, TABLE_NAME,
       ( data_length ) / 1024 / 1024 as dat,
       ( index_length ) / 1024 / 1024 as idx
FROM information_schema.TABLES
order by 3 desc ) a
order by 3 desc
limit 10;
```

テーブルの数

基になるファイルシステムでは、テーブルを表すファイル数が制限されている場合があります。ただし、MariaDB にはテーブルの数に制限はありません。これにもかかわらず、MariaDB InnoDB ストレージエンジンのテーブルの合計数は、これらのテーブルのサイズに関係なく、パフォーマンスの低下に寄与する可能性があります。オペレーティングシステムの影響を制限するために、同じ MariaDB DB インスタンス内の複数のデータベース間でテーブルを分割できます。そうすると、ディレクトリ内のファイル数が制限される可能性がありますが、全体的な問題は解決されません。

多数のテーブル (1 万以上) のためにパフォーマンスの低下がある場合、それは MariaDB が作業を行うことによって引き起こされるものです。この作業には、MariaDB のストレージファイルのオープンとクローズが含まれます。この問題に対処するには、`table_open_cache` および`table_definition_cache` パラメータのサイズを大きくします。ただし、これらのパラメータの値を大きくすると、MariaDB が使用するメモリの量が大幅に増加する場合もあります。使用可能なメモリをすべて消費することもあります。詳細については、MariaDB ドキュメントの「table_open_cache の最適化」を参照してください。

さらに、テーブルが多すぎると、MariaDB のスタートアップ時間に大きな影響を与える可能性があります。クリーンシャットダウンと再起動およびクラッシュ復旧の両方が影響を受ける可能性があります。DB インスタンス内のすべてのデータベースで、テーブル数合計を 1 万未満にすることをお勧めします。

ストレージエンジン

Amazon RDS for MariaDB のポイントインタイム復元とスナップショット復元の機能を利用するには、クラッシュからの回復が可能なストレージエンジンが必要です。MariaDB は様々な機能を持つ複数のストレージエンジンをサポートしていますが、それらすべてがクラッシュの回復とデータ耐久性に最適化されているわけではありません。例えば、Aria は耐クラッシュ性を備えていますが MyISAM の代わりに使用した場合、ポイントインタイム復元やスナップショット復元は意図したとおりに動作しないことがあります。その場合、MariaDB がクラッシュ後に再起動されるときに、データの損失やクラッシュにつながることがあります。InnoDB は、Amazon RDS での MariaDB DB インスタンスについて推奨およびサポートされているストレージエンジンです。Amazon RDS で Aria を使用する場合は、「サポートされない MariaDB ストレージエンジンを使用した自動バックアップ」で説明されている手順に従ってください。スナップショット復元機能の特定のシナリオに役立つことがあります。

既存の MyISAM テーブルを InnoDB テーブルに変換する場合は、MariaDB ドキュメントの「Converting Tables from MyISAM to InnoDB」で説明されているプロセスを使用できます。MyISAM と InnoDB にはそれぞれ長所と短所があるため、アプリケーションに対してこの切り替えを行う前に、影響の評価を十分に行ってください。

Oracle を使用するためのベストプラクティス

Amazon RDS for Oracle を使用するためのベストプラクティスについては、Amazon Web Services で Oracle データベースを実行するためのベストプラクティスを参照してください。

2020 年の AWS 仮想ワークショップにおいて、本稼働用 Oracle データベースの、Amazon RDS 上での実行に関するプレゼンテーションが取り上げられています。プレゼンテーションビデオは、こちらにあります。

PostgreSQL を使用するためのベストプラクティス

RDS for PostgreSQL のパフォーマンスを改善できる 2 つの重要な領域のうち、1 つは、DB インスタンスにデータをロードするときです。もう 1 つは、PostgreSQL autovacuum 機能を使用するときです。以下のセクションでは、これらの領域で推奨する方法のいくつかを説明します。

Amazon RDS がその他の一般的な PostgreSQL DBA タスクを実装する方法については、Amazon RDS for PostgreSQL の一般的な DBA タスク を参照してください。

PostgreSQL DB インスタンスにデータをロードする

Amazon RDS PostgreSQL DB インスタンスにデータをロードする場合は、DB インスタンスの設定や DB パラメータグループの値を変更してください。これらを設定すると、DB インスタンスにデータを最も効率的にインポートできます。

DB インスタンスの設定を次のように変更します。

- DB インスタンスのバックアップを無効にする (backup_retention を 0 に設定する)
- マルチ AZ を無効にする

次の設定を含むように DB パラメータグループを変更します。また、DB インスタンスの最も効率的な設定を見つけるために、パラメータ設定をテストしてください。

- `maintenance_work_mem` パラメータの値を大きくします。PostgreSQL のリソース消費のパラメータの詳細については、PostgreSQL のドキュメントを参照してください。
- ログ先行書き込み (WAL) ログへの書き込みの数を減らすには、`max_wal_size` パラメータと `checkpoint_timeout` パラメータの値を大きくします。
- `synchronous_commit` パラメータを無効にします。
- PostgreSQL の autovacuum パラメータを無効にします。
- インポートするいずれのテーブルもログの記録漏れがないことを確認します。ログの記録漏れのテーブルに保存されているデータはフェイルオーバー時に失われる可能性があります。詳細については、CREATE TABLE UNLOGGED を参照してください。

これらの設定で、`pg_dump -Fc` (圧縮) または `pg_restore -j` (並列) コマンドを使用します。

ロード操作が完了したら、DB インスタンスと DB パラメータを通常の設定に戻します。

PostgreSQL Autovacuum 機能の使用

PostgreSQL データベースの autovacuum 機能は、PostgreSQL DB インスタンスの状態を維持するために使用することを強くお勧めする機能です。autovacuum は、VACUUM および ANALYZE コマンドの実行を自動化します。autovacuum の使用は、PostgreSQL が必要とするもので、Amazon RDS では必須ではありませんが、良い性能を出すためには重要な要素です。この機能は、すべての

新しい Amazon RDS for PostgreSQL DB インスタンスで、デフォルトで有効になり、関連する設定パラメータがデフォルトで適切に設定されます。

データベース管理者は、このメンテナンスオペレーションを認識し、理解している必要があります。自動バキュームに関する PostgreSQL のドキュメントについては、The Autovacuum Daemon を参照してください。

自動バキュームは「リソースを使用しない」オペレーションではありませんが、バックグラウンドで動作し、可能な限りユーザーオペレーションを優先します。有効にした場合、autovacuum は、大量のタプルが更新または削除されたテーブルを確認します。また、トランザクション ID の循環のために非常に古いデータが失われることを防止します。詳細については、「Preventing Transaction ID Wraparound Failures」を参照してください。

autovacuum は、パフォーマンス向上のために削減できるオーバーヘッドの大きいオペレーションと考えられるものではありません。反対に、autovacuum が実行されていない場合、更新や削除が高速なテーブルのパフォーマンスが徐々に悪化します。

> **Important**
>
> autovacuum を実行しない場合、影響がさらに大きいバキュームオペレーションを実行するために、最終的には機能停止が必要になる可能性があります。場合によっては、autovacuum を過度に控えめに使用することで、RDS for PostgreSQL DB インスタンスが利用できなくなる場合があります。このような場合、PostgreSQL データベースはそれ自体を保護するためにシャットダウンします。この時点で、Amazon RDS は直接 DB インスタンスに対してシングルユーザーモードの完全バキュームを実行する必要があります。このように完全バキュームを行うと、数時間に及ぶ停止が発生する可能性があります。 したがって、デフォルトで有効になっている、autovacuum を無効にしないことを強くお勧めします。

autovacuum パラメータは、いつ、どのように autovacuum を実行するかを決定します。`autovacuum_vacuum_threshold` および `autovacuum_vacuum_scale_factor` パラメータは、autovacuum がいつ実行されるかを決定します。`autovacuum_max_workers`、`autovacuum_nap_time`、`autovacuum_cost_limit`、`autovacuum` の各パラメータは、どのように autovacuum を実行するかを決定します。自動バキューム、その実行のタイミング、および必要なパラメータの詳細については、PostgreSQL のドキュメントの Routine Vacuuming を参照してください。

次のクエリは、table1 というテーブルの「dead」タプルの数を示します。

```
SELECT relname, n_dead_tup, last_vacuum, last_autovacuum FROM
pg_catalog.pg_stat_all_tables
WHERE n_dead_tup > 0 and relname = 'table1';
```

このクエリの結果は次のようになります。

```
relname | n_dead_tup | last_vacuum | last_autovacuum
---------+------------+-------------+-----------------
 tasks   |   81430522 |             |
(1 row)
```

Amazon RDS for PostgreSQL のベストプラクティスの動画

2020 AWS re:Invent カンファレンスでは、Amazon RDS で PostgreSQL を使用する上での、新機能とベストプラクティスに関するプレゼンテーションが行われました。プレゼンテーションビデオは、こちらにあります。

SQL Server を使用するためのベストプラクティス

SQL Server DB インスタンスでのマルチ AZ 配置のベストプラクティスには、次のようなものがあります。

- フェイルオーバーをモニタリングするために Amazon RDS DB イベントを使用します。例えば、DB インスタンスがフェイルオーバーしたときに、テキストメッセージまたはメールで通知できます。Amazon RDS イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。
- アプリケーションが DNS 値をキャッシュする場合は、有効期限 (TTL) を 30 秒未満に設定します。TTL をこのように設定することは、フェイルオーバーが発生した場合に備えるための適切な方法です。フェイルオーバーでは、IP アドレスが変更される場合があり、キャッシュされた値がサービスで使用されなくなる場合があります。
- マルチ AZ に必要なトランザクションのログ作成を無効にするため、以下モードを有効にしないことをお勧めします。
  - シンプル復旧モード
  - オフラインモード
  - 読み取り専用モード
- DB インスタンスのフェイルオーバーにどのくらいの時間がかかるかを調べるためにテストします。フェイルオーバーの時間は、使用していたデータベース、インスタンスクラス、およびスト

レージタイプによって異なります。フェイルオーバーが発生した場合に、アプリケーションが機能し続けるかどうかもテストする必要があります。

- フェイルオーバー時間を短縮するには、次の操作を行います。
  - ワークロードのために割り当てられた十分なプロビジョンド IOPS があることを確認します。不十分な I/O によってフェイルオーバー時間が長くなる可能性があります。データベースの復旧には I/O が必要です。
  - より小さいトランザクションを使用します。データベース復旧はトランザクションに依存するため、大きいトランザクションを複数の小さいトランザクションに分割できる場合、フェイルオーバー時間は短くなります。
- フェイルオーバー中に、レイテンシーが高くなることを考慮します。フェイルオーバープロセスの一環として、Amazon RDS は新しいスタンバイ用のインスタンスに自動的にデータをレプリケートします。このレプリケーションは、新しいデータが 2 つの異なる DB インスタンスにコミットされていることを意味します。そのため、スタンバイ DB インスタンスが新しいプライマリ DB インスタンスに追いつくまでにレイテンシーが発生する場合があります。
- アプリケーションをすべてのアベイラビリティーゾーンにデプロイします。1 つのアベイラビリティーゾーンが機能を停止しても、他のアベイラビリティーゾーンのアプリケーションは利用できます。

SQL Server のマルチ AZ 配置を使用すると、Amazon RDS はインスタンスのすべての SQL Server データベースのレプリカを作成します。特定のデータベースのセカンダリレプリカを作成しない場合は、これらのデータベースでマルチ AZ を使用しないように別の DB インスタンスをセットアップします。

Amazon RDS for SQL Server のベストプラクティスの動画

2019 AWS re:Invent カンファレンスでは、Amazon RDS で SQL Server を使用する上での、新機能とベストプラクティスに関するプレゼンテーションが行われました。プレゼンテーションビデオは、こちらにあります。

DB パラメータグループを使用する

DB パラメータグループを変更した場合は、その変更をテスト DB インスタンスで試してから本稼働 DB インスタンスに適用することをお勧めします。DB パラメータグループに不適切な設定の DB エンジンパラメータがあると、パフォーマンスの低下やシステムの不安定化など、予期しない悪影響が生じることがあります。DB エンジンパラメータの変更時には常に注意が必要です。DB パラメータグループを変更する前に必ず DB インスタンスをバックアップしてください。

DB インスタンスのバックアップの詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

DB インスタンスの作成を自動化するためのベストプラクティス

Amazon RDS のベストプラクティスは、データベースエンジンの優先マイナーバージョンを使用して DB インスタンスを作成することです。AWS CLI、Amazon RDS API、または AWS CloudFormation を使用して DB インスタンスの作成を自動化できます。これらの方法を使用すると、メジャーバージョンのみを指定でき、Amazon RDS は優先マイナーバージョンを使用してインスタンスを自動的に作成します。例えば、PostgreSQL 12.5 が優先マイナーバージョンであり、`create-db-instance` でバージョン 12 を指定した場合、DB インスタンスのバージョンは 12.5 になります。

優先マイナーバージョンを確認するには、次の例に示すように、`describe-db-engine-versions` オプションを指定して `--default-only` コマンドを実行します。

```
aws rds describe-db-engine-versions --default-only --engine postgres

{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "EngineVersion": "12.5",
            "DBParameterGroupFamily": "postgres12",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 12.5-R1",
            ...some output truncated...
        }
    ]
}
```

DB インスタンスをプログラムで作成する方法については、次のリソースを参照してください。

- AWS CLI の使用 – create-db-instance
- Amazon RDS API – CreateDBInstance の使用
- AWS CloudFormation の使用 – AWS::RDS::DBInstance

Amazon RDS の新機能の動画

2023 AWS re:Invent カンファレンスでは、Amazon RDS の新機能に関するプレゼンテーションがありました。プレゼンテーションビデオは、こちらにあります。

Amazon RDS へのプログラムによるアクセス

Amazon RDS には、Amazon RDS リソースをプログラムで管理するための以下のツールが用意されています。

AWS Command Line Interface (AWS CLI)

RDS リソースを作成および管理するには、コマンドラインシェルの AWS CLI を使用します。AWS CLI は、Amazon RDS などの AWS のサービスの API に直接アクセスできます。Amazon RDS のコマンドの構文と例については、「AWS CLI コマンドリファレンス」の「rds」を参照してください。

AWS CloudFormation

この AWS Infrastructure as Code (IaC) ツールを使用すると、必要なすべての Amazon RDS リソースを記述するテンプレートを作成し、AWS CloudFormation によってそれらのリソースをプロビジョニングして設定できます。詳細については、「the section called “ での リソースの作成 AWS CloudFormation”」を参照してください。

AWS Software Development Kits (SDK)

AWS では、多くの一般的なテクノロジーとプログラミング言語用の SDK を提供しています。これにより、その言語またはテクノロジーでアプリケーション内から AWS のサービスを簡単に呼び出すことができます。これらの SDK の詳細については、「AWS でアプリケーションを開発および管理するためのツール」を参照してください。

Amazon RDS API

この API は Amazon RDS のプロトコルレベルのインターフェイスです。この API を使用する場合は、すべての HTTPS リクエストを正しくフォーマットし、すべてのリクエストに有効なデジタル署名を追加する必要があります。詳細については、「Amazon RDS API リファレンス」を参照してください。

Console-to-Code

このツールを使用すると、Amazon RDS コンソールで実行するアクションのコードを生成し、そのコードを AWS CloudFormation などの他のツールで使用できます。詳細については、「the section called “Console-to-Code”」を参照してください。

Console-to-Code を使用して Amazon RDS コンソールアクションのコードを生成します。

コンソールには、リソースの作成とプロトタイプのテストを行うためのガイド付きのパスが用意されています。同じリソースを大規模に作成したい場合は、自動化コードが必要です。Console-to-Code は Amazon Q Developer の機能であり、オートメーションコードの使用を開始するために役立ちます。Console-to-Code は、指定したデフォルト値やパラメータ値を含むコンソールのアクションを記録します。次に、生成 AI を使用して、選択したアクションに任意の言語と形式のコードを提案します。コンソールワークフローは指定されたパラメータ値の併用が有効であることを確実にするため、Console-to-Code を使用して生成されたコードには、互換性のあるパラメータ値があります。このコードを出発点として使用し、特定のユースケースの本番環境に対応するようにカスタマイズできます。

例えば、Console-to-Code では、RDS DB インスタンスの作成を記録し、AWS CloudFormation JSON 形式でコードを生成することを選択できます。その後、そのコードをコピーして、AWS CloudFormation テンプレートでの使用向けにカスタマイズできます。

現在、Console-to-Code は次の言語と形式で Infrastructure as Code (IaC) を生成できます。

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

Console-to-Code の使用方法に関する詳細と手順については、「Amazon Q Developer User Guide」の「Automating AWS services with Amazon Q Developer Console-to-Code」を参照してください。

Amazon RDS DB インスタンスの設定

このセクションでは、Amazon RDS DB インスタンスの設定方法を示します。DB インスタンスを作成する前に、DB インスタンスを実行する DB インスタンスクラスを決定します。また、AWS リージョンを選択して DB インスタンスを実行する場所を決定します。次に、DB インスタンスを作成します。

オプショングループと DB パラメータグループを使用して DB インスタンスを設定できます。

- オプショングループには、特定のAmazon RDS DB インスタンスに使用できるオプションという機能を指定できます。
- DB パラメータグループは、1 つ以上の DB インスタンスに適用されるエンジン設定値のコンテナとして機能します。

使用可能なオプションとパラメータは、DB エンジンと DB エンジンのバージョンによって異なります。DB インスタンスの作成時にオプショングループと DB パラメータグループを指定することができます。DB インスタンスを変更して指定することもできます。

トピック

- Amazon RDS DB インスタンスの作成
- AWS CloudFormation での Amazon RDS リソースの作成
- Amazon RDS DB インスタンスへの接続
- オプショングループを使用する
- Amazon RDS のパラメータグループ
- の Amazon RDS DB インスタンス設定を使用した Amazon ElastiCache キャッシュの作成
- AWS Database Migration Service を使用した、データベースの Amazon RDS への自動移行
- チュートリアル: カスタムパラメータと新しいオプショングループを使用した MySQL DB インスタンス の作成

Amazon RDS DB インスタンスの作成

Amazon RDS の基本構成要素は、データベースの作成先の DB インスタンスです。DB インスタンスの作成時に、エンジン固有の特性を選択します。また、データベースサーバーが実行されている AWS インスタンスのストレージ容量、CPU、メモリなどを選択します。

トピック

- DB インスタンスの前提条件
- DB インスタンスの作成
- DB インスタンスの設定

DB インスタンスの前提条件

> **Important**
>
> Amazon RDS DB インスタンスを作成したり、DB インスタンスに接続したりする前に、「Amazon RDS 環境のセットアップ」のタスクを完了します。

RDS DB インスタンスを作成するための前提条件を次に示します。

トピック

- DB インスタンスのネットワークを設定する
- 追加の前提条件

DB インスタンスのネットワークを設定する

Amazon RDS DB インスタンスは、Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) でのみ作成できます。また、少なくとも 2 つのアベイラビリティーゾーンを持つ AWS リージョン である必要があります。DB インスタンスに選択する DB サブネットグループは、少なくとも 2 つのアベイラビリティーゾーンを対象とする必要があります。この設定により、DB インスタンスを作成する時に マルチ AZ の配置を設定したり、将来的に簡単に移行することができます。

新しい DB インスタンスと同じ VPC 内の Amazon EC2 インスタンス間の接続を設定するには、DB インスタンスの作成中に設定します。同じ VPC 内の EC2 インスタンス以外のリソースから DB インスタンスに接続するには、ネットワーク接続を手動で設定できます。

トピック

- EC2 インスタンスとの自動ネットワーク接続を設定する
- ネットワークを手動で設定する

EC2 インスタンスとの自動ネットワーク接続を設定する

RDS DB インスタンスを作成する場合は、AWS マネジメントコンソール を使用して Amazon EC2 インスタンスと新しい DB インスタンス間の接続をセットアップできます。これを行うと、RDS では VPC とネットワークの設定を自動で行います。EC2 インスタンスが DB インスタンスにアクセスできるように、EC2 インスタンスと同じ VPC 内に DB インスタンスを作成します。

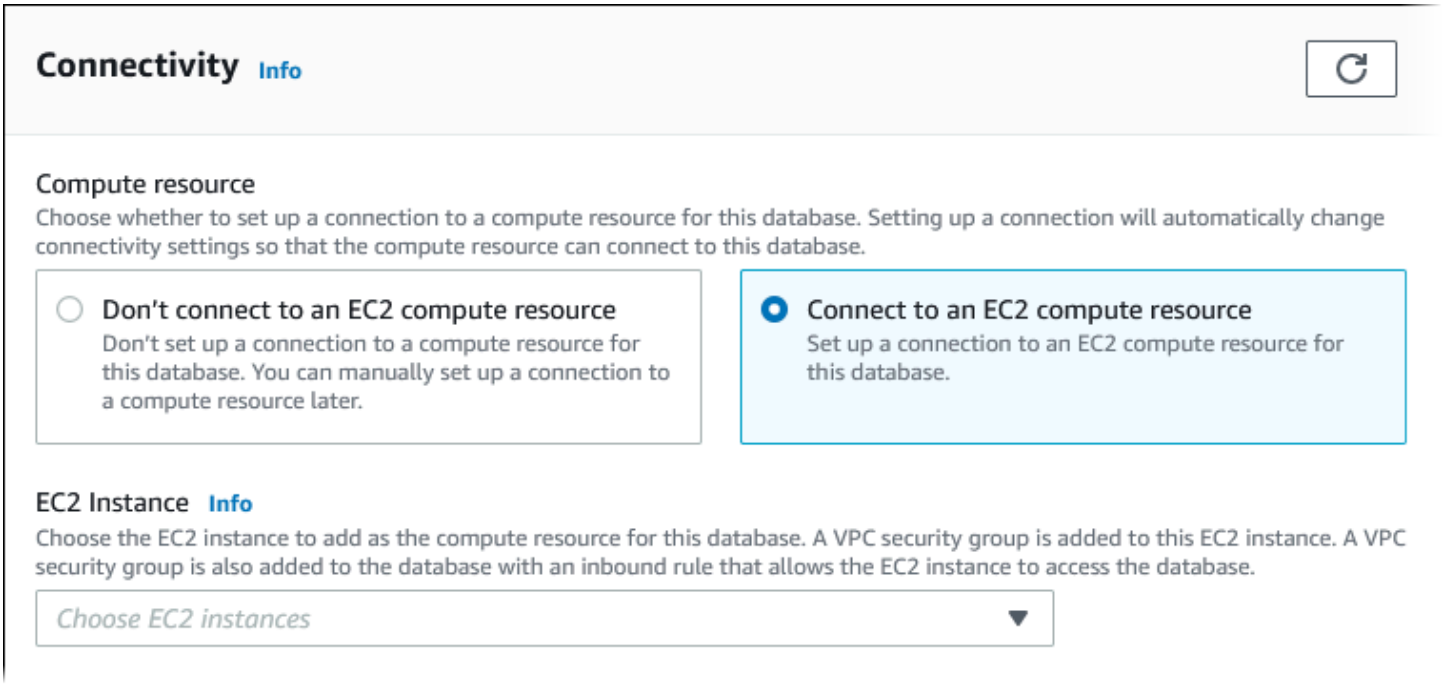
EC2 インスタンスと DB インスタンスを接続するための要件は次のとおりです。

- DB インスタンスを作成する前に、AWS リージョン に EC2 インスタンスが存在する必要があります。

  AWS リージョン に EC2 インスタンスが存在しない場合、コンソールには EC2 インスタンス作成用のリンクが表示されます。
- DB インスタンスを作成するユーザーには、次の操作を実行する権限が必要です。
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

このオプションを使用すると、プライベート DB インスタンスが作成されます。DB インスタンスでは、プライベートサブネットのみを持つ DB サブネットグループを使用して、VPC 内のリソースへのアクセスを制限します。

EC2 インスタンスを DB インスタンスに接続するには、[Create database] (データベースの作成) ページの [Connectivity] (接続) セクションで、[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択します。



[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択すると、RDS では次のオプションを自動的に設定します。[Don't connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続しない) を選択して EC2 インスタンスとの接続をセットアップしない限り、これらの設定は変更できません。

コンソールオプション	自動ログ記録
ネットワークの種類	RDS はネットワークタイプを IPv4 に設定します。EC2 インスタンスと DB インスタンス間の接続をセットアップする場合、デュアルスタックモードは現在サポートされていません。
仮想プライベートクラウド (VPC)。	RDS は EC2 インスタンスに関連付けられている VPC に設定します。

コンソールオプション	自動ログ記録
DB サブネットグループ	RDS では、EC2 インスタンスと同じアベイラビリティーゾーンにプライベートサブネットを持つ DB サブネットグループが必要です。この要件を満たす DB サブネットグループが存在する場合、RDS は既存の DB サブネットグループを使用します。デフォルトでは、このオプションは [Automatic setup] (自動セットアップ) に設定されています。 [Automatic setup] (自動セットアップ) を選択したとき、この要件を満たす DB サブネットグループがない場合、次のアクションが実行されます。RDS は 3 つのアベイラビリティーゾーンで 3 つの使用可能なプライベートサブネットを使用します。アベイラビリティーゾーンのうちの 1 つは EC2 インスタンスと同じです。プライベートサブネットがアベイラビリティーゾーンで使用できない場合、RDS はアベイラビリティーゾーンにプライベートサブネットを作成します。次に、RDS は DB サブネットグループを作成します。 プライベートサブネットが使用可能な場合、RDS はサブネットに関連付けられているルートテーブルを使用して、作成したサブネットをこのルートテーブルに追加します。プライベートサブネットが使用できない場合、RDS はインターネットゲートウェイにアクセスできないルートテーブルを作成し、作成したサブネットをルートテーブルに追加します。 RDS では、既存の DB サブネットグループを使用することもできます。既存の DB サブネットグループを使用する場合は、[Choose existing] (既存を選択) を選択します。
パブリックアクセス	RDS では [No] (いいえ) を選択して、DB インスタンスがパブリックアクセス可能にならないようにします。 セキュリティ上の理由から、データベースは非公開にし、インターネットからアクセスできないようにするのがベストプラクティスです。

コンソールオプション	自動ログ記録
VPC セキュリティグループ (ファイアウォール)	RDS では DB インスタンスに関連付けられている新しいセキュリティグループを作成します。セキュリティグループの名前は `rds-ec2-`*n* で、*n* は数字です。このセキュリティグループには、EC2 VPC セキュリティグループ (ファイアウォール) をソースとするインバウンドルールが含まれています。DB インスタンスに関連付けられているこのセキュリティグループにより、EC2 インスタンスが DB インスタンスにアクセスできるようになります。 また、RDS では EC2 インスタンス関連付けられている新しいセキュリティグループを作成します。セキュリティグループの名前は `ec2-rds-`*n* で、*n* は数字です。このセキュリティグループには、DB インスタンスの VPC セキュリティグループをソースとするアウトバウンドルールが含まれています。このセキュリティグループにより、EC2 インスタンスは DB インスタンスにトラフィックを送信できます。 [Create new] (新規作成) を選択して、新しいセキュリティグループの名前を入力すると、別のセキュリティグループを新規に追加できます。 既存のセキュリティグループを追加するには、[Choose existing] (既存を選択) を選択し、追加するセキュリティグループを選択します。

コンソールオプション	自動ログ記録
アベイラビリティーゾーン	[Availability & durability] (可用性と耐久性) (シングル AZ 配置) で、[Single DB instance] (シングル DB インスタンス) を選択した場合、RDS は EC2 インスタンスのアベイラビリティーゾーンを選択します。 [Availability & durability] (可用性と耐久性) (マルチ AZ DB インスタンス配置) で、[Multi-AZ DB instance] (マルチ AZ DB インスタンス) を選択した場合、RDS は、デプロイの 1 つの DB インスタンスに対して、EC2 インスタンスのアベイラビリティーゾーンを選択します。RDS は他の DB インスタンスに対し、異なるアベイラビリティーゾーンをランダムに選択します。プライマリ DB インスタンスまたはスタンバイレプリカのいずれかが、EC2 インスタンスと同じアベイラビリティーゾーンに作成されます。[Multi-AZ DB instance] (マルチ AZ DB インスタンス) を選択する場合、DB インスタンスと EC2 インスタンスが異なるアベイラビリティーゾーンにある場合は、アベイラビリティーゾーン間のコストが発生する可能性があります。

これらの設定の詳細については、「DB インスタンスの設定」をご参照ください。

DB インスタンスの作成後にこれらの設定を変更すると、EC2 インスタンスと DB インスタンス間の接続に影響する可能性があります。

ネットワークを手動で設定する

同じ VPC 内の EC2 インスタンス以外のリソースから DB インスタンスに接続するには、ネットワーク接続を手動で設定できます。AWS マネジメントコンソール を使用して DB インスタンスを作成する場合は、お客様に代わって Amazon RDS に VPC を自動的に作成させることができます。または、既存の VPC を使用するか、 DB インスタンス用に新しい VPC を作成することができます。どの方法を使用する場合でも、VPC を RDS DB インスタンスで使用するには、少なくとも 2 つのアベイラビリティーゾーンのそれぞれに 1 つ以上のサブネットが必要です。

デフォルトでは、Amazon RDS は DB インスタンスとアベイラビリティーゾーンを自動的に作成します。特定のアベイラビリティーゾーンを選択するには、[Availability & durability] (可用性と耐久性) 設定を [Single DB instance] (単一の DB インスタンス) に変更します。これにより、アベイラビリ

ティーゾーンの設定が表示され、VPC 内のアベイラビリティーゾーンの中から選択できます。ただし、マルチ AZ 配置を選択した場合、RDS はプライマリまたはライター DB インスタンスのアベイラビリティーゾーンを自動的に選択し、アベイラビリティーゾーン設定は表示されません。

場合によっては、デフォルト VPC を持っていない、または VPC を作成していない場合もあります。このような場合は、コンソールを使用して DB インスタンスを作成するときに、Amazon RDS に VPC を自動的に作成させることができます。それ以外の場合は以下の作業を行います。

- DB インスタンスをデプロイする AWS リージョン で、少なくとも 2 つのアベイラビリティーゾーンのそれぞれに 1 つ以上のサブネットを持つ VPC を作成します。詳細については、「VPC 内の DB インスタンスの使用」および「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。
- DB インスタンスへの接続を許可する VPC セキュリティグループを指定します。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」および「セキュリティグループによるアクセス制御」を参照してください。
- DB インスタンスが使用できる VPC 内の最低 2 つのサブネットを定義する RDS DB サブネットグループを指定します。詳細については、「DB サブネットグループの使用」を参照してください。

DB インスタンスと同じ VPC にないリソースに接続する場合は、「VPC の DB インスタンスにアクセスするシナリオ」の該当するシナリオを参照してください。

追加の前提条件

DB インスタンスを作成する前に、以下の追加の前提条件を考慮してください。

- AWS Identity and Access Management (IAM) 認証情報を使用して AWS に接続している場合は、AWS アカウントに特定の IAM ポリシーが必要です。これにより、Amazon RDS オペレーションを実行するために必要なアクセス権限が付与されます。詳細については、「Amazon RDS での Identity and Access Management」を参照してください。

  IAM を使用して RDS コンソールにアクセスする場合は、IAM ユーザーの認証情報を使用して AWS マネジメントコンソール にサインインします。次に、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) に移動します。
- DB インスタンスの設定パラメータを調整するには、必要なパラメータ設定を持つ DB パラメータグループを指定します。DB パラメータグループの作成と変更の詳細については、「Amazon RDS のパラメータグループ」を参照してください。

> **Important**
>
> Amazon RDS for Db2 の BYOL モデルを使用している場合は、DB インスタンスを作成する前に、まず、IBM Site ID と IBM Customer ID を含むカスタムパラメータグループを作成する必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL)」を参照してください。

- DB インスタンス用に指定する TCP/IP ポート番号を確認します。会社のファイアウォールによっては、RDS DB インスタンスのデフォルトポートへの接続がブロックされます。会社のファイアウォールがデフォルトのポートをブロックする場合は、お客様の DB インスタンス用に別のポートを選択します。Amazon RDS DB エンジンのデフォルトポートは次のとおりです。

RDS for Db2	RDS for MariaDB	RDS for MySQL	RDS for Oracle	RDS for PostgreSQL	RDS for SQL Server
50000	3306	3306	1521	5432	1433

RDS for SQL Server の場合、ポート `1234`, `1434`, `3260`, `3343`, `3389`, `47001`, および `49152-49156` は予約済みであり、DB インスタンスの作成時には使用できません。

DB インスタンスの作成

AWS マネジメントコンソール、AWS CLI または RDS API を使用して、Amazon RDS DB インスタンスを作成することができます。

> **Note**
>
> RDS for Db2 では、RDS for Db2 DB インスタンスを作成する前に、ライセンスモデルに必要な項目を設定することをお勧めします。詳細については、「Amazon RDS for Db2 のライセンスオプション」を参照してください。

コンソール

AWS マネジメントコンソール で [Easy Create (簡易作成)] を有効または無効にして、DB インスタンス を作成できます。[Easy create] を有効にして、DB エンジンタイプ、DB インスタンスサイ

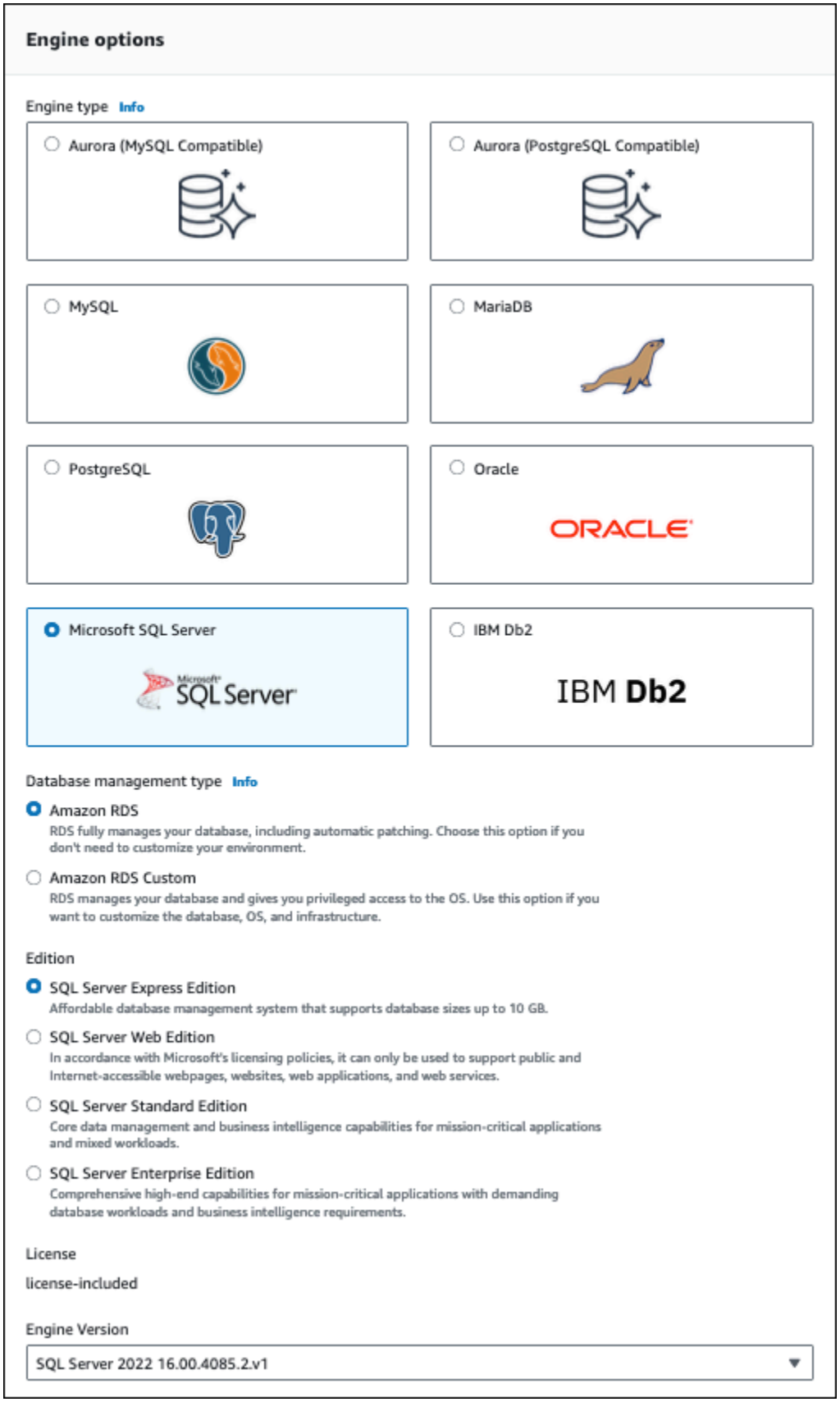
ズ、および DB インスタンス識別子のみを指定します。[Easy create] では、他の設定オプションにデフォルト設定を使用します。[Easy create] が有効になっていない場合は、データベースの作成時に、可用性、セキュリティ、バックアップ、メンテナンスなどの設定オプションを追加指定します。

> **Note**
>
> 次の手順では、[Standard Create (スタンダード作成)] が有効になっており、[Easy Create (簡易作成)] は有効になっていません。この手順では、Microsoft SQL Server を例として使用します。
> [Easy Create (簡易作成)] を使用し、各エンジンのサンプル DB インスタンスを作成して接続する例については、Amazon RDS のスタート方法 を参照してください。

DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[Databases] (データベース) を選択します。
4. [データベースの作成] を選択し、[標準作成] を選択します。
5. [エンジンタイプ] として、IBM Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle、または PostgreSQL を選択します。

   ここでは [Microsoft SQL Server] が示されています。


Engine options
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
IBM Db2
IBM Db2
Database management type Info
Amazon RDS
RDS fully manages your database, including automatic patching. Choose this option if you don't need to customize your environment.
Amazon RDS Custom
RDS manages your database and gives you privileged access to the OS. Use this option if you want to customize the database, OS, and infrastructure.
Edition
SQL Server Express Edition
Affordable database management system that supports database sizes up to 10 GB.
SQL Server Web Edition
In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services.
SQL Server Standard Edition
Core data management and business intelligence capabilities for mission-critical applications and mixed workloads.
SQL Server Enterprise Edition
Comprehensive high-end capabilities for mission-critical applications with demanding database workloads and business intelligence requirements.
License
license-included
Engine Version
SQL Server 2022 16.00.4085.2.v1

6. [データベース管理タイプ] では、Oracle または SQL Server を使用している場合は、[Amazon RDS] または [Amazon RDS Custom] を選択します。

   [Amazon RDS] がここに表示されています。RDS Custom については、「Amazon RDS Custom」を参照してください。

7. [エディション] では、Db2、Oracle、または SQL Server を使用している場合は、使用する DB エンジンのエディションを選択します。

   MySQL にはエディションのためのオプションが 1 つしかなく、MariaDB と PostgreSQL にはオプションがまったくありません。

8. [バージョン] で、エンジンのバージョンを選択します。

9. [テンプレート] で、ユースケースに合うテンプレートを選択します。[本番稼働用] を選択した場合、次のステップでは以下が既に選択されています。

   - [マルチ AZ] フェイルオーバーオプション
   - プロビジョンド IOPS SSD (io1)ストレージオプション
   - [Enable deletion protection (削除保護の有効化)] オプション

   本稼働環境では、これらの機能を使用することをお勧めします。

   > **Note**
   >
   > テンプレートの選択内容は、エディションごとに異なります。

10. [設定] セクションで、[認証情報の設定] を開きます。次に、以下の操作を実行します。

    a. (オプション)マスターユーザーネームの値を変更します。

    b. 次のいずれかの認証情報管理オプションを選択します。

       - で管理AWS Secrets Manager

         [暗号化キーを選択] で、Secrets Manager によって作成される KMS キーと自分で作成したキーのいずれかを選択します。

> **Note**
>
> 認証情報を管理するための最も安全な手法として、AWS Secrets Manager をお勧めします。別途 料金がかかります。詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。

- セルフマネージド

  パスワードを指定するには、[パスワードの自動生成] チェックボックスが選択されている場合はクリアします。[マスターパスワード] と [マスターパスワードを確認] に同じパスワードを入力します。

11. (オプション) この DB インスタンスのコンピューティングリソースへの接続を設定します。

    DB インスタンスの作成中に、Amazon EC2 インスタンスと新しい DB インスタンス間の接続を設定できます。詳細については、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。

12. [VPC セキュリティグループ (ファイアウォール)] の [接続] セクションで [新規作成] を選択した場合、ローカルコンピュータの IP アドレスにデータベースへのアクセスを許可するインバウンドルールを使用して VPC セキュリティグループが作成されます。

13. 残りのセクションで、DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。

14. [データベースの作成] を選択します。

    自動生成されたパスワードを使用することを選択した場合は、[データベース] ページに [認証情報の詳細の表示] ボタンが表示されます。

    DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

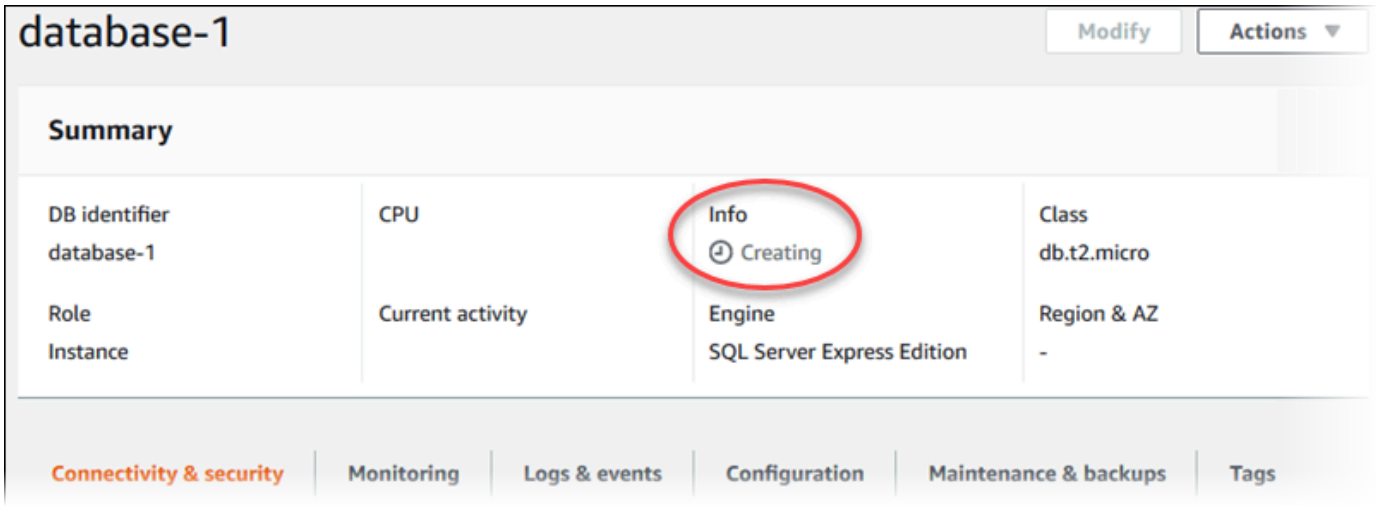
    マスターユーザーとして DB インスタンスに接続するには、表示されているユーザー名およびパスワードを使用します。

> **Important**
>
> マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。DB インスタンスが有効になった後にマスターユーザーのパスワードを変更する必要がある場合は、そのように DB インスタンス

を変更します。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

15. [Databases (データベース)] で、新しい DB インスタンスの名前を選択します。

    RDS コンソールに、新規の DB インスタンスの詳細が表示されます。DB インスタンスが作成されて使用できるようになるまで、DB インスタンスのステータスは [作成中] となります。ステータスが [Available] に変わると、DB インスタンスに接続できます。DB インスタンスクラスと割り当てられたストレージによっては、新しいインスタンスを使用できるようになるまで数分かかることがあります。



AWS CLI

> **Note**
>
> AWS Marketplace 経由の Db2 ライセンスを使用する場合は、まず AWS Marketplace のサブスクリプションを購入して、AWS マネジメントコンソール を使用して IBM に登録する必要があります。詳細については、「Db2 Marketplace サブスクリプションの購入と IBM での登録」を参照してください。

AWS CLIを使用して DB インスタンスを作成するには、以下のパラメータを指定して create-db-instance コマンドを呼び出します。

- `--db-instance-identifier`
- `--db-instance-class`

- --vpc-security-group-ids
- --db-subnet-group
- --engine
- --master-username
- --master-user-password-または---manage-master-user-password
- --allocated-storage
- --backup-retention-period

各設定の詳細については、「DB インスタンスの設定」を参照してください。

この例では、Microsoft SQL Server を使用しています。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --engine sqlserver-se \
    --db-instance-identifier mymsftsqlserver \
    --allocated-storage 250 \
    --db-instance-class db.t3.large \
    --vpc-security-group-ids mysecuritygroup \
    --db-subnet-group mydbsubnetgroup \
    --master-username masterawsuser \
    --manage-master-user-password \
    --backup-retention-period 3
```

Windows の場合:

```
aws rds create-db-instance ^
    --engine sqlserver-se ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 250 ^
    --db-instance-class db.t3.large ^
    --vpc-security-group-ids mysecuritygroup ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username masterawsuser ^
    --manage-master-user-password ^
    --backup-retention-period 3
```

このコマンドでは、次のような出力が生成されます。

```
DBINSTANCE  mydbinstance  db.t3.large  sqlserver-se  250  sa  creating  3  ****  n
 10.50.2789
SECGROUP  default  active
PARAMGRP  default.sqlserver-se-14  in-sync
```

RDS API

> **Note**
>
> AWS Marketplace 経由の Db2 ライセンスを使用する場合は、まず AWS Marketplace のサブスクリプションを購入して、AWS マネジメントコンソール を使用して IBM に登録する必要があります。詳細については、「Db2 Marketplace サブスクリプションの購入と IBM での登録」を参照してください。

Amazon RDS API を使用して DB インスタンスを作成するには、CreateDBInstance オペレーションを呼び出します。

各設定の詳細については、「DB インスタンスの設定」を参照してください。

DB インスタンスの設定

次の表は、DB インスタンスの作成時に選択する設定の詳細を示しています。この表には、各設定をサポートしている DB エンジンも示されています。

コンソール、create-db-instance CLI コマンド、や CreateDBInstance RDS API オペレーションを使用して、DB インスタンスを作成できます。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
ストレージ割り当て	DB インスタンスに割り当てるストレージの量 (ギビバイト(GiB)単位)。場合によっては、DB インスタンスに、データベースのサイズ以上のストレージを割り	CLI オプション: `--allocated-storage`	すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	当てると、I/O のパフォーマンスが改善することがあります。 詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。	API パラメータ: `AllocatedStorage`	
アーキテクチャ設定	[Oracle マルチテナントアーキテクチャ] を選択すると、RDS for Oracle はコンテナデータベース (CDB) を作成します。このオプションを選択しない場合、RDS for Oracle は非 CDB を作成します。非 CDB は、従来の Oracle データベースアーキテクチャを使用します。CDB にはプラグ可能なデータベース (PDB) を含められますが、非 CDB には含めることができません。 Oracle Database 21c は CDB アーキテクチャのみを使用します。Oracle Database 19c は、CDB アーキテクチャまたは非 CDB アーキテクチャのいずれかを使用できます。Oracle Database 19c よりも前のリリースでは、非 CDB アーキテクチャのみが使用されます。 詳細については、「RDS for Oracle CDB の概要」を参照してください。	CLI オプション: `--engine oracle-ee-cdb` (Oracle マルチテナント) `--engine oracle-se2-cdb` (Oracle マルチテナント) `--engine oracle-ee` (traditional) `--engine oracle-se2` (traditional) API パラメータ: `Engine`	Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
アーキテクチャ設定	これらの設定は、[アーキテクチャ設定] で、[Oracle マルチテナントアーキテクチャ] を選択した場合にのみ有効です。次のいずれかの設定を選択します。 • [マルチテナント設定] では、RDS for Oracle CDB インスタンスに、データベースのエディションと必要なオプションライセンスに応じて、1 ~ 30 個のテナントデータベースを含めることができます。Oracle データベースのコンテキストの場合、テナントデータベースは PDB です。アプリケーション PDB とプロキシ PDB はサポートされていません。 DB インスタンスは 1 つの初期テナントデータベースで作成されます。[テナントデータベース名]、[テナントデータベースマスターユーザー名]、[テナントデータベースマスターパスワード]、および [テナントデータベース文字セット] の値を選択します。 マルチテナント設定は永続的です。そのため、マルチテナント設定をシングルテナント設定に変換して戻すことはできません。マルチテナント設定でサポートされる最小リリース更新プログラム (RU) は、19.0.0.0.ru-2022-01.rur-2022.r1 です。	CLI オプション: `--multi-tenant` (マルチテナント設定) `--no-multi-tenant` (シングルテナント設定) API パラメータ: `MultiTenant`	Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	Note Amazon RDS 設定は、単なる Oracle DB エンジンではなく、Amazon RDS の機能であるため、「multitenant」ではなく「multi-tenant」と呼ばれています。同様に、RDS の用語「テナント」は、Oracle PDB だけでなく、RDS 設定のすべてのテナントを指します。RDS のドキュメントの「Oracle マルチテナント」という用語は、オンプレミスデプロイと RDS デプロイの両方に対応する Oracle データベース CDB アーキテクチャのみを指します。 • シングルテナント設定では、RDS for Oracle CDB には 1 つの PDB が含まれます。CDB を作成した際、デフォルトでこの設定になっています。最初の PDB を削除したり、さらに PDB を追加したりすることはできません。CDB のシングルテナント設定は後でマルチテナント設定に変換できますが、その後シングルテナント設定に戻すことはできません。		

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	どの設定を選択しても、CDB には初期 PDB が 1 つ含まれます。マルチテナント設定では、RDS API を使用して、後からさらに PDB を作成できます。 詳細については、「RDS for Oracle CDB の概要」を参照してください。		

マイナーバージョン自動アップグレード	[マイナーバージョン自動アップグレードの有効化] を選択すると、希望する DB エンジンのマイナーバージョンのアップグレードをリリースと同時に自動的に DB インスタンスに適用できます。これがデフォルトの動作です。Amazon RDS では、メンテナンスウィンドウでマイナーバージョンの自動アップグレードが実行されます。[マイナーバージョン自動アップグレードの有効化] を選択しなかった場合、新しいマイナーバージョンが利用可能になっても DB インスタンスは自動的にアップグレードされません。
	
	詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。
		CLI オプション:
		
		`--auto-minor-version-upgrade`
		
		`--no-auto-minor-version-upgrade`
		
		API パラメータ:
		
		`AutoMinorVersionUpgrade`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

アベイラビリティーゾーン	DB インスタンスのアベイラビリティーゾーン。デフォルト値 [No Preference] を使用します。ただし、特定のアベイラビリティーゾーンを指定する場合を除きます。
	
	詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。
		CLI オプション:
		
		`--availability-zone`
		
		API パラメータ:
		
		`AvailabilityZone`
			すべて

AWS KMS key	[Encryption] が [Enable encryption] に設定されている場合にのみ使用できます。この DB インスタンスの暗号化に使用する AWS KMS key を選択します。詳細については、「Amazon RDS リソースの暗号化」を参照してください。	CLI オプション:
		
		`--kms-key-id`
		
		API パラメータ:
		
		`KmsKeyId`
			すべて

AWS License Manager の設定	AWS License Manager ライセンス設定の名前を入力します。名前は 100 文字以下で、使用できるのは a ~ z、A ~ Z、0 ~ 9 のみです。 詳細については、「the section called “AWS License Manager との統合”」を参照してください。	CLI オプション: 詳細については、「AWS License Manager CLI」を参照してください。 API パラメータ: 詳細については、「AWS License Manager API」を参照してください。	Db2

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
バックアップレプリケーション	「別の AWS リージョンへのレプリケーションを有効にする」 を選択して、災害対策用の追加リージョンにバックアップを作成します。 次に、追加バックアップ先リージョンを選択します。	DB インスタンスの作成時には使用できません。AWS CLI または RDS API でクロスリージョンバックアップを有効にする方法については、「Amazon RDS のクロスリージョン自動バックアップの有効化」を参照してください。	Db2 Oracle PostgreSQL SQL Server

バックアップの保存期間	DB インスタンスの自動バックアップを保持する日数。重要な DB インスタンスでは、この値を**1** 以上に設定してください。
	
	詳細については、「バックアップの概要」を参照してください。
		CLI オプション:
		
		`--backup-retention-period`
		
		API パラメータ:
		
		`BackupRetentionPeriod`
			すべて

バックアップ ターゲット	選択して、AWS クラウド自動バックアップと手動スナップショットを親AWS リージョンに保存します。Outposts 点 (オンプレミス)を選択して、Outpost にローカルに保存します。
	
	このオプション設定は Outposts 上の RDS のみに適用されます。詳細については、「AWS Outposts の Amazon RDS での DB インスタンスの作成」を参照してください。
		CLI オプション:
		
		`--backup-target`
		
		API パラメータ:
		
		`BackupTarget`
			MySQL、PostgreSQL、SQL Server

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

バックアップ期間:	Amazon RDS が DB インスタンスのバックアップを自動的に作成する期間。データベースのバックアップを保持する期間を指定しない場合は、デフォルト値 [指定なし] を使用します。
	
	詳細については、「バックアップの概要」を参照してください。
		CLI オプション:
		
		`--preferred-backup-window`
		
		API パラメータ:
		
		`PreferredBackupWindow`
			すべて

認証局	DB インスタンスによって使用されるサーバー証明書の認定機関 (CA)。
	
	詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
		CLI オプション:
		
		`--ca-certificate-identifier`
		
		RDS API パラメータ:
		
		`CACertificateIdentifier`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

文字セット	DB インスタンスの文字セット。DB 文字セットのデフォルト値 AL32UTF8 は、Unicode 5.0 UTF-8 ユニバーサル文字セット用です。DB インスタンスの作成後に DB 文字セットを変更することはできません。
	
	シングルテナント設定では、デフォルト以外の DB 文字セットは PDB のみに影響し、CDB には影響しません。詳細については、「CDB アーキテクチャのシングルテナント設定」を参照してください。
	
	DB 文字セットは、NCHAR 文字セットと呼ばれる各国語文字セットとは異なります。DB 文字セットとは異なり、NCHAR 文字セットは、データベースのメタデータに影響を与えることなく、NCHAR データ型 (NCHAR、NVARCHAR2、NCLOB) 列のエンコーディングを指定します。
	
	詳細については、「RDS for Oracle 文字セット」を参照してください。
		CLI オプション:
		
		`--character-set-name`
		
		API パラメータ:
		
		`CharacterSetName`
			Oracle

照合	DB インスタンスのサーバーレベルの照合。
	
	詳細については、「Microsoft SQL Server のサーバーレベルの照合」を参照してください。
		CLI オプション:
		
		`--character-set-name`
		
		API パラメータ:
		
		`CharacterSetName`
			SQL Server

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

Copy tags to snapshots	このオプションは、スナップショットの作成時に、DB インスタンスタグを DB スナップショットにコピーします。
	
	詳細については、「Amazon RDS リソースのタグ付け」を参照してください。
		CLI オプション:
		
		`--copy-tags-to-snapshot`
		
		`--no-copy-tags-to-snapshot`
		
		RDS API パラメータ:
		
		`CopyTagsToSnapshot`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

データベース認証	使用するデータベース認証オプション。
	
	データベースパスワードのみを使用してデータベースのユーザーを認証するには、[パスワード認証] を選択します。
	
	ユーザーとロールでデータベースパスワードとユーザー認証情報を使用してデータベースユーザーを認証するには、[Password and IAM DB authentication] (パスワードと IAM DB 認証) を選択します。詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。このオプションは、MySQL および PostgreSQL でのみサポートされています。
	
	AWS Managed Microsoft AD を使用して作成された Directory Service でデータベースパスワードと Kerberos 認証を使用してデータベースユーザーを認証するには、[パスワードと Kerberos 認証] を選択します。次に、ディレクトリを作成するか、[ディレクトリの作成] を選択します。
	
	詳細については、以下のいずれかを参照してください。
	
	• RDS for Db2 による AWS Managed Active Directory の操作
		IAM:
		
		CLI オプション:
		
		`--enable-iam-database-authentication`
		
		`--no-enable-iam-database-authentication`
		
		RDS API パラメータ:
		
		`EnableIAMDatabaseAuthentication`
		
		Kerberos:
		
		CLI オプション:
		
		`--domain`
		
		`--domain-iam-role-name`
		
		RDS API パラメータ:
		
		`Domain`
		
		`DomainIAMRoleName`
			認証タイプによって異なる

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	• Amazon RDS for MySQL での Kerberos 認証の使用 • Amazon RDS for Oracle の Kerberos 認証の設定 • Amazon RDS for PostgreSQL で Kerberos 認証を使用する		
データベース管理タイプ	環境をカスタマイズする必要がなければ、Amazon RDSを選択します。 データベース、OS、およびインフラストラクチャをカスタマイズする場合、Amazon RDS Customを選択します。詳細については、「Amazon RDS Custom」を参照してください。	CLI と API では、データベースエンジンタイプを指定します。	Oracle SQL Server

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

データベースポート	DB インスタンスにアクセスするために経由するポート。デフォルトのポートが示されています。
	
	Note
	会社のファイアウォールによっては、デフォルトの MariaDB、MySQL、および PostgreSQL ポートへの接続がブロックされる場合があります。会社のファイアウォールがデフォルトのポートをブロックする場合は、お客様の DB インスタンス用に別のポートを選択します。
		CLI オプション:
		
		`--port`
		
		RDS API パラメータ:
		
		`Port`
			すべて

DB エンジンバージョン	使用するデータベースエンジンのバージョン。	CLI オプション:
		
		`--engine-version`
		
		RDS API パラメータ:
		
		`EngineVersion`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

DB インスタンスクラス:	DB インスタンスの設定。例えば、「db.t3.small」 DBインスタンスクラスには、2 GiBメモリ、2 vCPU、1 つの仮想コア、可変 ECU、および中程度の I/O 容量があります。
	
	可能であれば、一般的なクエリの作業セットをメモリに保持できる十分な大きさの DB インスタンスクラスを選択します。作業セットがメモリに保持されていると、システムによるディスクへの書き込みが回避され、これによりパフォーマンスが向上します。詳細については、「DB インスタンスクラス」を参照してください。
	
	Oracle 用の RDS では、[追加のメモリ設定を含める] を選択できます。これらの設定は、vCPU へのメモリの比率が高い場合のために最適化されています。例えば db.r5.6xlarge.tpc2.mem4x は、コアあたり 2 つのスレッド (tpc2)、スタンダードdb.r5.6xlarge DBインスタンスの 4 倍のメモリを持つ db.r5.8x DBインスタンスです。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。
		CLI オプション:
		
		`--db-instance-class`
		
		RDS API パラメータ:
		
		`DBInstanceClass`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

DB インスタンス識別子:	DB インスタンスの名前。オンプレミスのサーバーに名前を付けるのと同様に、DB インスタンスに名前を付けます。DB インスタンス識別子は、英数字 63 文字まで含めることができ、選択した AWS リージョン内で自分のアカウントに対して一意であることが必要です。	CLI オプション:
		`--db-instance-identifier`
		RDS API パラメータ:
		`DBInstanceIdentifier`
			すべて

DB パラメータグループ:	DB インスタンスのパラメータグループ。デフォルトのパラメータグループを選択するか、カスタムパラメータグループを作成できます。
	RDS for Db2 の BYOL モデルを使用している場合は、DB インスタンスを作成する前に、まず。IBM Site ID と IBM Customer ID を含むカスタムパラメータグループを作成する必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL)」を参照してください。
	詳細については、「Amazon RDS のパラメータグループ」を参照してください。
		CLI オプション:
		`--db-parameter-group-name`
		RDS API パラメータ:
		`DBParameterGroupName`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

DB サブネットグループ	DB クラスターで使用する DB サブネットグループ。
	既存の DB サブネットグループを使用するには、[Choose existing] (既存を選択) を選択します。次に、[Existing DB subnet groups] (既存の DB サブネットグループ) ドロップダウンリストから必要なサブネットグループを選択します。
	
	RDS が互換性のある DB サブネットグループを選択できるようにするには、[Automatic setup] (自動セットアップ) を選択します。存在しない場合、RDS はクラスターの新しいサブネットグループを作成します。
	
	詳細については、「DB サブネットグループの使用」を参照してください。
		CLI オプション:
		
		`--db-subnet-group-name`
		
		RDS API パラメータ:
		
		`DBSubnetGroupName`
			すべて

専用ログボリューム	専用ログボリューム (DLV) を使用して、データベーステーブルを含むボリュームとは別のストレージボリュームにデータベーストランザクションログを保存します。
	
	詳細については、「専用ログボリューム (DLV) を使用する」を参照してください。
		CLI オプション:
		
		`--dedicated-log-volume`
		
		RDS API パラメータ:
		
		`DedicatedLogVolume`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

削除保護	DB インスタンスが削除されないようにするには [Enable deletion protection (削除保護の有効化)] を選択します。AWS マネジメントコンソールで本稼働 DB インスタンスを作成する場合は、削除保護がデフォルトで有効になっています。
	
	詳細については、「DB インスタンスを削除する」を参照してください。
		CLI オプション:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API パラメータ:
		
		`DeletionProtection`
			すべて

暗号化:	この DB インスタンスを保管時に暗号化するには、[Enable Encryption] を選択します。
	
	詳細については、「Amazon RDS リソースの暗号化」を参照してください。
		CLI オプション:
		
		`--storage-encrypted`
		
		`--no-storage-encrypted`
		
		RDS API パラメータ:
		
		`StorageEncrypted`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

拡張モニタリング	[拡張モニタリングを有効にする] を選択すると、DB インスタンスが実行されているオペレーティングシステムに対してリアルタイムでのメトリクスの収集が有効になります。
	
	詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。
		CLI オプション:
		
		`--monitoring-interval`
		
		`--monitoring-role-arn`
		
		RDS API パラメータ:
		
		`MonitoringInterval`
		
		`MonitoringRoleArn`
			すべて

エンジンのタイプ	この DB インスタンスに使用するデータベースエンジンを選択します。	CLI オプション:
		
		`--engine`
		
		RDS API パラメータ:
		
		`Engine`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

初期データベース名	DB インスタンス上のデータベースの名前。名前を指定しない場合、Amazon RDS は DB インスタンスにデータベースを作成しません (Oracle および PostgreSQL を除きます)。データベースエンジンによって予約された単語を名前にすることはできません。また、DB エンジンに応じて他の制約があります。
	
	Db2:
	
	• 1 ~ 8 個の英数字を使用する必要があります。
	• a ~ z、A ~ Z、@、$、または # で始まり、a ~ z、A ~ Z、0 ~ 9、_、@、#、または $ が続く必要があります。
	• スペースを含めることはできません。
	• 詳細については、「その他の考慮事項」を参照してください。
	
	MariaDB および MySQL:
	
	• 1 ~ 64 個の英数字を使用する必要があります。
	
	Oracle:
	
	•
		CLI オプション:
		
		`--db-name`
		
		RDS API パラメータ:
		
		`DBName`
			SQL Server を除くすべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	1 ~ 8 個の英数字を使用する必要があります。 • NULL は使用できません。デフォルト値は ORCL です。 • 先頭は文字にする必要があります。 PostgreSQL: • 1 ~ 63 個の英数字を使用する必要があります。 • 先頭は英字またはアンダースコアにする必要があります。後続の文字には、英字、アンダースコア、または数字 (0 ~ 9) を含めることができます。 • 初期のデータベース名は postgres です。		

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

ライセンス	ライセンスモデルの有効値:
	• Db2 の bring-your-own-license または marketplace-license。
	• MariaDB の general-public-license
	• Microsoft SQL Server の license-included
	• MySQL の general-public-license
	• Oracle の license-included または bring-your-own-license
	• PostgreSQL の postgresql-license
		CLI オプション:
		`--license-model`
		RDS API パラメータ:
		`LicenseModel`
			すべて

ログのエクスポート	Amazon CloudWatch Logs に発行するデータベースログファイルのタイプ。
	詳細については、「Amazon CloudWatch Logs へのデータベースログの発行」を参照してください。
		CLI オプション:
		`--enable-cloudwatch-logs-exports`
		RDS API パラメータ:
		`EnableCloudwatchLogsExports`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

メンテナンスウィンドウ	DB インスタンスへの変更保留が適用される 30 分単位のウィンドウ。期間が重要ではない場合は、[No Preference] を選択します。
	
	詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。
		CLI オプション:
		
		`--preferred-maintenance-window`
		
		RDS API パラメータ:
		
		`PreferredMaintenanceWindow`
			すべて

でマスター認証情報を管理するAWS Secrets Manager	[Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択して、Secrets Manager でユーザーのパスワードをシークレットに管理します。
	
	オプションで、シークレットを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
	
	詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。
		CLI オプション:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		RDS API パラメータ:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

マスターパスワード:	マスターユーザーアカウントのパスワード。パスワードには、DB エンジンに応じて、次の数の印刷可能な ASCII 文字 (/、"、スペース、および @ を除く) が含まれます。
	• Db2: 8 ~ 255
	• Oracle: 8 ~ 30 文字
	• MariaDB および MySQL: 8 ~ 41 文字
	• SQL Server および PostgreSQL: 8 ~ 128 文字
		CLI オプション:
		`--master-user-password`
		RDS API パラメータ:
		`MasterUserPassword`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

マスターユーザーネーム	マスターユーザー名。この名前で DB インスタンスにログインすると、データベースに関するすべての権限を持つことになります。以下の命名制限があることに注意してください:
	• 名前には、1 ~ 16 文字の英数字とアンダースコアを使用できます。
	• 1 字目は文字である必要があります。
	• 名前にはデータベースエンジンの予約語を使用できません。
	
	DB インスタンスの作成後にマスターユーザー名を変更することはできません。
	
	Db2 では、セルフマネージド Db2 インスタンス名と同じマスターユーザー名を使用することをお勧めします。
	
	マスターユーザーに付与される権限の詳細については、マスターユーザーアカウント権限を参照してください。
		CLI オプション:
		
		`--master-username`
		
		RDS API パラメータ:
		
		`MasterUsername`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

Microsoft SQL Server の Windows 認証	Microsoft SQL Server の Windows 認証を有効化し、[Browse Directory (ディレクトリの参照)] をクリックして、許可されたドメインユーザーが Windows 認証を使用してこの SQL Server インスタンスで認証できるようにするディレクトリを選択します。	CLI オプション:
		`--domain`
		`--domain-iam-role-name`
		RDS API パラメータ:
		`Domain`
		`DomainIAMRoleName`
			SQL Server

マルチ AZ 配置:	[Create a standby instance (スタンバイインスタンスを作成する)] を選択して、フェイルオーバーサポート用に DB インスタンスのパッシブセカンダリレプリカを別のアベイラビリティーゾーンに作成します。本稼働環境のワークロードには、高可用性を維持するためにマルチ AZ をお勧めします。
	開発およびテスト用に、[Do not create a standby instance (スタンバイインスタンスを作成しない)] を選択することもできます。
	詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。
		CLI オプション:
		`--multi-az`
		`--no-multi-az`
		RDS API パラメータ:
		`MultiAZ`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

各国語文字セット (NCHAR)	DB インスタンスの各国語文字セット (通称は NCHAR 文字セット)。各国語文字セットは、AL16UTF16 (デフォルト) または UTF-8 のいずれかに設定できます。DB インスタンスの作成後に各国語文字セットを変更することはできません。
	
	各国語文字セットは、DB 文字セットとは異なります。DB 文字セットとは異なり、各国語文字セットは、データベースのメタデータに影響を与えることなく、NCHAR データ型 (NCHAR、NVARCHAR2、NCLOB) 列のエンコーディングのみを指定します。
	
	詳細については、「RDS for Oracle 文字セット」を参照してください。
		CLI オプション:
		
		`--nchar-character-set-name`
		
		API パラメータ:
		
		`NcharCharacterSetName`
			Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

ネットワークの種類	DB インスタンスでサポートされている IP アドレス設定プロトコル。
	
	リソースが、インターネットプロトコルバージョン 4 (IPv4) アドレス設定プロトコル経由でのみ DB インスタンスと通信できるように指定する IPv4 (デフォルト)。
	
	リソースが IPv4、インターネットプロトコルバージョン 6 (IPv6)、またはその両方で DB インスタンスと通信できるように指定するデュアルスタックモード。IPv6 アドレス設定プロトコルで DB インスタンスと通信する必要があるリソースがある場合は、デュアルスタックモードを使用します。また、IPv6 CIDR ブロックを、指定した DB サブネットグループ内のすべてのサブネットに関連付けてください。
	
	詳細については、「Amazon RDS IP アドレス指定」を参照してください。
		CLI オプション:
		
		`--network-type`
		
		RDS API パラメータ:
		
		`NetworkType`
			すべて

オプショングループ	DB インスタンスのオプショングループ。デフォルトオプショングループを選択するか、カスタムオプショングループを作成できます。
	
	詳細については、「オプショングループを使用する」を参照してください。
		CLI オプション:
		
		`--option-group-name`
		
		RDS API パラメータ:
		
		`OptionGroupName`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

Performance Insights	[Performance Insights の有効化] を選択すると、DB インスタンスの負荷をモニタリングし、データベースパフォーマンスの分析とトラブルシューティングを行うことができます。
	
	保持期間を選択して、保持するパフォーマンスインサイトデータ履歴の量を決定します。保持期間の設定は、デフォルト (7 日間) です。パフォーマンスデータをさらに長期間保持するには、1～24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。
	
	このデータベースボリュームの暗号化に使用されるキーを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
	
	詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。
		CLI オプション:
		
		`--enable-performance-insights`
		
		`--no-enable-performance-insights`
		
		`--performance-insights-retention-period`
		
		`--performance-insights-kms-key-id`
		
		RDS API パラメータ:
		
		`EnablePerformanceInsights`
		
		`PerformanceInsightsRetentionPeriod`
		
		`PerformanceInsightsKMSKeyId`
			Db2 を除くすべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

プロビジョンド IOPS	DB インスタンスのプロビジョンド IOPS (毎秒ごとの I/O オペレーション) の値。この設定は、[Storage type] (ストレージタイプ) で次のいずれかを選択した場合にのみ使用できます。
	• 汎用 SSD (gp3)
	• プロビジョンド IOPS SSD (io1)
	• プロビジョンド IOPS SSD (io2)
	詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。
		CLI オプション:
		`--iops`
		RDS API パラメータ:
		`Iops`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

パブリックアクセス	パブリック IP アドレスを DB インスタンスに割り当てる場合は [Yes (はい)] を選択します。これは、VPC の外部でアクセスできることを意味します。パブリックにアクセス可能となるよう、DB インスタンスは、VPC のパブリックサブネット内にある必要があります。
	
	DB インスタンスを VPC 内からのみアクセス可能にするには、[No] を選択します。
	
	詳細については、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。
	
	VPC の外部から DB インスタンスに接続するには、DB インスタンスがパブリックにアクセスできる必要があります。また、DB インスタンスのセキュリティグループのインバウンドルールを使用してアクセスを許可する必要があります。さらに、他の要件を満たす必要があります。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。
	
	DB インスタンスがパブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続または Direct Connect 接続を使用してプライベートネットワークからアクセスします。詳細については、
		CLI オプション:
		
		`--publicly-accessible`
		
		`--no-publicly-accessible`
		
		RDS API パラメータ:
		
		`PubliclyAccessible`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン
	「ネットワーク間のトラフィックのプライバシー」を参照してください。		

RDS 延長サポート	RDS 標準サポート終了日を過ぎてもサポートされているメジャーエンジンバージョンを 引き続き実行するには、[RDS 延長サポートを有効にする] を選択します。
	
	DB インスタンスを作成する場合、Amazon RDS はデフォルトで RDS 延長サポートを選択します。RDS 標準サポート終了日後に新しい DB インスタンスが作成されて RDS 延長サポートの料金が発生するのを避けるには、この設定を無効にします。既存の DB インスタンスについて、RDS 延長サポートの課金開始日以前に料金が発生することはありません。
	
	詳細については、「Amazon RDS の Amazon RDS 延長サポート」を参照してください。
		CLI オプション:
		
		`--engine-lifecycle-support`
		
		RDS API パラメータ:
		
		`EngineLifecycleSupport`
			MySQL
			
			PostgreSQL

RDS Proxy	[Create an RDS Proxy] (RDS Proxy の作成) を選択して、DB インスタンスにプロキシを作成します。Amazon RDS は、プロキシの IAM ロールと Secrets Manager シークレットを自動的に作成します。 詳細については、「Amazon RDS Proxy」を参照してください。	DB インスタンスの作成時には使用できません。	MariaDB MySQL PostgreSQL

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

ストレージのオートスケーリング	[Enable storage autoscaling (ストレージのオートスケーリングを有効にする)] - DB インスタンスのストレージスペースが不足しないように、必要に応じて Amazon RDS のストレージを自動的に増やせるようにします。
	
	[Maximum storage threshold (ストレージの最大しきい値)] を使用して、Amazon RDS で DB インスタンスのストレージを自動的に増やすための上限を設定します。デフォルトは 1,000 GiB です。
	
	詳細については、「Amazon RDS ストレージの自動スケーリングによる容量の自動管理」を参照してください。
		CLI オプション:
		
		`--max-allocated-storage`
		
		RDS API パラメータ:
		
		`MaxAllocatedStorage`
			すべて

ストレージスループット	DB インスタンスのストレージスループット値。この設定は、[Storage type] (ストレージタイプ) に汎用 SSD (gp3) を選択した場合にのみ使用できます。
	
	詳細については、「gp3 ストレージ (推奨)」を参照してください。
		CLI オプション:
		
		`--storage-throughput`
		
		RDS API パラメータ:
		
		`StorageThroughput`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

ストレージタイプ	DB インスタンスのストレージタイプ。
	
	汎用 SSD (gp3) を選択すると、[Advanced settings] (詳細設定) で追加のプロビジョンド IOPS とストレージスループットをプロビジョニングできます。
	
	[プロビジョンド IOPS SSD (io1)] または [プロビジョンド IOPS SSD (io2)] を選択する場合は、[プロビジョンド IOPS] の値を入力します。
	
	詳細については、「Amazon RDS ストレージタイプ」を参照してください。
		CLI オプション:
		
		`--storage-type`
		
		RDS API パラメータ:
		
		`StorageType`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

追加のストレージボリューム	RDS for Oracle または RDS for SQL Server DB インスタンスには、最大 3 つのストレージボリュームを追加できます。gp3 または io2 ストレージタイプを使用するように各追加ボリュームを設定できます。また、割り当てられたストレージ、IOPS、スループットのさまざまな設定を指定して、ワークロードの要件に合わせて最適化することもできます。
	
	ボリューム名は、`rdsdbdata2` 、`rdsdbdata3` 、または `rdsdbdata4` として指定してください。
	
	プライマリボリュームと追加ボリュームの合計ストレージは 256 TiB を超えることはできません。
		CLI オプション:
		
		`--additional-storage-volumes`
		
		RDS API パラメータ:
		
		`AdditionalStorageVolumes`
			Oracle と SQL Server

サブネットグループ	この DB インスタンスに関連付ける DB サブネットグループ。
	
	詳細については、「DB サブネットグループの使用」を参照してください。
		CLI オプション:
		
		`--db-subnet-group-name`
		
		RDS API パラメータ:
		
		`DBSubnetGroupName`
			すべて

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

テナントデータベース名	Oracle アーキテクチャのマルチテナント設定における最初の PDB の名前。この設定は、[アーキテクチャ設定] で [マルチテナント設定] を選択した場合にのみ使用できます。
	
	テナントデータベース名は、RDSCDB という名前の CDB の名前と異っている必要があります。CDB 名を変更することはできません。
		CLI オプション:
		
		`--db-name`
		
		RDS API パラメータ:
		
		`DBName`
			Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

テナントデータベースのマスターユーザー名	マスターユーザー名を使って、すべてのデータベース権限でテナントデータベース (PDB) にログインする際に使用する名前。この設定は、[アーキテクチャ設定] で [マルチテナント設定] を選択した場合にのみ使用できます。
	
	以下の命名制限があることに注意してください:
	
	• 名前には、1 ~ 16 文字の英数字とアンダースコアを使用できます。
	• 1 字目は文字である必要があります。
	• 名前にはデータベースエンジンの予約語を使用できません。
	
	次のことはできません。
	
	• テナントデータベースを作成した後に、テナントマスターのユーザー名を変更します。
	• テナントマスターユーザー名を使用して CDB にログインします。
		CLI オプション:
		
		`--master-username`
		
		RDS API パラメータ:
		
		`MasterUsername`
			Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

テナントデータベースのマスターパスワード	テナントデータベース (PDB) のマスターユーザーアカウントのパスワード。この設定は、[アーキテクチャ設定] で [マルチテナント設定] を選択した場合にのみ使用できます。
	
	パスワードには、/、"、スペース、および @ を除く 8 ~ 30 文字の印刷可能な ASCII 文字を使用します。
		CLI オプション:
		
		`--master-password`
		
		RDS API パラメータ:
		
		`MasterPassword`
			Oracle

テナントデータベース文字セット	初期テナントデータベースの文字セット。この設定は、[アーキテクチャ設定] で [マルチテナント設定] を選択した場合にのみ使用できます。RDS for Oracle CDB インスタンスのみがサポートされています。
	
	テナントデータベース文字セットのデフォルト値 AL32UTF8 は、Unicode 5.0 UTF-8 ユニバーサル文字セット用です。CDB の文字セットとは異なるテナントデータベース文字セットを選択できます。
	
	詳細については、「RDS for Oracle 文字セット」を参照してください。
		CLI オプション:
		
		`--character-set-name`
		
		RDS API パラメータ:
		
		`CharacterSetName`
			Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

テナントデータベース各国語文字セット	通称 NCHAR 文字セットと呼ばれる、テナントデータベースの各国語文字セット。この設定は、[アーキテクチャ設定] で [マルチテナント設定] を選択した場合にのみ使用できます。RDS for Oracle CDB インスタンスのみがサポートされています。
	
	各国語文字セットは、AL16UTF16 (デフォルト) または UTF-8 のいずれかに設定できます。テナントデータベースの作成後に各国語文字セットを変更することはできません。
	
	テナントデータベースの各国語文字セットは、テナントデータベース文字セットとは異なります。各国文字セットは、NCHAR データ型 (NCHAR、NVARCHAR2 および NCLOB) を使用する列にのみエンコーディングを指定し、データベースのメタデータには影響しません。
	
	詳細については、「RDS for Oracle 文字セット」を参照してください。
		CLI オプション:
		
		`--nchar-character-set-name`
		
		API パラメータ:
		
		`NcharCharacterSetName`
			Oracle

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	サポートされている DB エンジン

タイムゾーン	DB インスタンスのタイムゾーン。タイムゾーンを選択しない場合、DB インスタンスはデフォルトのタイムゾーンを使用します。DB インスタンスの作成後にタイムゾーンを変更することはできません。
	
	詳細については、「Amazon RDS for Db2 DB インスタンスのローカルタイムゾーン」および「Microsoft SQL Server DB インスタンスのローカルタイムゾーン」を参照してください。
		CLI オプション:
		
		`--timezone`
		
		RDS API パラメータ:
		
		`Timezone`
			Db2
			
			SQL Server
			
			RDS Custom for SQL Server

仮想プライベートクラウド (VPC)。	この DB インスタンスと関連付ける Amazon VPC サービスに基づく VPC。 詳細については、「Amazon VPC と Amazon RDS」を参照してください。	CLI と API の場合は、VPC セキュリティグループ ID を指定します。	すべて

VPC セキュリティグループ (ファイアウォール)	DB インスタンスに関連付けるセキュリティグループ。
	
	詳細については、「VPC セキュリティグループの概要」を参照してください。
		CLI オプション:
		
		`--vpc-security-group-ids`
		
		RDS API パラメータ:
		
		`VpcSecurityGroupIds`
			すべて

AWS CloudFormation での Amazon RDS リソースの作成

Amazon RDS は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。必要なすべての AWS リソース (DB インスタンスや DB パラメータグループなど) を記述するテンプレートを作成し、CloudFormation はそれらのリソースをプロビジョニングして設定します。

CloudFormation を使用すると、テンプレートを再利用して RDS リソースを同じように繰り返してセットアップできます。リソースを一度記述すると、同じリソースを複数の AWS アカウントおよびリージョンで何度でも繰り返してプロビジョニングできます。

RDS とCloudFormationテンプレート

CloudFormation テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートには、CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、CloudFormation デザイナー を使えば、CloudFormation テンプレートを使いこなすことができます。詳細については、AWS CloudFormation ユーザーガイドの「CloudFormation Designer とは」を参照してください。

RDS は、CloudFormation でのリソースの作成をサポートしています。これらのリソースの JSON テンプレートと YAML テンプレートの例を含む詳細については、AWS CloudFormation ユーザーガイドの「RDS リソースタイプのリファレンス」を参照してください。

の詳細情報CloudFormation

CloudFormation の詳細については、以下のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

Amazon RDS DB インスタンスへの接続

DB インスタンスに接続する前に、DB インスタンスを作成する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。Amazon RDS によって DB インスタンスがプロビジョニングされたら、標準のクライアントアプリケーションまたは DB エンジン用のユーティリティを使用して DB インスタンスに接続します。接続文字列では、DB インスタンスのエンドポイントの DNS アドレスをホストパラメータとして指定します。また、ポートパラメータとして、DB インスタンスのエンドポイントのポート番号を指定します。

Amazon RDS DB インスタンスの接続情報の検索、または VPC 内の DB インスタンスにアクセスするためのシナリオの詳細については、次のトピックを参照してください。

- Amazon RDS DB インスタンスの接続情報の検索
- VPC の DB インスタンスにアクセスするシナリオ

Amazon RDS DB インスタンスの接続情報の検索

DB インスタンスの接続情報には、エンドポイント、ポート、およびマスターユーザーなどの有効なデータベースユーザーが含まれます。例えば、MySQL DB インスタンスの場合、エンドポイントの値が `mydb.123456789012.us-east-1.rds.amazonaws.com` であるとします。この場合、ポート値は `3306` であり、データベースユーザーは `admin` です。この情報を考慮して、接続文字列に次の値を指定します。

- ホスト、ホスト名または DNS 名には、`mydb.123456789012.us-east-1.rds.amazonaws.com` を指定します。
- ポートで、`3306` を指定します。
- ユーザーには、`admin` を指定します。

エンドポイントは DB インスタンスごとに一意であり、ポートとユーザーの値はさまざまです。次のリストは、各 DB エンジンの最も一般的なポートを示しています。

- Db2 – 50000
- MariaDB – 3306
- Microsoft SQL Server – 1433
- MySQL – 3306
- Oracle – 1521

- PostgreSQL – 5432

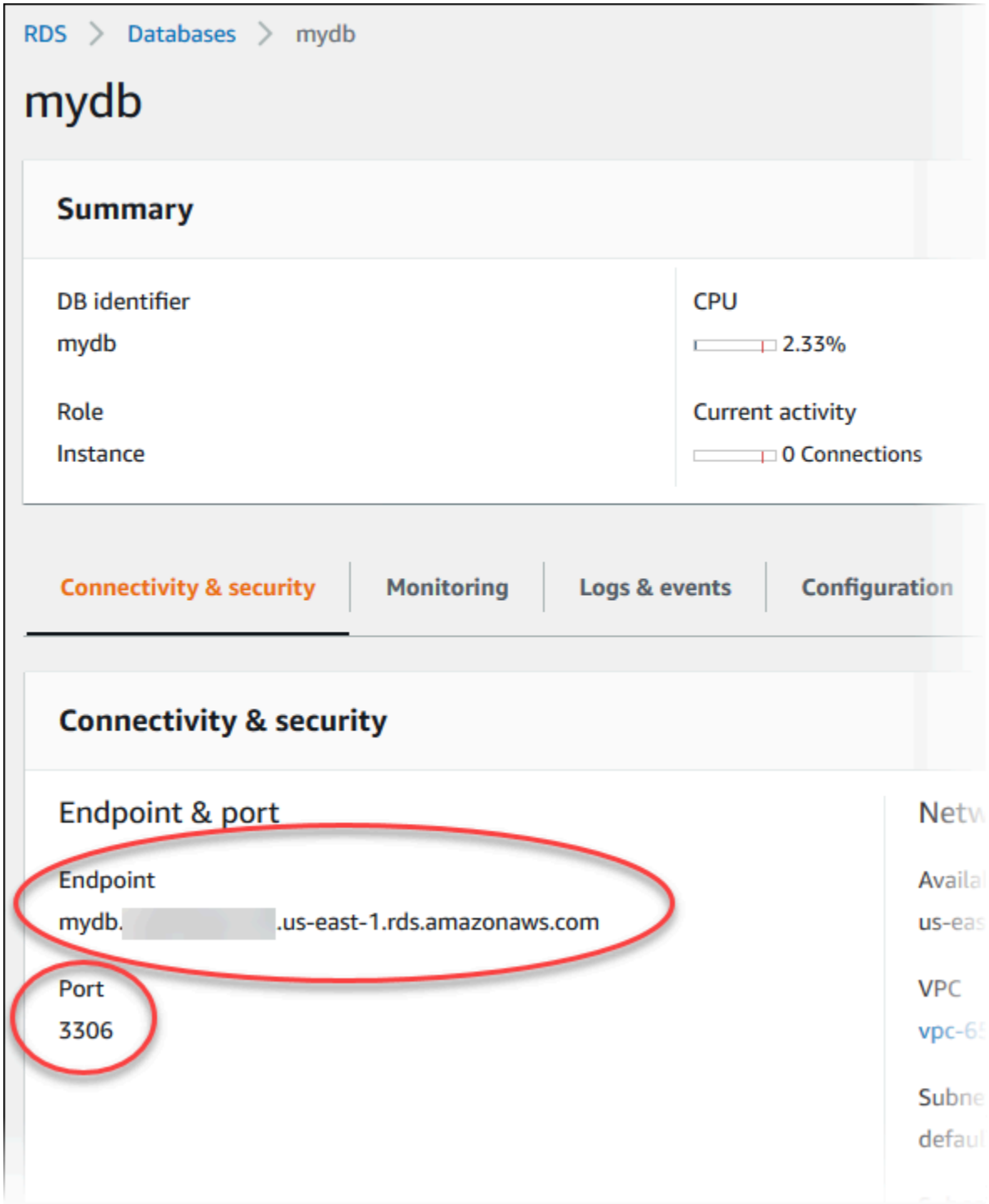
DB インスタンスに接続するには、DB エンジンの任意のクライアントを使用します。例えば、mysql ユーティリティを使用して MariaDB または MySQL DB インスタンスに接続できます。Microsoft SQL Server Management Studio を使用して、SQL Server の DB インスタンスに接続できます。Oracle SQL Developer を使用して Oracle DB インスタンスに接続したりできます。同様に、psql コマンドラインユーティリティを使用して、PostgreSQL DB インスタンスに接続できます。

DB インスタンスの接続情報を探すには、AWS マネジメントコンソール を使用します。また、AWS Command Line Interface (AWS CLI) describe-db-instances コマンドまたは RDS API DescribeDBInstances オペレーションを使用することもできます。

コンソール

AWS マネジメントコンソール で DB インスタンスの接続情報を探すには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択して DB インスタンスのリストを表示します。
3. DB インスタンスの名前を選択して、その詳細を表示します。
4. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



5. マスターユーザー名を見つける必要がある場合は、[設定] タブを選択し、[マスターユーザー名] の値を表示します。

AWS CLI

AWS CLI を使用して DB インスタンスの接続情報を検索するには、describe-db-instances コマンドを呼び出します。呼び出しで、DB インスタンス ID、エンドポイント、ポート、マスターユーザー名をクエリします。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows の場合:

```
aws rds describe-db-instances ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

出力は次のようになります。

```
[
    [
        "mydb",
        "mydb.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "myoracledb",
        "myoracledb.123456789012.us-east-1.rds.amazonaws.com",
        1521,
        "dbadmin"
    ],
    [
        "mypostgresqldb",
        "mypostgresqldb.123456789012.us-east-1.rds.amazonaws.com",
        5432,
        "postgresadmin"
    ]
]
```

RDS API

Amazon RDS API を使用して DB インスタンスの接続情報を検索するには、DescribeDBInstances オペレーションを呼び出します。出力で、エンドポイントアドレス、エンドポイントポート、およびマスターユーザー名の値を検索します。

VPC の DB インスタンスにアクセスするシナリオ

Amazon Virtual Private Cloud (Amazon VPC) を使用すると、Amazon RDS DB インスタンスなどの AWS リソースを仮想プライベートクラウド (VPC) で起動できます。Amazon VPC を使用する場合、仮想ネットワーキング環境を制御できます。独自の IP アドレスの範囲を選択し、サブネットを作成してルーティングおよびアクセスコントロールリストを設定できます。

VPC セキュリティグループは、VPC 内の DB インスタンスへのアクセスを制御します。VPC セキュリティグループの各ルールにより、その VPC セキュリティグループに関連付けられている VPC 内の DB インスタンスへのアクセスを特定のソースに許可できます。ソースとしては、アドレスの範囲 (203.0.113.0/24 など) または別の VPC セキュリティグループを指定できます。VPC セキュリティグループをソースとして指定すると、ソース VPC セキュリティグループを使用するすべてのインスタンス (通常はアプリケーションサーバー) からの受信トラフィックを許可することになります。

DB インスタンスに接続する前に、お客様のユースケース用に VPC を設定します。以下は、VPC の DB インスタンスにアクセスするための以下の一般的なシナリオです。

- 同じ VPC 内の Amazon EC2 インスタンスからアクセスする VPC 内の DB インスタンス – VPC 内の DB インスタンスの一般的な用途は、同じ VPC 内の EC2 インスタンスで実行されるアプリケーションサーバーとデータを共有することです。EC2 インスタンスは、DB インスタンスと対話するアプリケーションでウェブサーバーを実行することがあります。
- 別の VPC 内の EC2 インスタンスからアクセスする VPC 内の DB インスタンス – DB インスタンスが、アクセスに使用している EC2 インスタンスとは別の VPC にある場合もあります。その場合は、VPC ピアリングを使用して DB インスタンスにアクセスできます。
- インターネットを介してクライアントアプリケーションからアクセスする VPC 内の DB インスタンス – インターネット経由でクライアントアプリケーションから VPC 内の DB インスタンスにアクセスするには、1 つのパブリックサブネットを持つ VPC を設定します。また、インターネット経由の通信を有効にするために、インターネットゲートウェイを設定します。

  VPC の外部から DB インスタンスに接続するには、DB インスタンスがパブリックにアクセスできる必要があります。また、DB インスタンスのセキュリティグループのインバウンドルールを使用してアクセスを許可し、その他の要件を満たしている必要があります。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

- プライベートネットワークからアクセスされる VPC 内の DB インスタンス - DB インスタンスにパブリックにアクセスできない場合は、次のいずれかのオプションを使用してプライベートネットワークからアクセスできます。
  - AWS サイト間 VPN 接続
  - Direct Connect 接続
  - AWS Client VPN 接続

詳細については、「VPC の DB インスタンスにアクセスするシナリオ」を参照してください。

AWS ドライバーを使用した DB インスタンスへの接続

AWS のドライバースイートは、スイッチオーバーとフェイルオーバーの時間の短縮、AWS Secrets Manager、AWS Identity and Access Management (IAM)、フェデレーティッド ID での認証をサポートするように設計されています。AWS ドライバーは、DB インスタンスステータスをモニタリングし、インスタンストポロジを認識して新しいプライマリインスタンスを決定することを前提としています。このアプローチにより、スイッチオーバーとフェイルオーバーの時間が 1 桁秒に短縮されます (オープンソースドライバーの場合は数十秒)。

次の表は、各ドライバーでサポートされている機能を示しています。新しいサービス機能が導入されるにあたって、こうしたサービス機能を標準でサポートすることが AWS のドライバースイートの目標です。

機能	AWS JDBC ドライバー	AWS Python ドライバー	AWS MySQL 用 ODBC ドライバー
フェイルオーバーのサポート	はい	はい	はい
フェイルオーバーモニタリングの強化	はい	はい	はい
読み取り/書き込みの分割	はい	はい	いいえ
ドライバーメタデータ接続	はい	該当なし	該当なし
Telemetry	はい	はい	いいえ

機能	AWS JDBC ドライバー	AWS Python ドライバー	AWS MySQL 用 ODBC ドライバー
Secrets Manager	はい	はい	はい
IAM 認証	はい	はい	はい
フェデレーティッド ID (AD FS)	はい	はい	いいえ
フェデレーティッド ID (Okta)	はい	なし	いいえ
マルチ AZ DB クラスター	はい	はい	いいえ

AWS ドライバーの詳細については、使用している RDS for MariaDB、RDS for MySQL、または RDS for PostgreSQL DB インスタンスに対応する言語ドライバーを参照してください。

> **Note**
>
> RDS for MariaDB でサポートされている機能は、AWS Secrets Manager、AWS Identity and Access Management (IAM)、およびフェデレーティッド ID による認証のみです。

特定の DB エンジンを実行している DB インスタンスに接続する

特定の DB エンジンを実行している DB インスタンスへの接続方法については、DB エンジンの指示に従ってください。

- RDS for Db2
- RDS for MariaDB
- RDS for SQL Server
- RDS for MySQL
- RDS for Oracle
- RDS for PostgreSQL

RDS Proxy による接続の管理

Amazon RDS Proxy を使用して、RDS for MariaDB、RDS for Microsoft SQL Server、RDS for MySQL、および RDS for PostgreSQL DB インスタンスへの接続を管理することもできます。RDS Proxy を使用すると、アプリケーションでデータベース接続をプールおよび共有し、スケーラビリティを向上させることができます。詳細については、「Amazon RDS Proxy」を参照してください。

データベース認証オプション

Amazon RDS では、データベースユーザーを認証する次の方法がサポートされています。

- パスワード認証 – DB インスタンスがユーザーアカウントのすべての管理を行います。SQL 文でユーザーを作成し、パスワードを指定します。使用できる SQL 文は、DB エンジンによって異なります。
- AWS Identity and Access Management (IAM) データベース認証 – DB インスタンスに接続する際にパスワードを使用する必要はありません。代わりに、認証トークンを使用します。
- Kerberos 認証 – Kerberos および Microsoft Active Directory を使用して、データベースユーザーの外部認証を使用します。Kerberos は、ネットワーク経由でパスワードを送信する必要をなくすためにチケットと対称キー暗号化を使用するネットワーク認証プロトコルです。Kerberos は Active Directory に組み込まれており、データベースなどのネットワークリソースに対するユーザー認証を行えるように設計されています。

IAM データベース認証と Kerberos 認証は、特定の DB エンジンおよびバージョンでのみ使用できます。

詳細については、「Amazon RDS でのデータベース認証」を参照してください。

暗号化された接続

アプリケーションの Secure Socket Layer (SSL) または Transport Layer Security (TLS) を使用して、DB インスタンスへの接続を暗号化できます。各 DB エンジンには SSL/TLS を実装する独自のプロセスがあります。詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

オプショングループを使用する

DB エンジンによっては、データおよびデータベースの管理を容易にしたり、データベースのセキュリティを強化する追加の機能が用意されている場合があります。Amazon RDS は、オプショングループを使用してこれらの機能の有効化と設定を行います。オプショングループには、特定の Amazon RDS DB インスタンスに使用できる機能 (オプション) を指定できます。オプションの設定では、そのオプションの動作を指定できます。DB インスタンスをオプショングループに関連付けると、指定したオプションとそれらの設定がその DB インスタンスに対して有効になります。

Amazon RDS では、次のデータベースエンジンのオプションをサポートします。

データベースエンジン	関連資料
Db2	RDS for Db2 DB インスタンスのオプション
MariaDB	MariaDB データベースエンジンのオプション
Microsoft SQL Server	Microsoft SQL Server データベースエンジンのオプション
MySQL	MySQL DB インスタンスのオプション
Oracle	Oracle DB インスタンスへのオプションの追加
PostgreSQL	PostgreSQL はオプションとオプショングループを使用しません。PostgreSQL はエクステンションとモジュールを使用して追加機能を提供します。詳細については、「サポートされている PostgreSQL 拡張機能バージョン」を参照してください。

オプショングループの概要

Amazon RDS では、新しい DB インスタンスごとに空のデフォルトオプショングループが用意されます。このデフォルトのオプショングループを変更または削除することはできませんが、作成する新しいオプショングループは、デフォルトのオプショングループから設定を収得します。DB インスタンスにオプションを適用するには、以下を実行する必要があります。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. 1 つまたは複数のオプションをオプショングループに追加します。

3. オプショングループを DB インスタンスに関連付けます。

   オプショングループを DB インスタンスに関連付けるには、DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

DB インスタンスと DB スナップショットはいずれもオプショングループに関連付けることができます。場合によっては、DB スナップショットからの復元や、DB インスタンスに対するポイントインタイム復元を行うことができます。このような場合、DB スナップショットまたは DB インスタンスに関連付けられているオプショングループは、デフォルトで、復元された DB インスタンスに関連付けられています。復元された DB インスタンスに異なるオプショングループを関連付けることができます。ただし、新しいオプショングループには、元のオプショングループに含まれる永続オプションまたは固定オプションを含める必要があります。永続オプションおよび固定オプションについては後で説明します。

オプションでは、DB インスタンスで実行するために追加のメモリが必要です。そのため、DB インスタンスの現在の使用状況によっては、サイズのより大きいインスタンスの起動が必要になる場合があります。例えば、Oracle Enterprise Manager Database Control には約 300 MB の RAM が使用されます。サイズの小さい DB インスタンスに対してこのオプションを有効にした場合は、パフォーマンスの問題やメモリ不足のエラーが発生することがあります。

永続オプションと固定オプション

永続と固定の 2 種類のオプションをオプショングループに追加するときは、特別な考慮事項があります。

DB インスタンスがオプショングループに関連付けられている間は、永続オプションをオプショングループから削除することはできません。永続オプションの一例として、Microsoft SQL Server の透過的なデータ暗号化 (TDE) の TDE オプションがあります。オプショングループに関連付けられているすべての DB インスタンスの関連付けを解除してから、オプショングループから永続オプションを削除する必要があります。場合によっては、DB スナップショットから復元やポイントインタイム復元を行うことができます。DB スナップショットに関連付けられているオプショングループに永続オプションが含まれている場合、復元された DB インスタンスは、そのオプショングループにのみ関連付けることができます。

固定オプション (Oracle Advanced Security TDE の TDE オプションなど) をオプショングループから削除することはできません。固定オプションを使用している DB インスタンスのオプショングループは変更することができます。ただし、DB インスタンスに関連付けられているオプションに、同一の固定オプションが含まれている必要があります。場合によっては、DB スナップショットから復元

やポイントインタイム復元を行うことができます。DB スナップショットに関連付けられているオプショングループに固定オプションが含まれている場合、復元された DB インスタンスは、その固定オプションが含まれているオプショングループにのみ関連付けることができます。

Oracle DB インスタンスの場合、オプション Timezone または OLS (または両方) を持つ共有 DB スナップショットをコピーできます。そのためには、DB スナップショットをコピーするときにこれらのオプションを含むターゲットオプショングループを指定します。OLS オプションは、Oracle バージョン 12.2 以降を実行している Oracle DB インスタンスに対してのみ恒久的かつ永続的です。これらのオプションの詳細については、「Oracle のタイムゾーン」および「Oracle Label Security」を参照してください。

VPC に関する考慮事項

DB インスタンスに関連付けられているオプショングループは、DB インスタンスの VPC とリンクされます。つまり、別の VPC に DB インスタンスを復元しようとしても、そのインスタンスに割り当てられているオプショングループは使用できません。DB インスタンスを別の VPC に復元する場合は、以下のいずれかを行うことができます。

- デフォルトのオプショングループを DB インスタンスに割り当てます。
- VPC にリンクされているオプショングループを割り当てます。
- 新しいオプショングループを作成し、DB インスタンスに割り当てます。

Oracle TDE などの永続的または不変のオプションを使用する場合は、新しいオプショングループを作成する必要があります。別の VPC 内に DB インスタンスを復元する場合は、このオプショングループに永続的または不変のオプションを含める必要があります。

オプションの設定では、オプションの動作を制御します。例えば、Oracle Advanced Security オプション NATIVE_NETWORK_ENCRYPTION の設定を使用すると、DB インスタンスとの送受信のネットワークトラフィック用に暗号化アルゴリズムを指定できます。一部のオプションの設定は Amazon RDS に合わせて最適化されており、変更することはできません。

相互に排他的なオプション

一部のオプションは相互に排他的です。どちらか一方は使用できますが、両方を同時に使用することはできません。以下のオプションは相互に排他的です。

- Oracle Enterprise Manager Database Express「」および「Enterprise Manager Cloud Control 向け Oracle Management Agent」。

- Oracle ネイティブネットワーク暗号化、、および Oracle Secure Sockets Layer

オプショングループを作成する

デフォルトオプショングループからその設定を引き継いで新しいオプショングループを作成することはできます。次に 1 つまたは複数のオプションを新しいオプショングループに追加します。あるいは、既存のオプショングループがある場合は、そのオプショングループをすべてのオプションと共に新しいオプショングループにコピーできます。詳細については、「オプショングループをコピーする」を参照してください。

新しいオプショングループを作成すると、そこにオプションは含まれていません。オプショングループにオプションを追加する方法については、「オプショングループにオプションを追加する」を参照してください。必要なオプションを追加したら、DB インスタンスにオプショングループを関連付けることができます。これにより、オプションが DB インスタンスで利用可能になります。DB インスタンスにオプショングループを関連付ける方法については、オプショングループを使用する のデータベースエンジンのドキュメントを参照してください。

コンソール

オプショングループの作成方法の 1 つとして、AWS マネジメントコンソール を使用する方法があります。

新しいオプショングループをコンソールを使用して作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create option group] (オプショングループを作成) ウィンドウで次の操作を行います。
    a. [Name] (名前) には、AWS アカウント内で一意であるオプショングループの名前を入力します。名前には、英字、数字、ハイフンのみ使用できます。
    b. [Description] (説明) には、オプショングループの簡単な説明を入力します。説明は表示用に使用されます。
    c. [Engine] (エンジン) で、使用する DB エンジンを選択します。
    d. [Major Engine Version] (メジャーエンジンバージョン) で、必要な DB エンジンのメジャーバージョンを選択します。

5. 続行するには、[Yes, Create(はい、作成する)] を選択します。逆に、操作をキャンセルするには、[Cancel] を選択します。

AWS CLI

オプショングループを作成するには、以下の必須パラメータを指定して AWS CLI の create-option-group コマンドを使用します。

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

以下の例では、testoptiongroup という名前のオプショングループを作成し、Oracle Enterprise Edition DB エンジンに関連付けています。説明は引用符で囲みます。

Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name testoptiongroup \
    --engine-name oracle-ee \
    --major-engine-version 19 \
    --option-group-description "Test option group for Oracle Database 19c EE"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name testoptiongroup ^
    --engine-name oracle-ee ^-
    --major-engine-version 19 ^
    --option-group-description "Test option group for Oracle Database 19c EE"
```

RDS API

オプショングループを作成するには、Amazon RDS API の `CreateOptionGroup` オペレーションを呼び出します。以下のパラメータを含めます。

- `OptionGroupName`
- `EngineName`
- `MajorEngineVersion`
- `OptionGroupDescription`

オプショングループをコピーする

AWS CLI または Amazon RDS API を使用して、オプショングループをコピーできます。オプショングループをコピーすると便利です。例えば、既存のオプショングループがあり、そのカスタムパラメータと値のほとんどを新しいオプショングループに含める場合です。また、実稼働環境で使用しているオプショングループのコピーを作成し、コピーを変更してその他のオプションの設定をテストすることもできます。

> **Note**
>
> 現時点では、オプショングループを別の AWS リージョンにコピーすることはできません。

AWS CLI

オプショングループをコピーするには、AWS CLI の copy-option-group を使用します。次の必須パラメータを含めます。

- `--source-option-group-identifier`
- `--target-option-group-identifier`
- `--target-option-group-description`

Example

次の例では、オプショングループ `new-option-group` のローカルコピーである、`my-option-group` という名前のオプショングループを作成します。

Linux、macOS、Unix の場合:

```
aws rds copy-option-group \
    --source-option-group-identifier my-option-group \
    --target-option-group-identifier new-option-group \
    --target-option-group-description "My new option group"
```

Windows の場合:

```
aws rds copy-option-group ^
    --source-option-group-identifier my-option-group ^
    --target-option-group-identifier new-option-group ^
    --target-option-group-description "My new option group"
```

RDS API

オプショングループをコピーするには、Amazon RDS API の CopyOptionGroup オペレーションを呼び出します。以下の必須パラメータを含めます。

- SourceOptionGroupIdentifier
- TargetOptionGroupIdentifier
- TargetOptionGroupDescription

オプショングループにオプションを追加する

既存のオプショングループにオプションを追加できます。必要なオプションを追加したら、そのオプションが DB インスタンスで使用可能になるように、DB インスタンスにオプショングループを関連付けることができます。DB インスタンスにオプショングループを関連付ける方法については、オプショングループを使用するにリストされているご使用の DB エンジンのドキュメントを参照してください。

オプショングループの変更が直ちに適用されなければならないケースが 2 つあります。

- ポート値を追加または更新するオプションを追加した場合 (OEM など)。
- ポート値を含むオプションがあるオプショングループを追加、または削除した場合。

このような場合は、コンソールで [Apply Immediately (すぐに適用)] オプションを選択します。または、AWS CLI で --apply-immediately オプションを指定するか、Amazon RDS API で

ApplyImmediately パラメータを true に設定することもできます。ポート値を使用しないオプションはすぐに適用するか、DB インスタンスの次のメンテナンスウィンドウ中に適用できます。

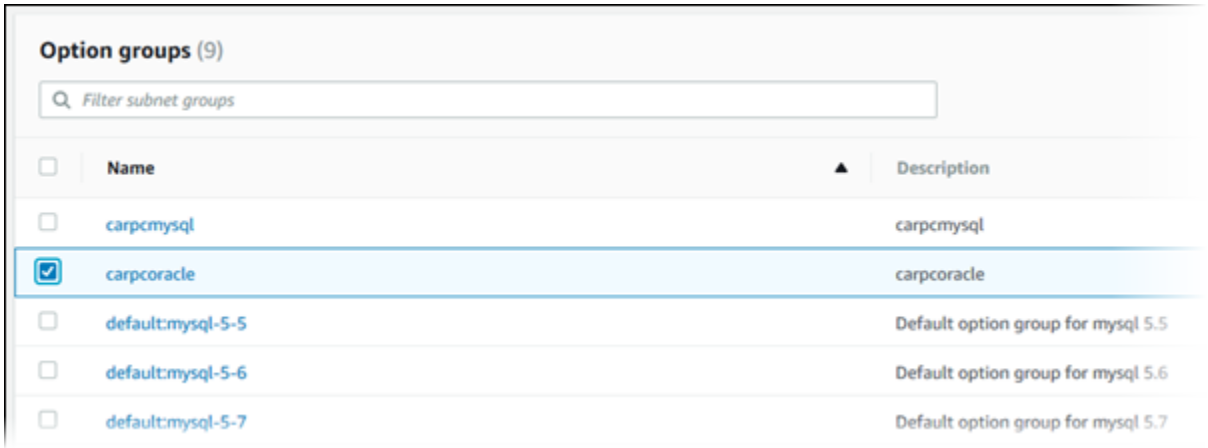
> **Note**
>
> セキュリティグループをオプショングループのオプションの値として指定する場合は、オプショングループを変更してセキュリティグループを管理します。DB インスタンスを変更してこのセキュリティグループを変更または削除することはできません。また、セキュリティグループは、AWS マネジメントコンソール、または AWS CLI コマンド describe-db-instances の出力の DB インスタンスの詳細には表示されません。

コンソール

AWS マネジメントコンソール を使用してオプショングループにオプションを追加できます。

オプションをコンソールを使用してオプショングループに追加するには

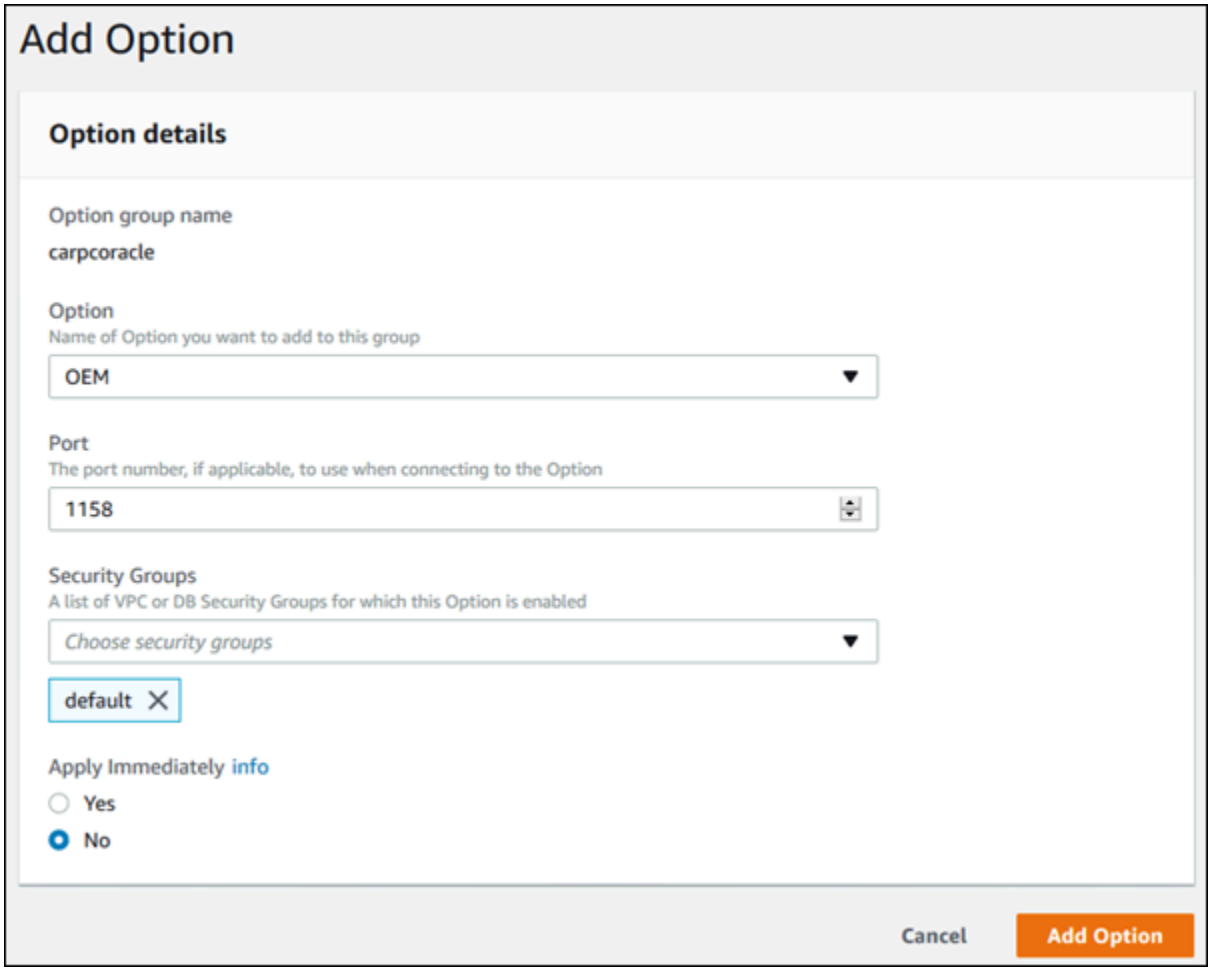
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 変更するオプショングループを選択し、[オプションの追加] を選択します。



4. [Add option(オプションの追加)] ウィンドウで、以下の操作を行います。

   a. 追加するオプションを選択します。選択したオプションによって、追加の値の指定が必要になる場合があります。例えば、OEM オプションを選択すると、ポート値を入力し、セキュリティグループを指定する必要もあります。

b. オプションを追加後すぐに、関連付けられているすべての DB インスタンスに対して有効にするには、[Apply Immediately] で [Yes] を選択します。[No] を選択した場合 (デフォルト)、オプションは次のメンテナンス時間中に、関連付けられている各 DB インスタンスに対して有効になります。



5. 設定が希望どおりになったら、[オプションの追加] を選択します。

AWS CLI

オプショングループにオプションを追加するには、AWS CLI add-option-to-option-group コマンドを追加するオプションで実行します。新しいオプションをすぐに、関連付けられているすべての DB インスタンスに対して有効にするには、`--apply-immediately` パラメータを指定します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスに対して有効になります。以下の必須パラメータを含めます。

- `--option-group-name`

Example

次の例では、America/Los_Angeles 設定の Timezone オプションを testoptiongroup という名前のオプショングループに追加し、すぐに有効にします。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
...{
       "OptionName": "Timezone",
       "OptionDescription": "Change time zone",
       "Persistent": true,
       "Permanent": false,
       "OptionSettings": [
           {
               "Name": "TIME_ZONE",
               "Value": "America/Los_Angeles",
               "DefaultValue": "UTC",
               "Description": "Specifies the timezone the user wants to change the
 system time to",
               "ApplyType": "DYNAMIC",
               "DataType": "STRING",
               "AllowedValues": "Africa/Cairo,...",
               "IsModifiable": true,
               "IsCollection": false
           }
       ],
       "DBSecurityGroupMemberships": [],
```

```
        "VpcSecurityGroupMemberships": []
    }...
```

Example

以下の例では、Oracle OEM オプションをオプショングループに追加します。また、カスタムポートと、Amazon EC2 VPC セキュリティグループのペアを使用できるように指定します。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2"
 \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options OptionName=OEM,Port=5500,VpcSecurityGroupMemberships="sg-test1,sg-test2"
 ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
OPTIONGROUP  False  oracle-ee  19 arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
 Test Option Group  testoptiongroup vpc-test
OPTIONS Oracle 12c EM Express   OEM     False   False   5500
VPCSECURITYGROUPMEMBERSHIPS     active  sg-test1
VPCSECURITYGROUPMEMBERSHIPS     active  sg-test2
```

Example

次の例では、Oracle オプション NATIVE_NETWORK_ENCRYPTION をオプショングループに追加し、オプションの設定を指定します。オプションの設定を指定しない場合、デフォルト値が使用されます。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
```

```
    --options '[{"OptionSettings":
[{"Name":"SQLNET.ENCRYPTION_SERVER","Value":"REQUIRED"},
{"Name":"SQLNET.ENCRYPTION_TYPES_SERVER","Value":"AES256,AES192,DES"}],"OptionName":"NATIVE_NET
 \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionSettings"=[{"Name"="SQLNET.ENCRYPTION_SERVER","Value"="REQUIRED"},
{"Name"="SQLNET.ENCRYPTION_TYPES_SERVER","Value"="AES256\,AES192\,DES"}],"OptionName"="NATIVE_N
 ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
...{
    "OptionName": "NATIVE_NETWORK_ENCRYPTION",
    "OptionDescription": "Native Network Encryption",
    "Persistent": false,
    "Permanent": false,
    "OptionSettings": [
        {
            "Name": "SQLNET.ENCRYPTION_TYPES_SERVER",
             "Value": "AES256,AES192,DES",
            "DefaultValue":
 "RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40",
            "Description": "Specifies list of encryption algorithms in order of
 intended use",
            "ApplyType": "STATIC",
            "DataType": "STRING",
            "AllowedValues":
 "RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40",
            "IsModifiable": true,
            "IsCollection": true
        },
        {
            "Name": "SQLNET.ENCRYPTION_SERVER",
            "Value": "REQUIRED",
            "DefaultValue": "REQUESTED",
            "Description": "Specifies the desired encryption behavior",
            "ApplyType": "STATIC",
```

```
            "DataType": "STRING",
            "AllowedValues": "ACCEPTED,REJECTED,REQUESTED,REQUIRED",
            "IsModifiable": true,
            "IsCollection": false
        },...
```

RDS API

Amazon RDS API を使用してオプショングループにオプションを追加するには、ModifyOptionGroup オペレーションを追加するオプションで呼び出します。新しいオプションをすぐに、関連付けられているすべての DB インスタンスに対して有効にするには、`ApplyImmediately` パラメータを指定して `true` に設定します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスに対して有効になります。以下の必須パラメータを含めます。

- `OptionGroupName`

オプショングループのオプションとオプション設定をリスト化する

オプショングループのオプションとそれらの設定をすべて一覧表示できます。

コンソール

AWS マネジメントコンソール を使用して、オプショングループのオプションとそれらの設定をすべて一覧表示できます。

オプショングループのオプションとその設定を一覧表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 詳細を表示するオプショングループの名前を選択します。オプショングループのオプションとその設定を一覧表示されます。

AWS CLI

オプショングループのオプションとオプションの設定を一覧表示するには、AWS CLI `describe-option-groups` コマンドを使用します。オプションとそれらの設定を表示するオプショングループの名前を指定します。オプショングループ名を指定しない場合、すべてのオプショングループが指定されます。

Example

以下の例では、すべてのオプショングループのオプションとそれらの設定を一覧表示しています。

```
aws rds describe-option-groups
```

Example

以下の例では、testoptiongroup という名前のオプショングループのオプションとそれらの設定を一覧表示しています。

```
aws rds describe-option-groups --option-group-name testoptiongroup
```

RDS API

オプショングループのオプションとオプションの設定を一覧表示するには、Amazon RDS API DescribeOptionGroups オペレーションを使用します。オプションとそれらの設定を表示するオプショングループの名前を指定します。オプショングループ名を指定しない場合、すべてのオプショングループが指定されます。

オプションの設定を変更する

変更可能なオプション設定を追加した後で、それらの設定はいつでも変更できます。オプショングループのオプションまたはそれらの設定を変更した場合、それらの変更は、そのオプショングループに関連付けられているすべての DB インスタンスに適用されます。さまざまなオプションで利用できる設定の詳細については、オプショングループを使用する のデータベースエンジンのドキュメントを参照してください。

オプショングループの変更が直ちに適用されなければならないケースが 2 つあります。

- ポート値を追加または更新するオプションを追加した場合 (OEM など)。
- ポート値を含むオプションがあるオプショングループを追加、または削除した場合。

このような場合は、コンソールで [Apply Immediately (すぐに適用)] オプションを選択します。または、--apply-immediately で AWS CLI をオプションを指定するか、RDS API で ApplyImmediately パラメータを true に設定することもできます。ポート値を使用しないオプションはすぐに適用するか、DB インスタンスの次のメンテナンスウィンドウ中に適用できます。

> **Note**
>
> セキュリティグループをオプショングループのオプションの値として指定する場合は、オプショングループを変更してセキュリティグループを管理します。DB インスタンスを変更してこのセキュリティグループを変更または削除することはできません。また、セキュリティグループは、AWS マネジメントコンソール、または AWS CLI コマンド `describe-db-instances` の出力の DB インスタンスの詳細には表示されません。

コンソール

AWS マネジメントコンソール を使用してオプション設定を変更できます。

コンソールを使用してオプション設定を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 変更するオプションを含むオプショングループを選択し、[Modify Option(オプションの変更)] を選択します。
4. [Modify Option(オプションの変更)] ウィンドウの [Installed Options(インストール済みのオプション)] から、設定を変更するオプションを選択します。必要な変更を行います。
5. オプションを追加後すぐに有効にするには、[Apply Immediately] で [Yes] を選択します。[No] を選択した場合 (デフォルト)、オプションは次のメンテナンス時間中に、関連付けられている各 DB インスタンスに対して有効になります。
6. 設定が希望どおりになったら、[Modify Option] を選択します。

AWS CLI

オプション設定を変更するには、変更するオプショングループおよびオプションを指定して、AWS CLI の `add-option-to-option-group` コマンドを使用します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスに対して有効になります。変更をすぐに、関連付けられているすべての DB インスタンスに適用するには、`--apply-immediately` パラメータを指定します。オプションの設定を変更するには、`--settings` 引数を使用します。

Example

以下の例では、testoptiongroup という名前のオプショングループの Oracle Enterprise Manager Database Control (OEM) 用のポートを変更し、その変更がすぐに適用されるように指定しています。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options OptionName=OEM,Port=5432,DBSecurityGroupMemberships=default ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
OPTIONGROUP   False  oracle-ee  19  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
   Test Option Group    testoptiongroup
OPTIONS Oracle 12c EM Express   OEM     False   False   5432
DBSECURITYGROUPMEMBERSHIPS   default  authorized
```

Example

次の例では、Oracle オプション NATIVE_NETWORK_ENCRYPTION を変更して、オプション設定を変更します。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name testoptiongroup \
    --options '[{"OptionSettings":
[{"Name":"SQLNET.ENCRYPTION_SERVER","Value":"REQUIRED"},
{"Name":"SQLNET.ENCRYPTION_TYPES_SERVER","Value":"AES256,AES192,DES,RC4_256"}],"OptionName":"NA
 \
```

```
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name testoptiongroup ^
    --options "OptionSettings"=[{"Name"="SQLNET.ENCRYPTION_SERVER","Value"="REQUIRED"},
{"Name"="SQLNET.ENCRYPTION_TYPES_SERVER","Value"="AES256\,AES192\,DES
\,RC4_256"}],"OptionName"="NATIVE_NETWORK_ENCRYPTION" ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
OPTIONGROUP   False  oracle-ee  19  arn:aws:rds:us-east-1:1234567890:og:testoptiongroup
   Test Option Group    testoptiongroup
OPTIONS Oracle Advanced Security - Native Network Encryption
 NATIVE_NETWORK_ENCRYPTION       False   False
OPTIONSETTINGS
 RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40 STATIC
 STRING
   RC4_256,AES256,AES192,3DES168,RC4_128,AES128,3DES112,RC4_56,DES,RC4_40,DES40
 Specifies list of encryption algorithms in order of intended use
   True     True    SQLNET.ENCRYPTION_TYPES_SERVER  AES256,AES192,DES,RC4_256
OPTIONSETTINGS  ACCEPTED,REJECTED,REQUESTED,REQUIRED    STATIC  STRING  REQUESTED
 Specifies the desired encryption behavior  False   True  SQLNET.ENCRYPTION_SERVER
 REQUIRED
OPTIONSETTINGS  SHA1,MD5   STATIC  STRING  SHA1,MD5    Specifies list of
 checksumming algorithms in order of intended use      True    True
 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  SHA1,MD5
OPTIONSETTINGS  ACCEPTED,REJECTED,REQUESTED,REQUIRED  STATIC  STRING
 REQUESTED     Specifies the desired data integrity behavior   False   True
 SQLNET.CRYPTO_CHECKSUM_SERVER  REQUESTED
```

RDS API

オプション設定を変更するには、変更するオプショングループおよびオプションを指定して、Amazon RDS API の ModifyOptionGroup コマンドを使用します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスに対して有効になります。変更をすぐに、関連付けられているすべての DB インスタンスに適用するには、ApplyImmediately パラメータを指定して、true に設定します。

オプショングループからオプションを削除する

オプションには、オプショングループから削除できるものとできないものがあります。永続オプションは、オプショングループに関連付けられているすべての DB インスタンスの関連付けが解除されるまで、そのオプショングループから削除することはできません。固定オプションは、オプショングループから削除することはできません。どのオプションが削除可能であるかについては、オプショングループを使用するにリストされているご使用のデータベースエンジンのドキュメントを参照してください。

オプショングループからすべてのオプションを削除しても、Amazon RDS ではオプショングループは削除されません。DB インスタンスは、その空のオプショングループに関連付けられたままになります。アクティブなオプションがなくなるだけです。逆に、DB インスタンスからすべてのオプションを削除するには、DB インスタンスをデフォルト (空) のオプショングループに関連付けます。

コンソール

AWS マネジメントコンソール を使用してオプショングループからオプションを削除できます。

オプションをコンソールを使用してオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 削除するオプションを含むオプショングループを選択し、[Delete Option(オプションの削除)] を選択します。
4. [Delete option(オプションの削除)] ウィンドウで、以下の操作を行います。
   - 削除するオプションのチェックボックスをオンにします。
   - 削除をすぐに有効にするには、[すぐに適用] で [はい] を選択します。[No] を選択した場合 (デフォルト)、オプションは次のメンテナンス時間中に、関連付けられている各 DB インスタンスで削除されます。

5. すべての設定が正しいことを確認したら、[Yes, Delete] を選択します。

AWS CLI

オプショングループからオプションを削除するには、削除するオプションで AWS CLI の [remove-option-from-option-group](#) コマンドを使用します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスから削除されます。変更をすぐに適用するには、`--apply-immediately` パラメータを指定します。

Example

以下の例では、`testoptiongroup` という名前のオプショングループから Oracle Enterprise Manager Database Control (OEM) オプションを削除し、その変更がすぐに適用されるように指定しています。

Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name testoptiongroup \
    --options OEM \
    --apply-immediately
```

Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name testoptiongroup ^
    --options OEM ^
    --apply-immediately
```

このコマンドでは、以下のような出力が生成されます。

```
OPTIONGROUP    testoptiongroup oracle-ee   19    Test option group
```

RDS API

オプショングループからオプションを削除するには、Amazon RDS API の ModifyOptionGroup アクションを使用します。デフォルトでは、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスから削除されます。変更をすぐに適用するには、ApplyImmediately パラメータを指定して、true に設定します。

以下のパラメータを含めます。

- OptionGroupName
- OptionsToRemove.OptionName

オプショングループを削除する

オプショングループを削除するには、次の条件を満たす必要があります。

- Amazon RDS リソースに関連付けられていないこと。オプショングループは、DB インスタンス、手動 DB スナップショット、または自動 DB スナップショットに関連付けることができます。
- デフォルトオプショングループでないこと。

DB インスタンスと DB スナップショットで使用されるオプショングループを特定するには、次の CLI コマンドを使用できます。

```
aws rds describe-db-instances \
    --query 'DBInstances[*].
[DBInstanceIdentifier,OptionGroupMemberships[].OptionGroupName]'
```

```
aws rds describe-db-snapshots | jq -r '.DBSnapshots[] | "\(.DBInstanceIdentifier),
\(.OptionGroupName)"' | sort | uniq
```

RDS リソースに関連付けられているオプショングループを削除しようとすると、以下のような工ラーが返ります。

```
An error occurred (InvalidOptionGroupStateFault) when calling the DeleteOptionGroup
 operation: The option group 'optionGroupName' cannot be deleted because it is in use.
```

オプショングループに関連付けられた Amazon RDS リソースを検索するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 詳細を表示するオプショングループを選択します。
4. [Associated Instances and Snapshots (関連付けられたインスタンスとスナップショット)] セクションで、関連付けられた Amazon RDS リソースを確認します。

DB インスタンスがオプショングループに関連付けられている場合は、別のオプショングループを使用するように DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

手動の DB スナップショットがオプショングループに関連付けられている場合は、別のオプショングループを使用するように DB スナップショットを変更します。これを行うには、AWS CLI `modify-db-snapshot` コマンドを使用します。

> **Note**
>
> 自動 DB スナップショットのオプショングループを変更することはできません。

コンソール

オプショングループの削除方法のひとつとして、AWS マネジメントコンソール を使用する方法があります。

コンソールを使用してオプショングループを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. オプショングループを選択します。
4. [Delete group (グループの削除)] を選択します。
5. 確認ページで、オプショングループの削除を終了するには [削除]、削除をキャンセルするには [キャンセル] を選択します。

AWS CLI

オプショングループを削除するには、以下の必須パラメータを指定して AWS CLI の `delete-option-group` コマンドを使用します。

- `--option-group-name`

Example

次の例では、`testoptiongroup` という名前のオプショングループを削除します。

Linux、macOS、Unix の場合:

```
aws rds delete-option-group \
    --option-group-name testoptiongroup
```

Windows の場合:

```
aws rds delete-option-group ^
    --option-group-name testoptiongroup
```

RDS API

オプショングループを削除するには、Amazon RDS API の `DeleteOptionGroup` オペレーションを呼び出します。次のパラメータを含めます。

- `OptionGroupName`

Amazon RDS のパラメータグループ

データベースパラメータは、データベースの設定方法を指定します。データベースパラメータでは、データベースに割り当てるメモリなどのリソースの量を指定できます。

DB インスタンスとマルチ AZ DB クラスターをパラメータグループに関連付けて、データベースの設定を管理します。Amazon RDS は、デフォルト設定を使用してパラメータグループを定義します。カスタマイズした設定を使用して独自のパラメータグループを定義できます。

> **Note**
>
> 一部の DB エンジンには、オプショングループのオプションとしてデータベースに追加できる追加機能があります。オプショングループの詳細については、オプショングループを使用するを参照してください。

トピック

- パラメータグループの概要
- Amazon RDS DB インスタンスの DB パラメータグループ
- マルチ AZ DB クラスターの DB クラスターパラメータグループを使用する
- DB パラメータグループを比較する
- DB パラメータの指定

パラメータグループの概要

DB パラメータグループは、1 つ以上の DB インスタンスに適用されるエンジン設定値のコンテナとして機能します。

DB クラスターパラメータグループは、マルチ AZ DB クラスターにのみ適用されます。マルチ AZ DB クラスターでは、DB クラスターパラメータグループの設定は、クラスター内のすべての DB インスタンスに適用されます。DB エンジンおよび DB エンジンバージョンのデフォルトの DB パラメータグループが、DB クラスター内の各 DB インスタンスに使用されます。

トピック

- デフォルトおよびカスタムパラメータグループ
- 静的および動的 DB インスタンスパラメータ
- 静的および動的 DB クラスターパラメータ

- 文字セットパラメータ
- サポートされるパラメータとパラメータ値

デフォルトおよびカスタムパラメータグループ

DB パラメータグループを指定せずに DB インスタンスを作成すると、DB インスタンスはデフォルトの DB パラメータグループを使用します。同様に、DB クラスターパラメータグループを指定せずにマルチ AZ DB クラスターを作成すると、DB クラスターではデフォルトの DB クラスターパラメータグループが使用されます。デフォルトの各パラメータグループには、エンジン、コンピューティングクラス、およびインスタンスの割り当てストレージに基づいた、データベースエンジンのデフォルトと Amazon RDS システムのデフォルトが含まれています。

デフォルトのパラメータグループのパラメータ設定は変更できません。代わりに、以下を実行できます。

1. 新しいパラメータグループを作成します。
2. 必要なパラメータの設定を変更します。パラメータグループ内のすべての DB エンジンパラメータが変更できるわけではありません。
3. DB インスタンスまたは DB クラスターを変更して、新しいパラメータグループを関連付けます。

   新しい DB パラメータグループを DB インスタンスに関連付けると、関連付けは直ちに有効になります。DB インスタンスの変更については、「」を参照してください。Amazon RDS DB インスタンスを変更するマルチ AZ DB クラスターの変更については、「Amazon RDS のマルチ AZ DB クラスターの変更」を参照してください。

   > **Note**
   >
   > カスタムパラメータグループを使用するように DB インスタンスを変更して、DB インスタンスを起動すると、RDS は起動プロセスの一環として DB インスタンスを自動的に再起動します。常時オプションやミラーリングオプションが有効になっている RDS for SQL Server のマルチ AZ インスタンスの場合、スタートアッププロセス後にインスタンスを再起動すると、フェイルオーバーが予想されます。

RDS は、DB インスタンスの再起動後にのみ、変更された静的パラメータと動的パラメータを新しく関連付けられたパラメータグループに適用します。ただし、DB インスタンスに関連付けた後に DB パラメータグループの動的パラメータを変更すると、これらの変更は再起動せずに直ちに適用さ

れます。DB パラメータグループの変更については、「Amazon RDS DB インスタンスを変更する」を参照してください。

DB パラメータグループ内のパラメータを更新すると、このパラメータグループに関連付けられたすべての DB インスタンスに変更が適用されます。同様に、マルチ AZ DB クラスターパラメータグループ内のパラメータを更新すると、この DB クラスターパラメータグループに関連付けられたすべての Aurora DB クラスターに変更が適用されます。

パラメータグループを最初から作成したくない場合は、AWS CLI copy-db-parameter-group コマンドまたは copy-db-cluster-parameter-group コマンドを使用して、既存のパラメータグループをコピーできます。場合によっては、パラメータグループをコピーすると便利なことがあります。例えば、既存の DB パラメータグループのカスタムパラメータと値のほとんどを新しい DB パラメータグループに含めたい場合です。

静的および動的 DB インスタンスパラメータ

DB インスタンスパラメータは静的または動的に使用することができます。違いについては下記をご覧ください。

- 静的パラメータを変更して DB パラメータグループを保存すると、パラメータの変更は関連付けられている DB インスタンスを手動で再起動した後に有効になります。静的パラメータの場合、コンソールは常に `ApplyMethod` として `pending-reboot` を使用します。
- 動的パラメータを変更すると、デフォルトでは、パラメータの変更は直ちに有効になり、再起動は不要です。AWS マネジメントコンソール を使用して DB インスタンスのパラメータ値を変更するときには、常に動的パラメータ向けの `ApplyMethod` として `immediate` を使用します。パラメータの変更を、関連付けられている DB インスタンスが再起動されるまで延期するには、AWS CLI または RDS API を使用します。パラメータを変更する場合は、`ApplyMethod` を `pending-reboot` に設定します。

> **Note**
>
> AWS CLI、あるいは RDS for SQL Server DB インスタンス上の RDS API で、動的パラメータとともに `pending-reboot` を使用するとエラーが発生します。RDS for SQL Server で `apply-immediately` を使用する

AWS CLI を使用してパラメータ値を変更する方法については、「modify-db-parameter-group」を参照してください。RDS API を使用してパラメータ値を変更する方法については、「ModifyDBParameterGroup」を参照してください。

DB インスタンスが、その関連付けられた DB パラメータグループに対する最新の変更を使用していない場合、コンソールに DB パラメータグループの [pending-reboot] (再起動の保留中) のステータスが表示されます。このステータスにより、次回のメンテナンスウィンドウで自動的に再起動されることはありません。パラメータの最新の変更を DB インスタンスに適用するには、DB インスタンスを手動で再起動します。

静的および動的 DB クラスターパラメータ

DB クラスターパラメータは静的または動的に使用することができます。違いについては下記をご覧ください。

- 静的パラメータを変更して DB クラスターのパラメータグループを保存する場合、パラメータの変更は、関連付けられている DB クラスターを手動で再起動した後に有効になります。静的パラメータの場合、コンソールは常に `ApplyMethod` として `pending-reboot` を使用します。
- 動的パラメータを変更すると、デフォルトでは、パラメータの変更は直ちに有効になり、再起動は不要です。AWS マネジメントコンソール を使用して DB クラスターのパラメータ値を変更するときには、常に動的パラメータ向けの `ApplyMethod` として `immediate` を使用します。パラメータの変更を、関連する DB クラスターが再起動されるまで延期するには、AWS CLI または RDS API を使用します。パラメータを変更する場合は、`ApplyMethod` を `pending-reboot` に設定します。

AWS CLI を使用してパラメータ値を変更する方法については、「modify-db-cluster-parameter-group」を参照してください。RDS API を使用してパラメータ値を変更する方法については、「ModifyDBClusterParameterGroup」を参照してください。

文字セットパラメータ

DB インスタンスまたはマルチ AZ DB クラスターを作成する前に、パラメータグループに含まれるデータベースの文字セットまたは照合に関するパラメータを設定します。また、その中にデータベースを作成する前にも行ってください。これにより、デフォルトのデータベースと新しいデータベースで、指定した文字セットと照合順序が使用されるようになります。文字セットまたは照合パラメータを変更した場合、パラメータの変更は既存のデータベースに適用されません。

一部の DB エンジンでは、`ALTER DATABASE` コマンドを使用して、既存のデータベースの文字セットまたは照合値を変更できます。次に例を示します。

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

データベースの文字セットまたは照合値の変更の詳細については、DB エンジンのドキュメントを参照してください。

サポートされるパラメータとパラメータ値

DB エンジンでサポートされているパラメータを決定するには、DB インスタンスまたは DB クラスターで使用される DB パラメータグループおよび DB クラスターパラメータグループのパラメータを表示します。詳細については、「Amazon RDS のDB パラメータグループのパラメータ値の表示」および「 DB クラスターパラメータグループのパラメータ値の表示」を参照してください。

多くの場合、表現、数式、関数を使用して、整数およびブール型パラメータを指定することができます。関数には、数学的な口グ式を含めることができます。ただし、すべてのパラメータが、パラメータ値の表現、数式、関数をサポートしているわけではありません。詳細については、「DB パラメータの指定」を参照してください。

パラメータグループに不適切な設定のパラメータがあると、パフォーマンスが低下したりシステムが不安定になったり、予期しない悪影響が生じることがあります。データベースパラメータの変更時には常に注意が必要です。パラメータグループの変更前にはデータをバックアップしてください。テスト用 DB インスタンスまたは DB クラスターでパラメータグループの設定の変更を試してから、本番稼働用 DB インスタンスまたは DB クラスターにそれらの変更を適用してください。

Amazon RDS DB インスタンスの DB パラメータグループ

DB インスタンスでは DB パラメータグループが使用されます。次のセクションでは、DB インスタンスパラメータグループの設定と管理について説明します。

トピック

- Amazon RDS での DB パラメータグループの作成
- Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける
- Amazon RDS の DB パラメータグループのパラメータの変更
- Amazon RDS の DB パラメータグループのパラメータをデフォルト値にリセットします
- Amazon RDS での DB パラメータグループのコピー
- Amazon RDS での DB パラメータグループの一覧表示
- Amazon RDS のDB パラメータグループのパラメータ値の表示
- Amazon RDS での DB パラメータグループの削除

Amazon RDS での DB パラメータグループの作成

新しい DB パラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使って作成できます。

DB パラメータグループ名には、次の制限事項が適用されます。

- 名前は、1 ~ 255 の英字、数字、ハイフンである必要があります。

  デフォルトのパラメータグループ名には、`default.mysql8.0` のようなピリオドを含めることができます。ただし、カスタムパラメータグループ名にはピリオドを含めることはできません。
- 1 字目は文字である必要があります。
- 名前の最後にハイフンを使用したり、ハイフンを 2 つ続けて使用したりすることはできません。

コンソール

DB パラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [パラメータグループの作成] を選択します。
4. [パラメータグループ名] に、新しい DB パラメータグループの名前を入力します。
5. [説明] に、新しい DB パラメータグループの説明を入力します。
6. [エンジンタイプ] で DB エンジンを選択します。
7. [パラメータグループファミリー] で、DB パラメータグループファミリーを選択します。
8. 該当する場合、[タイプ] で [DB パラメータグループ] を選択します。
9. [作成] を選択します。

AWS CLI

DB パラメータグループを作成するには、AWS CLI の `create-db-parameter-group` コマンドを使用します。以下の例では、MySQL バージョン 8.0 用に、mydbparametergroup という名前で、「My new parameter group」という説明の新しい DB パラメータグループを作成しています。

以下の必須パラメータを含めます。

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

使用可能なすべてのパラメータグループファミリーを一覧表示するには、次のコマンドを使用します。

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> 出力は重複が含まれます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL8.0 \
    --description "My new parameter group"
```

Windows の場合:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family MySQL8.0 ^
    --description "My new parameter group"
```

このコマンドでは、以下のような出力が生成されます。

```
DBPARAMETERGROUP  mydbparametergroup  mysql8.0  My new parameter group
```

RDS API

DB パラメータグループを作成するには、RDS API の CreateDBParameterGroup オペレーションを使用します。

以下の必須パラメータを含めます。

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける

カスタマイズした設定を使用して、独自の DB パラメータグループを作成できます。AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB パラメータグループを DB インスタンスに関連付けることができます。DB インスタンスを作成または変更するときに、これを行うことができます。

DB パラメータグループの作成については、Amazon RDS での DB パラメータグループの作成 を参照してください。DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。DB インスタンスの変更については、Amazon RDS DB インスタンスを変更するを参照してください。

> **Note**
>
> 新しい DB パラメータグループを DB インスタンスに関連付ける場合、変更された静的パラメータと動的パラメータは、DB インスタンスが再起動された後にのみ適用されます。ただし、DB インスタンスに関連付けた後に DB パラメータグループの動的パラメータを変更すると、これらの変更は再起動せずに直ちに適用されます。

コンソール

DB パラメータグループを DB インスタンスに関連付けるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] (変更) を選択します。Modify DB instance ページが表示されます。
4. DB パラメータグループの設定を変更します。
5. [Continue] を選択して、変更の概要を確認します。

6. (オプション) 変更をすぐに適用するには、[Apply immediately] (すぐに適用) を選択します。このオプションを選択すると、停止状態になる場合があります。詳細については、「スケジュール変更設定の使用」を参照してください。
7. 確認ページで、変更内容を確認します。正しい場合は、[Modify DB Instance (DB インスタンスを変更)] を選択して変更を保存します。

   または、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更をキャンセルします。

AWS CLI

DB パラメータグループを DB インスタンスに関連付けるには、以下のオプションを指定しながら AWS CLI の `modify-db-instance` コマンドを使用します。

- `--db-instance-identifier`
- `--db-parameter-group-name`

次の例では、mydbpg DB パラメータグループを database-1 DB インスタンスに関連付けます。`--apply-immediately` を使用すると変更はすぐに適用されます。`--no-apply-immediately` を使用して、次のメンテナンス時間中に変更を適用します。詳細については、 を参照してください。スケジュール変更設定の使用

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

RDS API

DB パラメータグループを DB インスタンスに関連付けるには、以下のパラメータを指定しながら RDS API の `ModifyDBInstance` オペレーションを使用します。

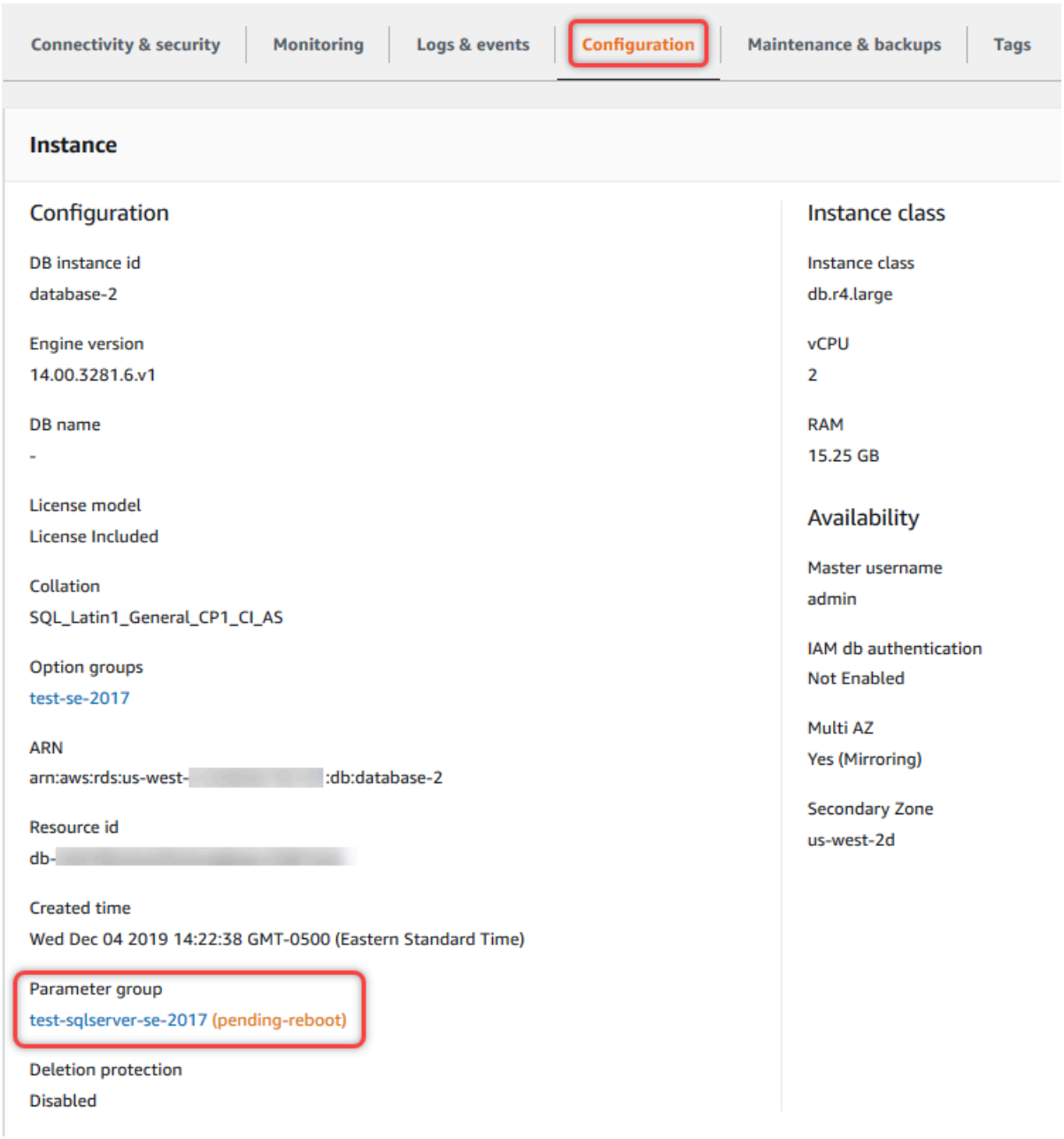
- `DBInstanceName`
- `DBParameterGroupName`

Amazon RDS の DB パラメータグループのパラメータの変更

ユーザー定義の DB パラメータグループのパラメータ値は変更できますが、デフォルトの DB パラメータグループのパラメータ値を変更することはできません。ユーザー定義の DB パラメータグループのパラメータの変更は、その DB パラメータグループに関連付けられたすべての DB インスタンスに適用されます。

パラメータには、動的パラメータと静的パラメータの 2 種類があります。動的パラメータへの変更は、再起動せずに直ちに DB インスタンスに適用されます。静的パラメータへの変更は、DB インスタンスの再起動後にのみ適用されます。

DB インスタンスに関連付けられている DB パラメータグループのステータスは、RDS コンソールの [設定] タブに表示されます。例えば、DB インスタンスがその関連付けられた DB パラメータグループの最新の変更を使用していない場合、RDS コンソールは、DB パラメータグループのステータスを [再起動の保留中] と表示します。パラメータの最新の変更を DB インスタンスに適用するには、DB インスタンスを手動で再起動します。



コンソール

DB パラメータグループ内のパラメータを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストで、変更するパラメータグループの名前を選択します。
4. [Parameter group actions (パラメータグループのアクション)] で、[編集] を選択します。

5. 変更するパラメータの値を変更します。ページ分割コントロールを使用してパラメータをスクロールできます。

   デフォルトパラメータグループの値を変更することはできません。

6. [Save changes] (変更を保存) をクリックします。

AWS CLI

DB パラメータグループを変更するには、次の必須オプションを指定して、AWS CLI の modify-db-parameter-group コマンドを使用します。

- `--db-parameter-group-name`
- `--parameters`

以下の例では、mydbparametergroup という名前の DB パラメータグループの `max_connections` と `max_allowed_packet` の値を変更しています。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

このコマンドでは、以下のような出力が生成されます。

```
DBPARAMETERGROUP  mydbparametergroup
```

RDS API

DB パラメータグループを変更するには、以下の必須パラメータを指定しながら RDS API の ModifyDBParameterGroup オペレーションを使用します。

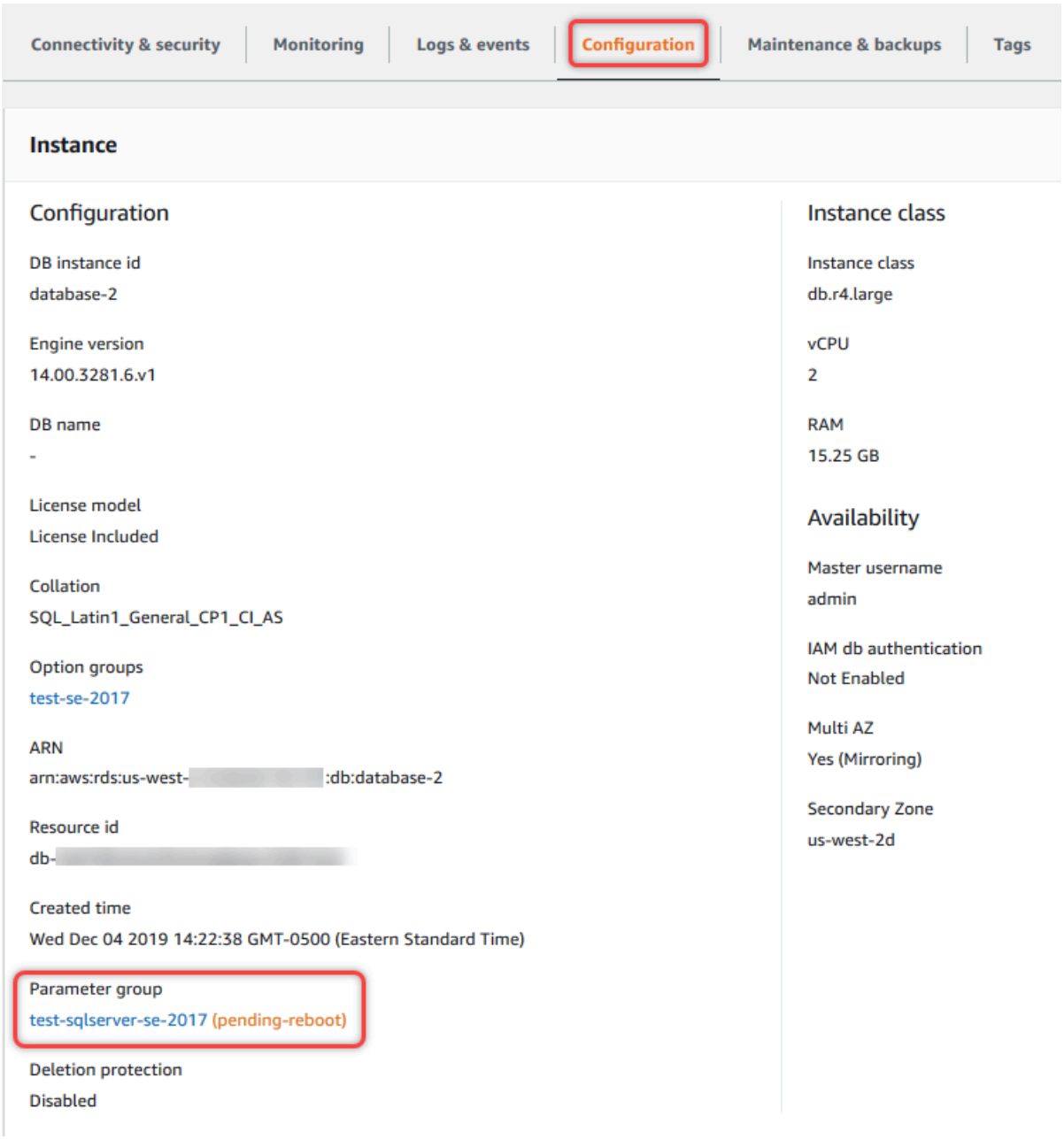
- DBParameterGroupName
- Parameters

Amazon RDS の DB パラメータグループのパラメータをデフォルト値にリセットします

顧客が作成した DB パラメータグループのパラメータ値をデフォルト値にリセットできます。ユーザー定義の DB パラメータグループのパラメータの変更は、その DB パラメータグループに関連付けられたすべての DB インスタンスに適用されます。

コンソールを使用すると、特定のパラメータをデフォルト値にリセットできます。ただし、DB パラメータグループのすべてのパラメータを一度にリセットするのは簡単ではありません。AWS CLI または RDS API を使用すると、特定のパラメータをデフォルト値にリセットできます。DB パラメータグループのすべてのパラメータを一度にリセットすることもできます。

一部のパラメータへの変更は、再起動せずに直ちに DB インスタンスに適用されます。他のパラメータの変更は、DB インスタンスの再起動後にのみ適用されます。DB インスタンスに関連付けられている DB パラメータグループのステータスは、RDS コンソールの [設定] タブに表示されます。例えば、DB インスタンスがその関連付けられた DB パラメータグループに対する最新の変更を使用していないとします。その場合、RDS コンソールは、DB パラメータグループのステータスを [pending-reboot] (再起動の保留中) と表示します。パラメータの最新の変更を DB インスタンスに適用するには、DB インスタンスを手動で再起動します。



> **Note**
>
> デフォルトの DB パラメータグループでは、パラメータは常にデフォルト値に設定されます。

コンソール

DB パラメータグループのパラメータをデフォルト値にリセットするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストからパラメータグループを選択します。
4. [Parameter group actions (パラメータグループのアクション)] で、[編集] を選択します。
5. デフォルト値にリセットするパラメータを選択します。ページ分割コントロールを使用してパラメータをスクロールできます。

   デフォルトのパラメータグループの値をリセットすることはできません。
6. リセットを選択し、パラメータをリセットを選択して確定します。

AWS CLI

DB パラメータグループの一部またはすべてのパラメータをリセットするには、AWS CLI`reset-db-parameter-group`コマンドに必須オプション`--db-parameter-group-name`を指定します。

DB パラメータグループのすべてのパラメータをリセットするには、`--reset-all-parameters`オプションを指定します。特定のパラメータをリセットするには、`--parameters`オプションを指定します。

次の例では、mydbparametergroup という名前の DB パラメータグループ内のすべてのパラメータをデフォルト値にリセットします。

Example

Linux、macOS、Unix の場合:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Windows の場合:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
```

```
    --reset-all-parameters
```

次の例では、mydbparametergroup という名前の DB パラメータグループのおよびmax_connectionsオプションmax_allowed_packetをデフォルト値にリセットします。

Example

Linux、macOS、Unix の場合:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

このコマンドでは、以下のような出力が生成されます。

```
DBParameterGroupName  mydbparametergroup
```

RDS API

DB パラメータグループ内のパラメータをデフォルト値にリセットするには、RDS API ResetDBParameterGroup コマンドに必須パラメータ DBParameterGroupName を指定します。

DB パラメータグループのすべてのパラメータをリセットするには、ResetAllParametersパラメータをtrueに設定します。特定のパラメータをリセットするには、Parametersパラメータを指定します。

Amazon RDS での DB パラメータグループのコピー

作成したカスタム DB パラメータグループをコピーできます。パラメータグループのコピーは便利なソリューションです。例としては、作成済みの DB パラメータグループがあり、そのほとんどのカスタムパラメータと値を、新しい DB パラメータグループに含める場合です。AWS マネジメントコンソール を使用して、DB パラメータグループをコピーできます。AWS CLI copy-db-parameter-group コマンド、または RDS API CopyDBParameterGroup オペレーションも使用できます。

DB パラメータグループをコピーした後で、その DB パラメータグループをデフォルトのパラメータグループとして使用する最初の DB インスタンスを作成するまで、最低 5 分待ちます。これにより、パラメータグループを使用する前に、Amazon RDS はコピーアクションが完全に終了できます。これは、DB インスタンスのデフォルトのデータベースを作成する際に不可欠となるパラメータで特に重要です。例としては、`character_set_database` パラメータで定義するデフォルトのデータベースの文字セットが挙げられます。DB パラメータグループが作成されたことを確認するには、Amazon RDS コンソールの [パラメータグループ] オプション、または describe-db-parameters コマンドを使用します。

> **Note**
>
> デフォルトのパラメータグループをコピーすることはできません。ただし、デフォルトのパラメータグループに基づく新しいパラメータグループを作成できます。
> DB パラメータグループを別の AWS アカウント または AWS リージョン にコピーすることはできません。

コンソール

DB パラメータグループをコピーするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストで、コピーするカスタムパラメータグループを選択します。
4. [Parameter group actions (パラメータグループのアクション)] で、[コピー] を選択します。
5. [新規の DB パラメータグループの識別子] に、新しいパラメータグループの名前を入力します。
6. [説明] に、新しいパラメータグループの説明を入力します。
7. [Copy (コピー)] を選択します。

AWS CLI

DB パラメータグループをコピーするには、次の必須オプションを指定しながら AWS CLI の `copy-db-parameter-group` コマンドを使用します。

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`

- `--target-db-parameter-group-description`

次の例は、DB パラメータグループ mygroup2 のコピーである mygroup1 という名前の新しい DB パラメータグループを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Windows の場合:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

RDS API

DB パラメータグループをコピーするには、以下の必須パラメータを指定して、RDS API の CopyDBParameterGroup オペレーションを使用します。

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Amazon RDS での DB パラメータグループの一覧表示

AWS アカウント用に作成した DB パラメータグループを一覧表示できます。

> **Note**
>
> デフォルトのパラメータグループは、特定の DB エンジンとバージョンの DB インスタンスを作成するときに、デフォルトのパラメータテンプレートから自動的に作成されます。これらのデフォルトのパラメータグループには、優先されるパラメータ設定が含まれています。

これを変更することはできません。カスタムパラメータグループを作成する場合、パラメータ設定を変更できます。

コンソール

AWS アカウントのすべての DB パラメータグループを一覧表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB パラメータグループがリストに表示されます。

AWS CLI

AWS アカウントのすべての DB パラメータグループを一覧表示するには、AWS CLI の `describe-db-parameter-groups` コマンドを使用します。

Example

以下の例では、AWS アカウントに使用できるすべての DB パラメータグループを一覧表示しています。

```
aws rds describe-db-parameter-groups
```

このコマンドでは次のようなレスポンスが返されます。

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

次の例は、mydbparamgroup1 パラメータグループを表しています。

Linux、macOS、Unix の場合:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Windows の場合:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

このコマンドでは次のようなレスポンスが返されます。

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

RDS API

AWS アカウントのすべての DB パラメータグループを一覧表示するには、RDS API の DescribeDBParameterGroups オペレーションを使用します。

Amazon RDS のDB パラメータグループのパラメータ値の表示

DB パラメータグループのすべてのパラメータとそれらの値のリストを取得できます。

コンソール

DB パラメータグループのパラメータ値を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB パラメータグループがリストに表示されます。
3. パラメータを一覧表示するパラメータグループの名前を選択します。

AWS CLI

DB パラメータグループのパラメータ値を表示するには、以下の必須パラメータを指定して、AWS CLI の describe-db-parameters コマンドを使用します。

- `--db-parameter-group-name`

Example

以下の例では、mydbparametergroup という名前の DB パラメータグループのパラメータとその値を一覧表示しています。

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

このコマンドでは次のようなレスポンスが返されます。

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                   engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768            system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string
 static      false
```

RDS API

DB パラメータグループのパラメータ値を表示するには、以下の必須パラメータを指定して、RDS API の `DescribeDBParameters` コマンドを使用します。

- `DBParameterGroupName`

Amazon RDS での DB パラメータグループの削除

DB パラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して削除できます。パラメータグループは、DB インスタンスに関連付けられていない場合にのみ削除できます。

コンソール

DB パラメータグループを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB パラメータグループがリストに表示されます。
3. 削除するパラメータグループの名前を選択します。

4. [アクション] を選択し、[削除] を選択します。
5. パラメータグループ名を確認して、[削除] を選択します。

AWS CLI

DB パラメータグループを削除するには、AWS CLI の delete-db-parameter-group コマンドを使用して、次の必須パラメータを指定します。

- `--db-parameter-group-name`

Example

次の例では、mydbparametergroup という名前の DB パラメータグループを削除します。

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

RDS API

DB パラメータグループを削除するには、RDS API DeleteDBParameterGroup コマンドを使用して、次の必須パラメータを指定します。

- `DBParameterGroupName`

マルチ AZ DB クラスターの DB クラスターパラメータグループを使用する

マルチ AZ DB クラスターでは、DB クラスターパラメータグループが使用されます。次のセクションでは、DB クラスターパラメータグループの設定と管理について説明します。

トピック

- DB クラスターパラメータグループの作成
- DB クラスターパラメータグループのパラメータの変更
- DB クラスターパラメータグループのパラメータのリセット
- DB クラスターパラメータグループのコピー
- DB クラスターパラメータグループの一覧表示
- DB クラスターパラメータグループのパラメータ値の表示
- DB クラスターパラメータグループの削除

DB クラスターパラメータグループの作成

新しい DB クラスターパラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使って作成できます。

DB クラスターパラメータグループの作成後、その DB クラスターパラメータグループを使用する最初の DB クラスターが作成されるまで、5 分以上かかります。これにより、Amazon RDS は新しい DB クラスターによって使用される前に、パラメータグループを完全に作成することができます。DB クラスターパラメータグループが作成されたことを確認するには、Amazon RDS コンソールの [パラメータグループ] ページまたは describe-db-cluster-parameters コマンドを使用できます。

DB クラスターパラメータグループ名には、次の制限事項が適用されます。

- 名前は、1 ~ 255 の英字、数字、ハイフンである必要があります。

  デフォルトのパラメータグループ名には、`default.mysql5.7` のようなピリオドを含めることができます。ただし、カスタムパラメータグループ名にはピリオドを含めることはできません。
- 1 字目は文字である必要があります。
- 名前の最後にハイフンを使用したり、ハイフンを 2 つ続けて使用したりすることはできません。

コンソール

DB クラスターパラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [パラメータグループの作成] を選択します。
4. [パラメータグループ名] に、新しい DB クラスターパラメータグループの名前を入力します。
5. [説明] に、新しい DB クラスターパラメータグループの説明を入力します。
6. [エンジンタイプ] でデータベースエンジンを選択します。
7. [パラメータグループファミリー] で、DB パラメータグループファミリーを選択します。
8. [作成] を選択します。

AWS CLI

DB クラスターのパラメータグループを作成するには、AWS CLI の create-db-cluster-parameter-group コマンドを使用します。

次の例では、RDS for MySQL バージョン 8.0 用に、mydbclusterparametergroup という名前で、「My new cluster parameter group」(新しいクラスターパラメータグループ) という説明の DB クラスターパラメータグループを作成しています。

以下の必須パラメータを含めます。

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

使用可能なすべてのパラメータグループファミリーを一覧表示するには、次のコマンドを使用します。

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> 出力は重複が含まれます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family mysql8.0 \
    --description "My new cluster parameter group"
```

Windows の場合:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
```

```
    --db-parameter-group-family mysql8.0 ^
    --description "My new cluster parameter group"
```

このコマンドでは、以下のような出力が生成されます。

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "mysql8.0",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup2"
    }
}
```

RDS API

DB クラスターのパラメータグループを作成するには、RDS API の CreateDBClusterParameterGroup アクションを使用します。

以下の必須パラメータを含めます。

- DBClusterParameterGroupName
- DBParameterGroupFamily
- Description

DB クラスターパラメータグループのパラメータの変更

ユーザーが作成した DB クラスターパラメータグループのパラメータ値は変更できます。デフォルト DB クラスターパラメータグループのパラメータ値は変更できません。ユーザー定義の DB クラスターパラメータグループのパラメータの変更は、その DB クラスターパラメータグループに関連付けられたすべての DB クラスターに適用されます。

コンソール

DB クラスターパラメータグループを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストで、変更するパラメータグループを選択します。
4. [Parameter group actions (パラメータグループのアクション)] で、[編集] を選択します。
5. 変更するパラメータの値を変更します。ページ分割コントロールを使用してパラメータをスクロールできます。

   デフォルトパラメータグループの値を変更することはできません。
6. [変更を保存] をクリックします。
7. クラスター内のを再起動して、変更を適用します。

   クラスターを再起動しない場合、フェイルオーバーオペレーションに通常よりも時間がかかることがあります。

AWS CLI

DB クラスターパラメータグループを変更するには、以下の必須パラメータを指定しながら AWS CLI の [modify-db-cluster-parameter-group](#) コマンドを使用します。

- `--db-cluster-parameter-group-name`
- `--parameters`

以下の例では、mydbclusterparametergroup という名前の DB クラスターパラメータグループの `server_audit_logging` と `server_audit_logs_upload` の値を変更しています。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-cluster-parameter-group ^
```

```
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

このコマンドでは、以下のような出力が生成されます。

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

RDS API

DB クラスターのパラメータグループを変更するには、以下の必須パラメータを指定しながら RDS API の ModifyDBClusterParameterGroup コマンドを使用します。

- DBClusterParameterGroupName
- Parameters

DB クラスターパラメータグループのパラメータのリセット

顧客が作成した DB クラスターパラメータグループの、デフォルト値のパラメータはリセットできます。ユーザー定義の DB クラスターパラメータグループのパラメータの変更は、その DB クラスターパラメータグループに関連付けられたすべての DB クラスターに適用されます。

> **Note**
>
> デフォルトの DB クラスターパラメータグループでは、パラメータは常にデフォルト値に設定されます。

コンソール

DB クラスターパラメータグループのパラメータをデフォルト値にリセットするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストからパラメータグループを選択します。

4. [Parameter group actions (パラメータグループのアクション)] で、[編集] を選択します。
5. デフォルト値にリセットするパラメータを選択します。ページ分割コントロールを使用してパラメータをスクロールできます。

   デフォルトのパラメータグループの値をリセットすることはできません。
6. リセットを選択し、パラメータをリセットを選択して確定します。
7. 。

AWS CLI

DB クラスターのパラメータグループにおいて、パラメータをデフォルト値にリセットするには、以下の `--db-cluster-parameter-group-name` オプション (必須) を指定しながら AWS CLI の `reset-db-cluster-parameter-group` コマンドを使用します。

DB クラスターパラメータグループのパラメータをすべてリセットするには、`--reset-all-parameters` オプションを指定します。特定のパラメータをリセットするには、`--parameters`オプションを指定します。

次の例では、mydbparametergroup という名前の DB パラメータグループ内のすべてのパラメータをデフォルト値にリセットします。

Example

Linux、macOS、Unix の場合:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Windows の場合:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

以下の例では、mydbclusterparametergroup という名前の DB クラスターパラメータグループにある `server_audit_logging` と `server_audit_logs_upload` をデフォルト値にリセットしています。

Example

Linux、macOS、Unix の場合:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

このコマンドでは、以下のような出力が生成されます。

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

RDS API

DB クラスターパラメータグループのパラメータをデフォルト値にリセットするには、以下の必須パラメータを指定して、RDS API ResetDBClusterParameterGroupコマンドを使用します。DBClusterParameterGroupName

DB クラスターパラメータグループのパラメータをすべてリセットするには、ResetAllParameters パラメータを true に設定します。特定のパラメータをリセットするには、Parametersパラメータを指定します。

DB クラスターパラメータグループのコピー

作成したカスタム DB クラスターパラメータグループをコピーできます。パラメータグループのコピーは、作成済みの DB クラスターパラメータグループがあり、そのグループの多くのカスタムパラメータと値を新しい DB クラスターパラメータグループに含める必要がある場合に便利な方法です。DB クラスターパラメータグループをコピーするには、AWS CLI copy-db-cluster-parameter-group コマンド、または RDS API CopyDBClusterParameterGroup オペレーションを使用できます。

DB クラスターパラメータグループをコピーした後で、この DB クラスターパラメータグループを使用する DB クラスターを作成するまで、5 分以上かかります。これにより、Amazon RDS は新しい DB クラスターによって使用される前に、パラメータグループを完全にコピーすることができます。DB クラスターパラメータグループが作成されたことを確認するには、Amazon RDS コンソールの [Parameter groups] (パラメータグループ) ページまたは describe-db-cluster-parameters コマンドを使用できます。

> **Note**
>
> デフォルトのパラメータグループをコピーすることはできません。ただし、デフォルトのパラメータグループに基づく新しいパラメータグループを作成できます。
> DB クラスターパラメータグループを別の AWS アカウント または AWS リージョン にコピーすることはできません。

コンソール

DB クラスターパラメータグループをコピーするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストで、コピーするカスタムパラメータグループを選択します。
4. [Parameter group actions (パラメータグループのアクション)] で、[コピー] を選択します。
5. [新規の DB パラメータグループの識別子] に、新しいパラメータグループの名前を入力します。
6. [説明] に、新しいパラメータグループの説明を入力します。
7. [Copy (コピー)] を選択します。

AWS CLI

DB クラスターのパラメータグループをコピーするには、以下の必須パラメータを指定しながら AWS CLI の `copy-db-cluster-parameter-group` コマンドを使用します。

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

次の例は、DB クラスターパラメータグループ mygroup2 のコピーである mygroup1 という名前の新しい DB クラスターパラメータグループを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Windows の場合:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

RDS API

DB クラスターパラメータグループをコピーするには、以下の必須パラメータを指定して、RDS API の CopyDBClusterParameterGroup オペレーションを使用します。

- SourceDBClusterParameterGroupIdentifier
- TargetDBClusterParameterGroupIdentifier
- TargetDBClusterParameterGroupDescription

DB クラスターパラメータグループの一覧表示

AWS アカウント用に作成した DB クラスターパラメータグループを一覧表示できます。

> **Note**
>
> デフォルトのパラメータグループは、特定の DB エンジンとバージョンの DB クラスターを作成するときに、デフォルトのパラメータテンプレートから自動的に作成されます。これらのデフォルトのパラメータグループには、優先されるパラメータ設定が含まれています。これを変更することはできません。カスタムパラメータグループを作成する場合、パラメータ設定を変更できます。

コンソール

AWS アカウントのすべての DB クラスターパラメータグループを一覧表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB クラスターパラメータグループは、[Type] (タイプ) が [DB cluster parameter group] (DB クラスターパラメータグループ) のリストに表示されます。

AWS CLI

AWS アカウントにある、すべての DB クラスターのパラメータグループを一覧表示するには、AWS CLI の `describe-db-cluster-parameter-groups` コマンドを使用します。

Example

以下の例では、AWS アカウントに使用できるすべての DB クラスターパラメータグループを一覧表示しています。

```
aws rds describe-db-cluster-parameter-groups
```

次の例は、mydbclusterparametergroup パラメータグループを表しています。

Linux、macOS、Unix の場合:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Windows の場合:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

このコマンドでは次のようなレスポンスが返されます。

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup2",
```

```
            "DBParameterGroupFamily": "mysql8.0",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

RDS API

AWS アカウントにある、すべての DB クラスターのパラメータグループを一覧表示するには、RDS API の `DescribeDBClusterParameterGroups` アクションを使用します。

DB クラスターパラメータグループのパラメータ値の表示

DB クラスターパラメータグループのすべてのパラメータとそれらの値のリストを取得できます。

コンソール

DB クラスターパラメータグループのパラメータ値を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB クラスターパラメータグループは、[タイプ] が [DB クラスターパラメータグループ] のリストに表示されます。
3. パラメータを一覧表示する DB クラスターパラメータグループの名前を選択します。

AWS CLI

DB クラスターのパラメータグループについて、そのパラメータ値を表示するには、以下の必須パラメータを指定しながら AWS CLI の `describe-db-cluster-parameters` コマンドを使用します。

- `--db-cluster-parameter-group-name`

Example

以下の例では、JSON 形式の mydbclusterparametergroup という名前の DB クラスターパラメータグループのパラメータとその値を一覧表示しています。

このコマンドでは次のようなレスポンスが返されます。

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
        {
            "ParameterName": "activate_all_roles_on_login",
            "ParameterValue": "0",
            "Description": "Automatically set all granted roles as active after the
 user has authenticated successfully.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
...
```

RDS API

DB クラスターパラメータグループのパラメータ値を表示するには、以下の必須パラメータを指定して、RDS API の DescribeDBClusterParameters コマンドを使用します。

- DBClusterParameterGroupName

場合によっては、パラメータに指定できる値が表示されないことがあります。これらは常にソースがデータベースエンジンのデフォルトであるパラメータです。

これらのパラメータの値を表示するには、次の SQL ステートメントを実行します。

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

DB クラスターパラメータグループの削除

DB クラスターパラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して削除できます。DB クラスターパラメータグループは、DB クラスターに関連付けられていない場合にのみ削除できます。

コンソール

パラメータグループを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   パラメータグループがリストに表示されます。
3. 削除する DB クラスターパラメータグループの名前を選択します。

4. [アクション] を選択し、[削除] を選択します。
5. パラメータグループ名を確認して、[削除] を選択します。

AWS CLI

DB クラスターパラメータグループを削除するには、AWS CLI の [delete-db-cluster-parameter-group](#) コマンドを使用して、次の必須パラメータを指定します。

- `--db-parameter-group-name`

Example

次の例では、mydbparametergroup という名前の DB クラスターパラメータグループを削除します。

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

RDS API

DB クラスターパラメータグループを削除するには、RDS API の [DeleteDBClusterParameterGroup](#) コマンドを使用して、次の必須パラメータを指定します。

- `DBParameterGroupName`

DB パラメータグループを比較する

AWS マネジメントコンソール を使用して、2 つの DB パラメータグループ間の差異を表示できます。

指定されたパラメータグループは、両方とも DB パラメータグループであるか、両方とも DB クラスターパラメータグループである必要があります。DB エンジンとバージョンが同じであっても当てはまります。例えば、`aurora-mysql8.0` (Aurora MySQL バージョン 3) DB パラメータグループと `aurora-mysql8.0` DB クラスターパラメータグループを比較することはできません。

Aurora MySQL と RDS for MySQL DB のパラメータグループは、バージョンが異なっていても比較できますが、Aurora PostgreSQL と RDS for PostgreSQL DB のパラメータグループは比較できません。

2 つの DB パラメータグループを比較するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. リストで、比較する 2 つのパラメータグループを選択します。

> **Note**
>
> デフォルトのパラメータグループとカスタムパラメータグループを比較するには、まず、[デフォルト] タブでデフォルトパラメータグループを選択し、次に [カスタム] タブでカスタムパラメータグループを選択します。

4. [アクション] で、[比較] を選択します。

DB パラメータの指定

DB パラメータの種類には、次のものがあります。

- 整数
- ブール値
- 文字列
- Long
- ダブル
- タイムスタンプ
- 他の定義されたデータ型オブジェクト
- 整数、ブール値、文字列、長整数、倍精度、タイムスタンプ、オブジェクト型の値の配列

表現、数式、関数を使用して、整数およびブール型パラメータを指定することもできます。

Oracle エンジンの場合は、`DBInstanceClassHugePagesDefault` 式変数を使用して、ブール型 DB パラメータを指定します。「DB パラメータ式の変数」を参照してください。

PostgreSQL エンジンでは、式を使用してブール型 DB パラメータを指定できます。「ブール型 DB パラメータ式」を参照してください。

目次

- DB パラメータ式
  - DB パラメータ式の変数
  - DB パラメータ式の演算子
- DB パラメータ関数
- ブール型 DB パラメータ式
- DB パラメータログ式
- DB パラメータ値の例

DB パラメータ式

DB パラメータ式は整数値に解決される式あるいはブール値です。式は中かっこ {} で囲みます。式は、DB パラメータ値、または DB パラメータ関数の引数として使用できます。

構文

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

DB パラメータ式の変数

各式の変数は整数あるいはブール値を返します。変数名では大文字と小文字が区別されます。

AllocatedStorage

データボリュームのサイズ (バイト単位) を表す整数を返します。

DBInstanceClassHugePagesDefault

ブール値を返します。現在のところ、これは Oracle エンジンのみでサポートされています。

詳細については、「サポートされている RDS for Oracle インスタンスで HugePages をオンにする」を参照してください。

DBInstanceClassMemory

データベースプロセスに対し、使用可能なメモリのバイト数を整数で返します。この数値は、DB インスタンスクラスの合計メモリ量から始めて内部的に計算されます。この数から、インスタン

スを管理する RDS プロセスとオペレーティングシステム用に予約されているメモリ量を減算します。したがって、この数値は、「DB インスタンスクラス」のインスタンスクラステーブルに示されているメモリ値よりも、常にやや低くなります。正確な値は、複数の要因の組み合わせによって異なります。これらには、インスタンスクラスと DB エンジンが含まれ、この値が適用されるのが RDS インスタンスなのか、Aurora クラスターに含まれているインスタンスなのかによっても異なります。

DBInstanceVCPU

Amazon RDS がインスタンスの管理に使用する仮想中央演算装置 (vCPU) の数を表す整数を返します。

EndPointPort

DB インスタンスに接続するときに使用されるポートを表す整数を返します。

TrueIfReplica

DB インスタンスがリードレプリカである場合は 1、リードレプリカでない場合は 0 が返されます。これは MySQL の`read_only` パラメータのデフォルト値です。

DB パラメータ式の演算子

DB パラメータ式では、2 つ演算子 (除算と乗算) がサポートされています。

除算演算子: /

被除数を除数で割り、整数の商を返します。商の小数部分は四捨五入されず切り捨てられます。

構文

```
dividend / divisor
```

被除数と除数の引数は整数式である必要があります。

乗算演算子: *

式を乗算し、式の積を返します。式の小数部分は四捨五入されず切り捨てられます。

構文

```
expression * expression
```

両方の式は整数である必要があります。

DB パラメータ関数

DB パラメータ関数の引数は、整数または数式で指定します。各関数には 1 つ以上の引数が必要です。複数の引数をカンマ区切りのリストで指定します。リストには、argument1,,argument3 など、空のメンバーを使用することはできません。関数名では大文字と小文字は区別されません。

IF

引数を返します。

現在のところ、これは Oracle エンジンに対してのみサポートされ、最初の引数が {DBInstanceClassHugePagesDefault} の場合のみサポートされています。詳細については、「サポートされている RDS for Oracle インスタンスで HugePages をオンにする」を参照してください。

構文

```
IF(argument1, argument2, argument3)
```

最初の引数が true と評価する場合に、2 番目の引数を返します。それ以外の場合には、3 番目の引数を返します。

GREATEST

整数またはパラメータ式のリストから最大値を返します。

構文

```
GREATEST(argument1, argument2,...argumentn)
```

整数を返します。

LEAST

整数またはパラメータ式のリストから最小値を返します。

構文

```
LEAST(argument1, argument2,...argumentn)
```

整数を返します。

SUM

指定した整数またはパラメータ式の値を加算します。

構文

```
SUM(argument1, argument2,...argumentn)
```

整数を返します。

ブール型 DB パラメータ式

ブール型 DB パラメータ式は、ブール値 1 または 0 に解決されます。式は引用符で囲みます。

> **Note**
> ブール型 DB パラメータ式は、PostgreSQL エンジンでのみサポートされています。

構文

```
"expression operator expression"
```

どちらの式も整数に解決する必要があります。式には、次のものがあります。

- 整数定数
- DB パラメータ式
- DB パラメータ関数
- DB パラメータ変数

ブール型 DB パラメータ式は、次の不等号演算子をサポートしています。

より大きい演算子: >

構文

```
"expression > expression"
```

より小さい演算子: <

構文

```
"expression < expression"
```

より大きいか等しい演算子: >=、=>

構文

```
"expression >= expression"
"expression => expression"
```

より小さいか等しい演算子: <=、=<

構文

```
"expression <= expression"
"expression =< expression"
```

Exampleブール型 DB パラメータ式の使用

次のブール型 DB パラメータ式の例では、パラメータ式の結果を整数と比較します。これは、PostgreSQL DB インスタンスのブール型 DB パラメータ wal_compression を変更するためです。パラメータ式は、vCPU の数と値 2 を比較します。仮想 CPU の数が 2 より大きい場合、wal_compression DB パラメータが true に設定されます。

```
aws rds modify-db-parameter-group --db-parameter-group-name group-name \
--parameters "ParameterName=wal_compression,ParameterValue=\"{DBInstanceVCPU} > 2\" "
```

DB パラメータログ式

整数 DB パラメータ値をログ式に設定できます。式は中かっこ {} で囲みます。次に例を示します。

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

log 関数はログベース 2 を表します。この例では、DBInstanceClassMemory 数式変数も使用しています。「DB パラメータ式の変数」を参照してください。

> **Note**
>
> 現在、MySQLinnodb_log_file_sizeパラメータを整数以外の値で指定することはできません。

DB パラメータ値の例

これらの例は、DB パラメータの値に対して数式、関数、および式を使用していることを示しています。

> **Warning**
>
> DB パラメータグループのパラメータを不適切に設定すると、意図しない悪影響が生じる可能性があります。これには、パフォーマンスの低下やシステムの不安定化が含まれます。データベースパラメータの変更時には注意が必要です。DB パラメータグループの変更前にはデータをバックアップしてください。パラメータグループの変更は、テスト DB インスタンス (ポイントインタイム復元を使用して作成) で試してから、本番稼働用 DB インスタンスに適用してください。

Example DB パラメータ関数 GREATEST の使用

Oracle プロセスパラメータで GREATEST 関数を指定できます。これを使用して、ユーザープロセスの数を 80、 または DBInstanceClassMemory を 9,868,951 で割った値の大きい方に設定します。

```
GREATEST({DBInstanceClassMemory/9868951},80)
```

Example DB パラメータ関数 LEAST の使用

MySQL LEAST パラメータ値で max_binlog_cache_size 関数を指定できます。これを使用して、トランザクションが MySQL インスタンスで使用できる最大キャッシュサイズを 1MB または DBInstanceClass/256 のいずれか小さい方に設定します。

```
LEAST({DBInstanceClassMemory/256},10485760)
```

の Amazon RDS DB インスタンス設定を使用した Amazon ElastiCache キャッシュの作成

ElastiCache は、フルマネージドのインメモリキャッシュサービスであり、マイクロ秒単位の読み取り/書き込みレイテンシーを提供し、柔軟なリアルタイムユースケースをサポートします。ElastiCache は、アプリケーションとデータベースのパフォーマンスを向上させるのに役立ちます。ElastiCache は、ゲームのリーダーボード、ストリーミング、データ分析など、データの耐久性を必要としないユースケースのプライマリデータストアとして使用できます。ElastiCache は、分散コンピューティング環境のデプロイと管理に関連する複雑さを排除するのに役立ちます。詳細については、Memcached の場合は、「Common ElastiCache Use Cases and How ElastiCache Can Help」を、Redis OSS の場合は、「Common ElastiCache Use Cases and How ElastiCache Can Help」を参照してください。ElastiCache キャッシュを作成する際に、Amazon RDS コンソールを使用できます。

Amazon ElastiCache は 2 つの形式で運用できます。サーバーレスキャッシュで始めるか、独自のキャッシュクラスターを設計するかを選択できます。独自のキャッシュクラスターを設計する場合、ElastiCache は Redis OSS エンジンと Memcached エンジンの両方で動作します。使用するエンジンが不明な場合は、「Comparing Memcached and Redis OSS」を参照してください。Amazon ElastiCache の詳細については、「Amazon ElastiCache ユーザーガイド」を参照してください。

トピック

- RDS DB インスタンス設定による ElastiCache キャッシュ作成の概要
- 既存の RDS DB インスタンスの設定で ElastiCache キャッシュを作成する

RDS DB インスタンス設定による ElastiCache キャッシュ作成の概要

新規作成または既存の RDS DB インスタンスと同じ設定を使用して、Amazon RDS から ElastiCache キャッシュを作成できます。

ElastiCache キャッシュを DB インスタンスに関連付けるためのいくつかのユースケース:

- RDS で ElastiCache を使用すると、RDS だけで実行するよりもコストを節約し、パフォーマンスを向上させることができます。

  例えば、RDS for MySQL で ElastiCache を使用すると、RDS for MySQLS のみを使用する場合と比較して、コストを最大 55% 節約し、読み取りパフォーマンスを最大 80 倍高速化できます。

- ElastiCache キャッシュは、データの耐久性を必要としないプライマリデータストアとしてアプリケーションで使用できます。Redis OSS または Memcached を使用するアプリケーションでは ElastiCache を利用できます。ほとんど変更はありません。

RDS から ElastiCache キャッシュを作成すると、ElastiCache キャッシュは、関連する RDS DB インスタンスから以下の設定を継承します。

- ElastiCache 接続設定
- ElastiCache セキュリティ設定

要件に応じて、キャッシュ設定を指定できます。

アプリケーションで ElastiCache を設定する

アプリケーションは ElastiCache キャッシュを利用するように設定する必要があります。また、要件に応じてキャッシュ戦略を使用するようにアプリケーションを設定することで、キャッシュのパフォーマンスを最適化して改善できます。

- ElastiCache キャッシュにアクセスして開始するには、「Getting started with ElastiCache (Redis OSS)」と「Getting started with ElastiCache (Memcached)」を参照してください。
- キャッシュ戦略の詳細については、Memcached の場合は「Caching strategies and best practices」、Redis OSS の場合は「Caching strategies and best practices」を参照してください。
- ElastiCache (Redis OSS) クラスターの高可用性の詳細については、「High availability using replication groups」を参照してください。
- バックアップストレージ、リージョン内またはリージョン間のデータ転送、または使用に関連するコストが発生する可能性があります。AWS Outposts価格設定の詳細については、「Amazon ElastiCache 料金表」を参照してください。

既存の RDS DB インスタンスの設定で ElastiCache キャッシュを作成する

DB インスタンスから継承された設定を使用して、 RDS DB インスタンスの ElastiCache キャッシュを作成できます。

DB インスタンスの設定で ElastiCache キャッシュを作成する

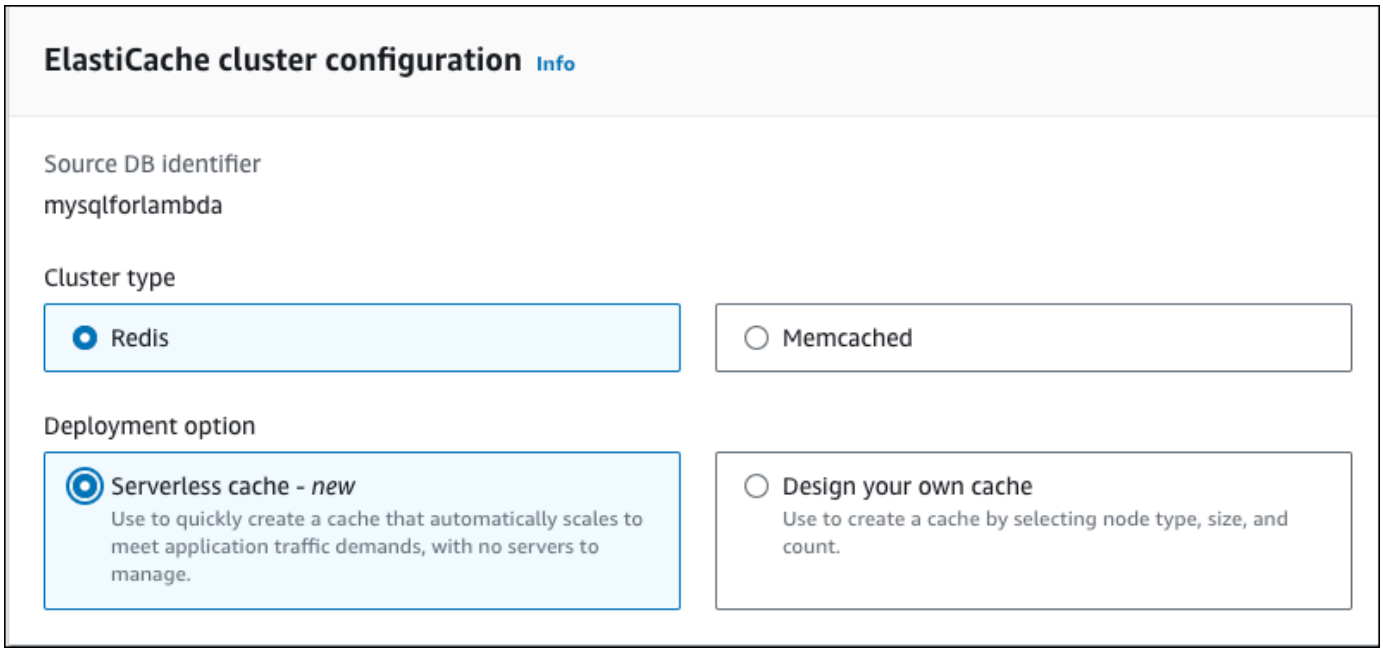
1. DB インスタンスを作成するには、「Amazon RDS DB インスタンスの作成」の手順に従います。

2. RDS DB インスタンスを作成すると、コンソールに [推奨されるアドオン] ウィンドウが表示されます。[DB 設定を使用して RDS から ElastiCache クラスターを作成する] を選択します。

   既存のデータベースでは、[データベース] ページで、必要な DB インスタンスを選択します。[アクション] ドロップダウンメニューで [ElastiCache クラスターの作成] を選択して、既存の RDS DB インスタンスと同じ設定の ElastiCache キャッシュを RDS に作成します。

   [ElastiCache 設定セクション] で、[ソース DB 識別子] に ElastiCache キャッシュが設定を継承する DB インスタンスが表示されます。

3. Redis OSS または Memcached クラスターを作成するかどうかを選択します。詳細については、「Comparing Memcached and Redis OSS」を参照してください。



4. その後、[サーバーレスキャッシュ] を作成するか、[独自のキャッシュを設計] するかを選択します。詳細については、「デプロイオプションの選択」を参照してください。

   [サーバーレスキャッシュ] を選択した場合：

   a. [キャッシュ設定] で、[名前] と [説明] に値を入力します。

   b. [デフォルト設定の表示] ではデフォルト設定のままにして、キャッシュと DB インスタンス間の接続を確立します。

   c. [デフォルト設定をカスタマイズ] を選択して、デフォルトの設定を編集することもできます。[ElastiCache の接続設定]、[ElastiCache のセキュリティ設定]、[使用量の上限] を選択します。

5. [独自のキャッシュ を設計] を選択した場合：

a. [Redis OSS クラスター] を選択した場合は、クラスターモードを [有効] にするか [無効] にするかを選択します。詳細については、「Replication: Redis OSS (Cluster Mode Disabled) vs. Redis OSS (Cluster Mode Enabled)」を参照してください。

b. [名前]、[説明]、[エンジンバージョン] の値を入力します。

   [エンジンバージョン] については、推奨されるデフォルト値は最新のエンジンバージョンです。要件に最も合う ElastiCache キャッシュの [エンジンバージョン] を選択することもできます。

c. [ノードタイプ] オプションでノードタイプを選択します。詳細については、「ノードの管理」を参照してください。

   [クラスターモード] を [有効] に設定して Redis OSS クラスターを作成する場合、[シャード数] オプションにシャード (パーティション/ノードグループ) の数を入力します。

   [レプリカ数] に各シャードのレプリカ数を入力します。

   > **Note**
   >
   > 選択したノードタイプ、シャード数、レプリカ数はすべて、キャッシュのパフォーマンスとリソースコストに影響します。これらの設定がデータベースのニーズに合っていることを確認してください。料金情報については、「Amazon ElastiCache 料金表」を参照してください。

d. [ElastiCache の接続設定] と [ElastiCache のセキュリティ設定] を選択します。デフォルト設定のままにすることも、要件に応じて設定をカスタマイズすることもできます。

6. ElastiCache キャッシュのデフォルト設定と継承された設定を確認します。一部の設定は、作成後に変更できません。

   > **Note**
   >
   > RDS は、60 分の最小ウィンドウ要件を満たすように ElastiCache キャッシュのバックアップウィンドウを調整する場合があります。ソースデータベースのバックアップウィンドウは変わりません。

7. 準備が完了したら、[ElastiCache キャッシュの作成] を選択します。

コンソールに、ElastiCache キャッシュの作成に関する確認バナーが表示されます。バナーにあるリンクから ElastiCache コンソールにアクセスすると、キャッシュの詳細が表示されます。ElastiCache コンソールには、新しく作成された ElastiCache キャッシュが表示されます。

AWS Database Migration Service を使用した、データベースの Amazon RDS への自動移行

RDS コンソールを使用して、EC2、オンプレミス、またはその他のクラウドプロバイダーインスタンスから RDS にデータベースを移行できます。これには AWS Database Migration Service (AWS DMS) が使用されます。詳細については、「AWS Database Migration Service ユーザーガイド」の「AWS Database Migration Service とは」を参照してください。

移行を開始するには、データの移行先となる同等の RDS DB インスタンスを作成する必要があります。ターゲットデータベースを作成したら、そのデータベースにソースをインポートできます。ソースデータベースが 1TiB 未満の場合、この移行アクションにより、データを RDS に移行するために必要となる時間とリソースが節約されます。

概要

RDS コンソールで、EC2、オンプレミス、またはその他のクラウドプロバイダーのデータベースを同等の RDS データベースに移行できます。コンソールからの移行を有効にするには、RDS データベースを作成する必要があります。

> **Note**
>
> データベースを同等にするには、同じデータベースエンジンおよび互換性のあるエンジンバージョンが必要です。

このアプローチは、次のデータベースエンジンに使用できます。

- MySQL
- MariaDB
- PostgreSQL

この移行プロセスには、以下のステップが含まれます。

- RDS で同等のデータベースを作成します。次に、ソースとターゲットの間に適切なネットワークを設定します。同じリージョン、アカウント、VPC の EC2 インスタンスの場合、ネットワークのセットアップをスキップできます。詳細については、「AWS Database Migration Service ユーザーガイド」の「ネットワークのセットアップ」を参照してください。データベースを作成する手順については、「Amazon RDS DB インスタンスの作成」を参照してください。

- データベースのレプリケーションのタイプを選択します。
  - フルロード移行 – RDS は、ソースデータベース全体をターゲットデータベースにコピーし、必要に応じてターゲットに新しいテーブルを作成します。

    > **Note**
    >
    > このオプションにはダウンタイムが必要です。移行プロセス中、ターゲット RDS データベースはアプリケーションで使用できなくなります。

  - フルロードと変更データキャプチャ (CDC) 移行 – フルロード移行と同様に、このオプションでは、RDS はソースデータベース全体をターゲットデータベースにコピーします。ただし、フルロード移行の後は、RDS はソースでキャプチャされた変更をターゲットデータベースに適用します。変更データキャプチャでは、データベースエンジンのネイティブ API を使用してデータベースログへの変更を収集します。

    > **Note**
    >
    > このオプションにはダウンタイムが必要です。移行プロセス中、ターゲット RDS データベースはアプリケーションで使用できなくなります。

  - 変更データキャプチャ (CDC) — このオプションを使用すると、移行中もターゲットデータベースを使用できます。RDS は、ソースデータベースの進行中の変更をターゲットデータベースに移行します。
- RDS は、移行を容易にするために必要なリソースを作成します。RDS が必要なリソースを作成すると、作成されたリソースについて通知が行われ、データ転送を開始できます。

  移行の完了に必要な時間は、レプリケーションのタイプとソースデータベースのサイズによって異なります。

前提条件

- ネットワークの設定 (同じリージョン、アカウント、VPC 内の EC2 の場合、スキップできます)
- ソースデータベースとターゲットデータベースの設定
  - MySQL および MariaDB

    ソースデータベースの次の基本的な前提条件に従ってください。
    - MySQL または MariaDB をソースとして使用する

ターゲットデータベースの次の基本的な前提条件に従ってください。

- MySQL または MariaDB をターゲットとして使用する

さらに、MySQL ソースデータベースから移行する場合、RDS アカウントにレプリケーション管理者ロールが必要です。また、そのロールに適切な権限を適用する必要があります。

- PostgreSQL

ソースデータベースの次の前提条件に従ってください。

- PostgreSQL のソースとしての使用

ターゲットデータベースの次の前提条件に従ってください。

- ターゲットとしての PostgreSQL の使用

> **Note**
>
> 一部の AWS DMS トランザクションでは、DMS エンジンで再度そのパラメータが使用されるまでの間、しばらくアイドル状態になります。PostgreSQL バージョン 9.6 以降で `idle_in_transaction_session_timeout` パラメータを使用すると、アイドル状態のトランザクションでタイムアウトやエラーが生じる場合があります。

制限事項

自動移行プロセスには、次の制限が適用されます。

- ソースデータベースの移行を開始するには、ターゲットデータベースのステータスが[使用可能] である必要があります。
- ソースデータベースは、次のデータベースにのみ移行できます。
  - それはクラスターのメンバーではありません
  - ここに記載されている、サポートされているバージョンの MySQL、PostgreSQL、または MariaDB を使用している
- DMS の制限事項

> **Note**
>
> 基盤となる AWS DMS ツールは、特定の移行シナリオの選択ルールをサポートしていますが、データベースの RDS への自動移行機能はサポートしていません。

同種移行のための IAM リソースの作成

RDS は AWS DMS を使用してデータを移行します。データベースにアクセスしてデータを移行するために、AWS DMS は同種データ移行用のサーバーレス環境を作成します。AWS DMS はこの環境で、VPC ピアリング、ルートテーブル、セキュリティグループ、その他の AWS リソースにアクセスする必要があります。また、AWS DMS は、各データ移行のログ、メトリクス、進行状況を Amazon CloudWatch に保存します。データ移行プロジェクトを作成するには、AWS DMS がこのようなサービスにアクセスする必要があります。

また、AWS DMS は、一連のユーザー認証情報を表すシークレットにアクセスして、ソースおよびターゲット接続のためにデータベース接続を認証する必要もあります。

> **Note**
>
> [EC2 インスタンスからデータを移行] アクションを使用すると、RDS コンソールを使ってこれらの IAM リソースを生成できます。コンソールで生成された IAM リソースを使用する場合は、この手順を省略してください。

このプロセスでは、次の IAM リソースが必要です。

トピック

- 同種データ移行のための IAM ポリシーの作成
- 同種データ移行のための IAM ロールの作成
- シークレットアクセスポリシーとロールの作成
- Amazon VPC を管理するために AWS DMS の IAM ロールを作成する

同種データ移行のための IAM ポリシーの作成

このステップでは、AWS DMS に Amazon EC2 と CloudWatch のリソースへのアクセスを付与する IAM ポリシーを作成します。その後、IAM ロールを作成して、このポリシーをアタッチします。

データ移行のために IAM ポリシーを作成するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール https://console.aws.amazon.com/iam/ を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーの作成] ページで、[JSON] タブをクリックします。
5. 次の JSON コードをエディタに貼り付けます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. [次へ: タグ]、[次へ: 確認] の順に選択します。
7. [名前*] には **HomogeneousDataMigrationsPolicy** と入力して、[ポリシーを作成] をクリックします。

同種データ移行のための IAM ロールの作成

このステップでは、AWS Secrets Manager、Amazon EC2、CloudWatch へのアクセスを付与する IAM ロールを作成します。

データ移行のための IAM ロールを作成するには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。
3. [ロールの作成] を選択してください。

4. [信頼されたエンティティを選択] ページの [信頼されたエンティティタイプ] では、AWS[サービス] を選択します。[その他の AWS サービスのユースケース] では、[DMS] を選択します。
5. [DMS] チェックボックスをオンにして、[次へ] をクリックします。
6. [許可を追加] ページで、作成した [HomogeneousDataMigrationsPolicy] を選択します。[次へ] を選択します。
7. [名前、確認、および作成] ページで、[ロール名] に **HomogeneousDataMigrationsRole** と入力して、[ロールの作成] をクリックします。
8. [ロール] ページの [ロール名] には、**HomogeneousDataMigrationsRole** と入力します。[HomogeneousDataMigrationsRole] を選択します。
9. [HomogeneousDataMigrationsRole] ページで、[信頼関係] タブをクリックします。[Edit trust policy] (信頼ポリシーを編集) を選択します。
10. [信頼ポリシーを編集] ページで、次の JSON コードをエディタに貼り付けて、既存のテキストを置き換えます。

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

    上記の例の *your_region* は独自の AWS リージョン ユーザー名に置き換えます。

上記のリソースベースのポリシーは、カスタマーマネージドの HomogeneousDataMigrationsPolicy ポリシーに沿ってタスクを実行するアクセス許可を AWS DMS サービスプリンシパルに付与します。

11. [ポリシーの更新] を選択します。

シークレットアクセスポリシーとロールの作成

以下の手順に従って、シークレットアクセスポリシーとロールを作成し、DMS がソースデータベースとターゲットデータベースのユーザー認証情報にアクセスできるようにします。

シークレットアクセスポリシーとロールを作成して、Amazon RDS の AWS Secrets Manager へのアクセスを許可して適切なシークレットにアクセスできるようにするには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/iam の AWS Identity and Access Management (IAM) マネジメントコンソールを開きます。
2. [Policies] (ポリシー) を選択し、次に[Create Policy] (ポリシー作成) を選択します
3. [JSON] を選択し、次のポリシーを入力して、シークレットへのアクセスと復号化を有効にします。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName-ABCDEF"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        }
    ]
}
```

secret_arn は、必要に応じて SecretsManagerSecretId より取得できるシークレットの ARN で、*kms_key_arn* は、シークレットを暗号化するために使用する AWS KMS キーの ARN です。次に例を示します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> AWS Secrets Manager で作成されたデフォルトの暗号化キーを使用する場合、*kms_key_arn* に AWS KMS 許可を指定する必要はありません。ポリシーで両方のシークレットへのアクセスを許可する場合は、もう 1 つの *[secret_arn]* に追加の JSON リソースオブジェクトを指定するだけです。

4. フレンドリ名とオプションの説明を使用して、ポリシーを確認して作成します。
5. [Roles] (ロール)、[Create role] (ロールの作成) の順に選択します。
6. 信頼されたエンティティの種類に、[AWS service] (サービス) を選択します。
7. 信頼されたサービスのリストから [DMS] を選択し、[Next: Permissions] (次へ: アクセス許可) を選択します。
8. ステップ 4 で作成したポリシーを検索しアタッチし、任意のタグを追加しながらロールを確認します。この時点で、信頼されたエンティティとして Amazon RDS リージョンサービスプリンシパルを使用するように、ロールの信頼関係を編集します。このプリンシパルの形式は以下のとおりです。

```
dms.region-name.amazonaws.com
```

ここで、*region-name* は 例えば us-east-1 などといったリージョンの名前です。したがって、このリージョンの Amazon RDS リージョンサービスプリンシパルが続きます。

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Amazon VPC を管理するために AWS DMS の IAM ロールを作成する

リソースの VPC 設定を管理するには、AWS DMS の IAM ロールを作成する必要があります。移行を成功させるためには、このロールを用意する必要があります。

データベース移行のために **dms-vpc-role** を作成する

<result>

これにより、移行の VPC 設定を管理するための DMS のロールが作成されます。
</result>

1. AWS マネジメントコンソール にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. [信頼されたエンティティを選択] オプションで [AWS のサービス] オプションを選択します。

   [ユースケース] で、[DMS] を選択します。
4. [許可を追加] ステップで、[AmazonDMSVPCManagementRole] を選択して [次へ] 選択します。

5. [名前、確認、および作成] ページで、[ロール名] を [dms-vpc-role] に設定し、[ロールを作成] を選択します。

EC2 データベースのデータ移行の設定

EC2 ソースデータベースからデータの移行を開始するには、同等の RDS データベースを作成する必要があります。データベースを作成する手順については、「Amazon RDS DB インスタンスの作成」を参照してください。

ターゲットデータベースを作成したら、次のステップを使用してデータ移行を設定します。

データ移行プロジェクトを設定する

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. [アクション] ドロップダウンを選択し、[EC2 データベースからデータを移行] オプションを選択します。サポート対象のターゲットデータベースについては、「制限事項」を参照してください。
3. [ソース EC2 データベースの選択] セクションで、次の操作を行います。

    1. [エンジンのタイプ] をチェックし、ソースデータベースと同じであることを確認します。

        また、エンジンのバージョンに互換性があるかどうかを確認します。
    2. [EC2 インスタンス] で、ソースデータベースが存在する EC2 インスタンスを選択します。
    3. [ポート] で、ソースデータベースがトラフィックを許可するポートを入力します。
    4. 既存のシークレットがない場合は、[シークレット] で [新しいシークレットを作成して使用] を選択します。ソースデータベースの [ユーザー名] と [パスワード] を入力します。また、シークレットの暗号化に使用する KMS キーも選択します。

        既存のシークレットがある場合は、[既存のシークレットを使用] を選択し、ドロップダウンからシークレットを選択します。
    5. 既存の IAM ロールがある場合は、[シークレットの IAM ロール] で、[既存の IAM ロールを使用] を選択し、前のステップのシークレット ID へのアクセスが許可された IAM ロールをドロップダウンから選択します。

        既存の IAM ロールがない場合は、[新しい IAM ロールを作成して使用] を選択します。[IAM ロール名] に、ロールの新しい名前を入力します。下のリンクでこのロールに関連付けられたアクセス許可を確認できます。

4. [ターゲット RDS データベースの表示] セクションで、次の操作を行います。

   1. セクションでターゲットデータベースの設定を確認します。
   2. ターゲットデータベースの認証情報を保持する既存のシークレットがない場合は、[シークレット] で [新しいシークレットを作成して使用] を選択します。

      既存のシークレットがある場合は、ドロップダウンからシークレットを選択します。
   3. [シークレットの IAM ロール] で、前のステップのシークレットへのアクセスが許可された IAM ロールを選択します。既存の IAM ロールがない場合は、新しい IAM ロールを作成することもできます。

      ドロップダウンに IAM ロールが表示されない場合は、[IAM ロール ARN] を `arn:aws:iam:account_id:role/roleName` 形式で指定します。

5. [データ移行の設定] セクションで、次の操作を行います。

   1. [フルロード]、[フルロードおよび変更データキャプチャ (CDC)]、[変更データキャプチャ (CDC)] の中から、データ移行のタイプを選択します。これらのパラメータの詳細については「概要」を参照してください。

      移行の開始後に移行タイプを変更することはできません。
   2. 既存の IAM ロールがある場合は、[データ移行の IAM ロール] で [既存の IAM ロールを使用] を選択し、移行に必要なリソースを作成するアクセス許可を DMS に付与する IAM ロールをドロップダウンから選択します。既存の IAM ロールがない場合は、[新しい IAM ロールを作成して使用] を選択します。

   > **Note**
   >
   > アカウントでロールマネージャーが有効になっている場合、Amazon RDS は自動的にロールをアタッチします。また、ここで説明するロールオプション (ソースシークレット用、ターゲットシークレット用、データ移行用) は [カスタマイズ] オプションに置き換えられます。別のロールを使用するには、[カスタマイズ] を選択します。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

6. [移行設定を表示] タブに、データ移行を適切に設定するために必要な設定が表示されていることを確認します。
7. [移行] を選択して、移行のセットアップを完了します。

これらのステップを完了した後、コンソールの進行状況バナーで [詳細を表示] を選択すると、データ移行用に設定されるリソースを確認できます。必要なリソースが設定されると、移行が自動的に開始されます。作成する場合

複数のデータベースをターゲットデータベースに移行するには、新しい EC2 データベースの詳細情報を使用して上記のプロセスを再度開始します。

データ移行の管理

RDS コンソールで [EC2 データベースからデータを移行] アクションを使用すると、RDS が自動的に移行を開始します。

AWS DMS コンソールを使用して移行リソースを作成した場合は、ユーザーが移行プロセスを開始できます。

データ移行の開始

データ移行を開始するには、次の手順に従います。

データ移行を開始する

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. [データの移行] タブの [関連するデータ移行] に、使用可能なデータ移行が一覧表示されます。

   RDS コンソールを使用して設定された移行は、必要なリソースが設定されると自動的に開始されます。

   DMS コンソールを使用して設定された移行は、[準備完了] に設定されます。

   これらの移行を開始するには、[アクション] ドロップダウンを選択し、[開始] を選択します。
4. これにより、EC2 データベースのデータ移行が開始されます。

データ移行の停止

レプリケーションタイプがフルロードのデータ移行の場合、移行を停止するとプロセスが停止し、再開できません。停止したら、移行を再度開始する必要があります。

レプリケーションタイプが変更データキャプチャ (CDC) またはフルロードと CDC に設定された移行の場合、連続するレプリケーションプロセスを停止して、後でプロセスを再開できます。

データ移行の停止

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. [データの移行] タブの [関連するデータ移行] に、進行中のデータ移行が一覧表示されます。

   移行を停止するには、データ移行を選択し、[アクション] ドロップダウンで [停止] を選択します。
4. これにより、EC2 データベースのデータ移行が停止します。

データ移行の再開

レプリケーションタイプがフルロードと変更データキャプチャ (CDC) または変更データキャプチャ (CDC) のデータ移行の場合、CDC プロセスを最後の停止ポイントから再開できます。

データ移行の再開

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. [データの移行] タブの [関連するデータ移行] に、停止中のデータ移行が一覧表示されます。

   移行を再開するには、データ移行を選択し、[アクション] ドロップダウンで [処理を再開] を選択します。
4. これにより、EC2 データベースのデータ移行が再開されます。

データ移行の削除

関連するデータ移行を削除するには、以下の手順に従います。

データ移行の削除

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. 移行を削除するには、データ移行を選択し、[アクション] ドロップダウンで [削除] を選択します。
4. これにより、データ移行が削除されます。

進行中のデータ移行を削除しても、ターゲットデータベースに既にロードされているデータには影響しません。

データ移行の再開

CDC 開始点から関連するデータ移行をやり直すには、次の手順を使用します。

データ移行の再開

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. 移行をやり直すには、データ移行を選択し、[アクション] ドロップダウンで [再起動] を選択します。
4. これにより、CDC 開始点からデータ移行が再度開始されます。

進行中のデータ移行をやり直しても、ターゲットデータベースに既にロードされているデータには影響しません。

データ移行のモニタリング

データ移行を開始したら、移行のステータスと進行状況をモニタリングできます。大規模なデータセットのデータ移行の場合、完了するまでに数時間かかります。データ移行の信頼性、可用性、パフォーマンスを維持するには、移行の進捗を定期的にモニタリングします。

データ移行のステータスと進捗を確認するには

1. RDS コンソールの [データベース] ページでターゲットデータベースを選択します。
2. データベースの詳細ページで、[データの移行] タブを選択します。
3. [関連するデータ移行] セクションにデータ移行が一覧表示されます。[ステータス] 列を確認します。
4. [移行の進行状況] 列に、実行中のデータ移行の移行済みデータの割合が % で表示されます。
5. CloudWatch でプロセスを監視するには、[CloudWatch] 列にあるリンクを使用します。

移行ステータス

実行するデータ移行ごとに、RDS コンソールに [ステータス] が表示されます。次のリストにステータスを示します。

- Ready: データ移行を開始する準備が整っています。
- Starting: RDS はデータ移行のためのサーバーレス環境を作成しています。
- Load running: RDS はフルロード移行を実行しています。
- Load complete, replication ongoing: RDS はフルロードを完了し、進行中の変更をレプリケートしています。このステータスは、フルロードと CDC タイプの移行にのみ適用されます。
- Replication ongoing: RDS は進行中の変更をレプリケートしています。このステータスは CDC タイプの移行にのみ適用されます。
- Stopping: RDS はデータ移行を停止しています。このステータスは、[アクション] メニューからデータ移行を停止する場合に適用されます。
- Stopped: RDS はデータ移行を停止しました。
- Failed: データ移行が失敗しました。詳細については、ログファイルを確認してください。
- Restarting: データ移行は、進行中のデータレプリケーションを CDC 開始点からやり直しました。

チュートリアル: カスタムパラメータと新しいオプショングループを使用した MySQL DB インスタンス の作成

このチュートリアルでは、カスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンス を作成します。カスタムパラメータとオプショングループの詳細については、「Amazon RDS のパラメータグループ」および「オプショングループを使用する」を参照してください。

> **Important**
>
> AWS アカウントを作成するための料金はかかりません。ただし、このチュートリアルを完了すると、使用する AWS リソースのコストが発生する可能性があります。これらのリソースが不要になった場合は、チュートリアルの完了後に削除できます。

カスタム構成と設定で DB インスタンスを作成するには、カスタムパラメータと新しいオプショングループを使用できます。カスタムパラメータと新しいオプショングループは、複数のデータベースを使用し、それらの設定を統一したい場合に特に便利です。

これらの手順を実行することで、次のことが理解できます。

- Amazon RDS を使用して、カスタムパラメータと新しいオプショングループを持つ MySQL DB インスタンスを作成する方法。
- MySQL DB インスタンスの特定のパラメータとオプションを使用する方法。

このチュートリアルを完了するには、次のタスクを実行します。

1. MySQL パラメータ `default_password_lifetime` と `disconnect_on_expired_password` を使用して、カスタムパラメータグループを作成します。
2. MySQL オプション機能 `MariaDB Audit Plugin` を使用して新しいオプショングループを作成します。オプショングループを作成する手順については、「オプショングループを使用する」を参照してください。
3. 作成したカスタム パラメータグループと新しいオプショングループを使用して MySQL DB インスタンスを作成します。

トピック

- 前提条件
- Amazon RDS パラメータグループを作成する
- カスタムパラメータグループのパラメータ値を変更する
- 新しい Amazon RDS オプショングループを作成する
- 新しいオプショングループにオプションを追加する
- カスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンスを作成する

前提条件

このチュートリアルでは、AWS アカウントと管理アクセスを持つユーザーが必要です。まだ設定していない場合は、次のセクションの手順を完了してください。

- AWS アカウント へのサインアップ

Amazon RDS パラメータグループを作成する

このチュートリアルでは、コンソールで MySQL DB インスタンスの default_password_lifetime と disconnect_on_expired_password を使用してカスタムパラメータグループを作成する方法について説明します。default_password_lifetime パラメータは、クライアントパスワードが自動的に期限切れになるまでの日数を制御します。disconnect_on_expired_password パラメータは、パスワードの有効期限が切れたときに MySQL DB インスタンスがクライアントを切断するかどうかを制御します。MySQL DB インスタンスで使用できるその他のパラメータの詳細については、「MySQL のパラメータ」「」を参照してください。

パラメータグループを作成するには

1. Amazon RDS コンソールを開き、[パラメータグループ] を選択します。
2. [カスタムパラメータグループ] で、[パラメータグループの作成] を選択します。
3. パラメータグループの詳細を設定します。
    1. パラメータグループの名前を入力します。
    2. パラメータグループの説明を入力します。
    3. [エンジンタイプ] で、[MySQL Community] を選択します。
    4. [パラメータグループファミリー] で、[MySQL 8.0] を選択します。
4. [作成] を選択します。

新しい パラメータグループは、Amazon RDS コンソールの [パラメータグループ] ページに表示されます。次の手順は、パラメータ値を変更してパラメータグループをカスタマイズする方法を示しています。

カスタムパラメータグループのパラメータ値を変更する

次のステップを使用して、「Amazon RDS パラメータグループを作成する」で作成したパラメータグループのパラメータ値を変更します。

パラメータグループ内のパラメータ値を変更するには

1. Amazon RDS コンソールを開き、[パラメータグループ] を選択します。
2. [カスタムパラメータグループ] で、作成した パラメータグループの名前を選択します。
3. [編集] を選択します。
4. [パラメータをフィルタリング] 検索ボックスで、カスタムパラメータ `default_password_lifetime` を検索します。
5. パラメータの横にあるチェックボックスを選択し、このパスワード有効期間パラメータに設定する日数の値を入力します。
6. [Save Changes] (変更を保存) をクリックします。
7. パラメータ `disconnect_on_expired_password` に対して同じ手順を繰り返します。このパラメータを選択すると、ドロップダウンメニューから 0 または 1 の値を選択するように求められます。期限切れのパスワードで切断するには、1 を選択します。

これで、カスタムパラメータグループを Amazon RDS for MySQL 8.0 DB インスタンスに関連付けることができるようになりました。次に、DB インスタンスの新しいオプショングループを作成します。

新しい Amazon RDS オプショングループを作成する

[MariaDB 監査プラグイン] オプションを使用して新しいオプショングループを作成します。このプラグインは、セキュリティとコンプライアンスに関するサーバーアクティビティを ログに記録します。MySQL DB インスタンスで使用できるその他のカスタムオプションの詳細については、「MySQL DB インスタンスのオプション」を参照してください。

オプショングループを作成するには

1. Amazon RDS コンソールを開き、[オプショングループ] を選択します。

2. [オプショングループ] で、[グループを作成] を選択します。
3. オプショングループの詳細を設定します。
   - オプショングループの名前を入力します。
   - オプショングループの説明を入力します。
   - [エンジン] で [MySQL] を選択します。
   - [メジャーエンジンバージョン] で、[8.0] を選択します。
4. [作成] を選択します。

新しいオプショングループは、Amazon RDS コンソールの [オプショングループ] ページに表示されます。次の手順は、オプショングループにオプションを追加する方法を示しています。

新しいオプショングループにオプションを追加する

次の手順を使用して、「新しい Amazon RDS オプショングループを作成する」で作成した新しいオプショングループにオプションを追加します。

オプショングループにオプションを追加するには

1. Amazon RDS コンソールを開き、[オプショングループ] を選択します。
2. [オプショングループ] で、作成したオプショングループの名前を選択します。
3. [オプション] で、[オプションの追加] を選択します。
4. オプショングループの詳細を設定します。
   - [オプション名] で、MariaDB 監査プラグインの [MARIADB_AUDIT_PLUGIN] オプションを選択します。
   - [オプション設定] で、すべてのデフォルトオプションを選択したままにします。
   - [すぐに適用] で、[はい] を選択します。
5. [オプションの作成] を選択します。

これで、関連付けられたすべての DB インスタンスでこの新しいオプショングループが使用可能になりました。次に、カスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンスを作成します。

カスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンスを作成する

最後に、前のステップで作成したカスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンスを作成します。次の手順は、カスタムパラメータと新しいオプショングループを使用して MySQL DB インスタンスを作成する方法を示しています。

カスタムパラメータと新しいオプショングループを使用して DB インスタンスを作成するには

1. Amazon RDS コンソールを開き、[データベース] を選択します。
2. [データベースの作成] を選択します。
3. [データベース作成方法を選択] で [標準作成] を選択します。
4. [エンジンのオプション] として、[MySQL] を選択します。
5. [可用性と耐久性] で、[単一の DB インスタンス] を選択します。このステップは、カスタムパラメータまたは新しいオプショングループをサポートするために必要です。
6. [追加設定] を選択します。
   - [最初のデータベース名] で、DB インスタンスの名前を選択します。
   - [DB パラメータグループ] ドロップダウンで、前に作成したカスタムパラメータグループの名前を選択します。
   - [オプショングループ] ドロップダウンで、前に作成した新しいオプショングループの名前を選択します。
7. このチュートリアルでは、他の DB 設定はデフォルト設定のままにすることも、必要に応じて変更することもできます。
8. [データベースの作成] を選択します。

RDS は、カスタムパラメータグループと新しいオプショングループを使用して新しい MySQL DB インスタンスを作成します。このデータベースの詳細については、Amazon RDS コンソールの「データベース」のページを参照してください。

このチュートリアルでは、カスタムパラメータと新しいオプショングループを使用して、カスタマイズされた設定で MySQL DB インスタンスを設定します。この新しく作成された MySQL DB インスタンスは、`default_password_lifetime` パラメータを使用してユーザーパスワードの有効期間を管理します。この DB インスタンスは、`disconnect_on_expired_password` パラメータを使用して、期限切れのパスワードで接続するユーザーも切断します。また、`MariaDB Audit Plugin`

オプションを使用して、サーバーアクティビティも追跡します。データベースを最適化するには、カスタムパラメータグループに追加の設定を適用し、オプションを追加します。

カスタマイズした DB インスタンスの作成が完了したら、不要なコストが発生しないようにリソースを削除する必要があります。DB インスタンスを削除するには、「DB インスタンスを削除する」の手順に従います。

Amazon RDS DB インスタンスの管理

Amazon RDS DB インスタンスの管理と保守の手順は次の通りです。

トピック

- 一時的に Amazon RDS DB インスタンスを停止する
- 以前に停止した Amazon RDS DB インスタンスを開始する
- DB インスタンスの再起動
- EC2 インスタンスと DB インスタンスを自動的に接続する
- Lambda 関数と DB インスタンスを自動的に接続する
- Amazon RDS DB インスタンスを変更する
- DB インスタンスのメンテナンス
- DB インスタンスのエンジンバージョンのアップグレード
- DB インスタンスの名前を変更する
- DB インスタンスのリードレプリカの操作
- Amazon RDS リソースのタグ付け
- Amazon RDS の Amazon リソースネーム (ARN)
- Amazon RDS DB インスタンスのストレージを使用する
- DB インスタンスを削除する
- チュートリアル: 開発から本番環境までの MySQL DB インスタンス環境の管理

一時的に Amazon RDS DB インスタンスを停止する

一時的なテストや毎日の開発作業のために、DB インスタンスを断続的に (最大連続日数は 7 日間まで) 停止できます。最も一般的なユースケースはコスト最適化です。DB インスタンスを停止する時間は、いくつかの要因によって異なります。詳細については、「ユースケース」および「時間に関する考慮事項」を参照してください。

> **Warning**
>
> DB インスタンスの起動にはインスタンスの復旧が必要で、数分から数時間かかる場合があります。したがって、インスタンスの可用性が懸念される場合は、本番稼働用インスタンスを一時的に停止することに注意してください。詳細については、「以前に停止した Amazon RDS DB インスタンスを開始する」を参照してください。

同じオペレーションで DB インスタンスを停止して起動するには、DB インスタンスを再起動します。詳細については、「 DB インスタンスの再起動」を参照してください。

トピック

- DB インスタンス停止のユースケース
- DB インスタンスを停止する際の時間に関する考慮事項
- サポートされている DB エンジン、インスタンスクラス、リージョン
- マルチ AZ 配置で DB インスタンスを停止する
- DB インスタンスの停止の仕組み
- DB インスタンスの停止に関する制限事項
- オプショングループとパラメータグループに関する考慮事項
- パブリック IP アドレスに関する考慮事項
- DB インスタンスの一時停止: 基本的な手順

DB インスタンス停止のユースケース

DB インスタンスを停止して起動する方が、DB スナップショットを作成して DB インスタンスを削除した後、インスタンスにアクセスするときにスナップショットを復元するよりも高速かつ効率的です。インスタンス停止の一般的なユースケースは次のとおりです。

- コスト最適化 — 非本番データベースの場合、コストを節約するために Amazon RDS DB インスタンスを一時的に停止できます。インスタンスの停止中は、DB インスタンス時間に対して課金されません。

> **Important**
>
> DB インスタンスが停止していても、プロビジョニング済みストレージ (プロビジョンド IOPS を含む) に対して課金されます。また、指定された保持ウィンドウ内の手動スナップショットや自動バックアップを含むバックアップストレージに対しても課金されます。さらに、DB インスタンスがパブリックにアクセスできる場合、パブリック IPv4 アドレスに対して引き続き課金されます。ただし、DB インスタンス時間に対しては請求されません。詳細については、「請求に関するよくある質問」を参照してください。

- 日常的な開発作業 – 開発目的のために DB インスタンスを維持する場合は、必要なときにインスタンスを起動し、必要でないときはインスタンスをシャットダウンできます。
- テスト — バックアップとリカバリの手順、移行、アプリケーションのアップグレード、または関連するアクティビティをテストするために、一時的な DB インスタンスが必要になる場合があります。このようなユースケースでは、必要でないときは DB インスタンスを停止できます。
- トレーニング – RDS でトレーニングを行っている場合は、トレーニングセッション中に DB インスタンスを起動し、後でシャットダウンする必要があるかもしれません。

DB インスタンスを停止する際の時間に関する考慮事項

DB インスタンスを停止する時間は、インスタンスクラス、ネットワーク状態、DB エンジンタイプ、データベース状態、ワークロードなどの要因によって異なります。このプロセスには数分、または最大 1 時間かかる場合があります。最終的なスナップショットを作成することを選択しなくても、サービスはデータの整合性を確保するために内部オペレーションを実行する必要があります。これらのオペレーションにより、停止時間が長くなる場合があります。スキップすることはできません。DB インスタンスのステータスは、すべてのオペレーションが完了するまで停止状態のままです。これらのオペレーションには以下のアクションが含まれます。

- データベースエンジンプロセスをシャットダウンします。
- RDS プラットフォームプロセスをシャットダウンします。
- DB インスタンスに関連付けられている EBS ストレージボリュームをデタッチします。
- 基盤となる Amazon EC2 インスタンスを停止します。

遅延を最小限に抑えるには、次のアクションを実行することを検討してください。

- 手動バックアップを作成します。
- アクティビティが少ない間は DB インスタンスを停止します。
- 最新の自動バックアップを維持します。

サポートされている DB エンジン、インスタンスクラス、リージョン

以下のエンジンを実行中の Amazon RDS DB インスタンスを停止して再度起動することができます。

- Db2
- MariaDB
- Microsoft SQL Server (RDS Custom for SQL Server を含む)
- MySQL
- Oracle
- PostgreSQL

DB インスタンスの停止と起動は、すべての DB インスタンスクラスとすべての AWS リージョンでサポートされています。

マルチ AZ 配置で DB インスタンスを停止する

マルチ AZ 配置で DB インスタンスを停止および開始できます。以下の制限事項に留意してください。

- マルチ AZ 配置は、データベースエンジンがサポートしている場合にのみ作成できます。エンジンのサポートの詳細については、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。
- RDS for SQL Server では、マルチ AZ 配置の DB インスタンスの停止はサポートされていません。詳細については、「Microsoft SQL Server マルチ AZ 配置の制限、注意事項、および推奨事項」を参照してください。
- DB インスタンスを停止するには、長い時間がかかる場合があります。

DB インスタンスの停止の仕組み

停止操作は、次の段階で実行されます。

1. DB インスタンスは通常のシャットダウンプロセスを開始します。

   DB インスタンスのステータスが `stopping` に変更されます。
2. インスタンスは最大 7 日間連続して実行を停止します。

   DB インスタンスのステータスが `stopped` に変更されます。

停止した DB インスタンスの特徴

停止状態にある DB インスタンスには次の特徴があります。

- 停止状態の DB インスタンスでは、次のものが保持されます。
  - インスタンス ID
  - ドメインネームサーバー (DNS) エンドポイント
  - パラメータグループ
  - セキュリティグループ
  - オプショングループ
  - Amazon S3 トランザクションログ (特定の時点への復旧に必要)

  DB インスタンスを再起動すると、停止したときと同じ構成になります。
- すべてのストレージボリュームは DB インスタンスに接続されたままとなり、データは保持されます。RDS は、DB インスタンスの RAM に保存されているデータを削除します。

  DB インスタンスが停止していても、プロビジョニング済みストレージ (プロビジョンド IOPS を含む) に対して課金されます。また、指定された保持ウィンドウ内の手動スナップショットや自動バックアップを含むバックアップストレージに対しても課金されます。
- RDS は、予定されているメンテナンスアップデートも含め、保留中のアクションを削除します。ただし DB インスタンスのオプショングループまたは DB パラメータグループの保留中アクションは除きます。

> **Note**
>
> RDS for PostgreSQL DB インスタンスが正常にシャットダウンしないことがあります。この場合、後で再起動すると、インスタンスで復旧プロセスが実行されていることがわかります。これは、データベースの整合性を保護することを目的とした、データベースエンジンで想定されている動作です。一部のメモリベースの統計およびカウンターでは履歴は保持されず、再起動後に再び初期化され、実行している運用ワークロードをキャプチャします。

停止した DB インスタンスの自動再起動

連続 7 日後に DB インスタンスを手動で起動しなかった場合、RDS は DB インスタンスを自動的に起動します。これにより、インスタンスに必要なメンテナンスの更新が遅延することはありません。インスタンスをスケジュールに従って停止および起動する方法については、「Step Functions を使用して Amazon RDS インスタンスを 7 日以上停止する方法を教えてください」を参照してください。

DB インスタンスの停止に関する制限事項

停止オペレーションには次のようないくつかの制限があります。

- リードレプリカが含まれているか、リードレプリカである DB インスタンスは停止できません。
- 停止された DB インスタンスを変更することはできません。
- 停止された DB インスタンスに関連付けられているオプショングループを削除することはできません。
- 停止した DB インスタンスに関連付けられている DB パラメータグループを削除することはできません。
- マルチ AZ 配置では、次の制限に注意してください。
  - RDS for SQL Server の DB インスタンスは停止できません。
  - DB インスタンスのスタート後に、プライマリおよびセカンダリのアベイラビリティーゾーンが切り替わる場合があります。

RDS Custom for SQL Server に対して、追加の制限が適用されます。詳細については、「RDS Custom for SQL Server DB インスタンスの起動と停止」を参照してください。

オプショングループとパラメータグループに関する考慮事項

オプショングループに関連付けられた DB インスタンスがある場合、永続オプション (固定オプションを含む) をオプショングループから削除することはできません。この機能は、`stopping`、`stopped`、`starting` の状態の DB インスタンスにも当てはまります。

停止した DB インスタンスに関連付けられている DB パラメータグループを変更することはできません。ただし、この変更は DB インスタンスを次回起動するまで発生しません。変更をすぐに適用することを選択した場合、変更は DB インスタンスの起動時に発生します。それ以外の場合は、DB インスタンスを起動後の次のメンテナンスウィンドウで変更が発生します。

パブリック IP アドレスに関する考慮事項

DB インスタンスを停止すると、その DNS エンドポイントが保持されます。パブリック IP アドレスを持つ DB インスタンスを停止すると、Amazon RDS はパブリック IP アドレスを保持します。DB インスタンスが再起動されると、同じパブリック IP アドレスが割り当てられます。

> **Note**
>
> DB インスタンスへの接続には、IP アドレスではなく、常に DNS エンドポイントを使用する必要があります。

DB インスタンスの一時停止: 基本的な手順

AWS マネジメントコンソール、AWS CLI、RDS API のいずれかを使用して DB スナップショットを停止できます。

コンソール

DB インスタンスを停止するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、停止する DB インスタンスを選択します。
3. [Actions] (アクション) として、[Stop temporarily] (一時的に停止) を選択します。
4. [Stop DB instance temporarily] (DB インスタンスを一時的に停止) ウィンドウで、DB インスタンスが 7 日後に自動的に再起動することを確認するメッセージを選択します。

5. (オプション) [Save the DB instance in a snapshot] (DB インスタンスをスナップショットに保存する) を選択して、[Snapshot name] (スナップショット名) としてスナップショット名を入力します。DB インスタンスを停止する前に DB インスタンスのスナップショットを作成する場合は、このオプションを選択します。
6. [Stop temporarily] (一時的に停止) を選択して DB インスタンスを停止するか、[Cancel] (キャンセル) を選択して操作をキャンセルします。

AWS CLI

AWS CLI を使用して DB インスタンスを停止するには、次のオプションを指定して stop-db-instance コマンドを呼び出します。

- `--db-instance-identifier` - DB インスタンスの名前です。

Example

```
aws rds stop-db-instance --db-instance-identifier mydbinstance
```

RDS API

Amazon RDS API を使用して DB インスタンスを停止するには、以下のパラメータを指定して StopDBInstance オペレーションを呼び出します。

- `DBInstanceIdentifier` - DB インスタンスの名前です。

以前に停止した Amazon RDS DB インスタンスを開始する

このトピックでは、以前に停止した Amazon RDS DB インスタンスを起動する方法について説明し、データベースの使用を再開するために必要な手順と重要な考慮事項の概要を示します。

コストを節約するために一時的に Amazon RDS DB インスタンスを停止できます。DB インスタンスを停止した後は、使用を再開するために再起動できます。DB インスタンスの停止の詳細については、「一時的に Amazon RDS DB インスタンスを停止する」を参照してください。

以前に停止した DB インスタンスを開始する場合、この DB インスタンスには、以下のような情報が保持されます。

- [インスタンス ID]
- ドメインネームサーバー (DNS) エンドポイント
- DB パラメータグループ
- VPC セキュリティグループ
- DB オプショングループ

Amazon RDS は、データベースインスタンスとアタッチされたストレージに対して 1 秒単位で請求します。インスタンスの起動には 10 分の最低料金がかかります。

インスタンスを起動するために、Amazon RDS サービスは次のようなアクションを実行します。

- 基盤となる Amazon EC2 インスタンスのプロビジョニング
- RDS プロセスの開始
- データベースエンジンプロセスの開始
- EBS ストレージボリュームのアタッチ
- Performance Insights が以前に有効になっている場合の有効化
- DB インスタンスの復旧 (通常のシャットダウン後も復旧が発生します)

DB インスタンスを起動する時間は、インスタンスクラス、ネットワーク状態、DB エンジンタイプ、データベースサイズ、インスタンスがシャットダウンされたときのデータベース状態などの要因によって異なります。このプロセスには数分から数時間かかることがあります。アベイラビリティープランを作成するときは、起動時間のばらつきを考慮することをお勧めします。

コンソール

DB インスタンスを開始するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[データベース] を選択し、開始する DB インスタンスを選択します。
3. [Actions] (アクション) で [Start] (開始) を選択します。

AWS CLI

AWS CLI を使用して DB インスタンスを開始するには、次のオプションを指定して start-db-instance コマンドを呼び出します。

- `--db-instance-identifier` - DB インスタンスの名前です。

Example

```
aws rds start-db-instance --db-instance-identifier mydbinstance
```

RDS API

Amazon RDS API を使用して DB インスタンスを開始するには、次のパラメータを指定して StartDBInstance オペレーションを呼び出します。

- `DBInstanceIdentifier` - DB インスタンスの名前です。

DB インスタンスの再起動

RDS DB インスタンスのデータベースサービスは、再起動と呼ばれる 1 回のオペレーションで停止および開始できます。再起動は、設定変更を適用したり、軽微な問題を解決したり、データベースの完全な再起動や移行を行わずにネットワークの問題を解決するために必要になることがあります。

> **Note**
>
> このトピックは、シングル AZ またはマルチ AZ の DB インスタンスの再起動にのみ適用されます。マルチ AZ DB クラスターを再起動する手順については、「the section called “マルチ AZ DB クラスターの再起動”」を参照してください。

トピック

- DB インスタンスの再起動のユースケース
- DB インスタンスの再起動の仕組み
- マルチ AZ 配置での DB インスタンスの再起動の仕組み
- DB インスタンスを再起動する際の考慮事項
- DB インスタンスを再起動するための前提条件
- DB インスタンスの再起動: 基本的な手順

DB インスタンスの再起動のユースケース

通常、メンテナンスの理由から DB インスタンスを再起動して、変更を有効にします。一般的なユースケースは次のとおりです。

- 新しい DB パラメータグループの関連付け – 新しい DB パラメータグループを DB インスタンスに関連付ける場合、RDS は、DB インスタンスが再起動された後にのみ、変更された静的および動的パラメータを適用します。ただし、DB インスタンスに関連付けた後に DB パラメータグループの動的パラメータを変更すると、これらの変更は再起動せずに直ちに適用されます。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
- 既存の DB パラメータグループの静的パラメータへの変更適用 – 静的パラメータを変更して DB パラメータグループを保存すると、コンソールでこのパラメータグループに関連付けられた DB インスタンスのステータスが [再起動を保留中] に変わります。パラメータの変更は、関連付けられた

DB インスタンスが再起動された後にのみ有効になります。動的パラメータを変更すると、デフォルトでパラメータの変更が直ちに有効になります。再起動は不要です。

> **Note**
>
> [再起動を保留中] のステータスにより、次回のメンテナンスウィンドウで自動的に再起動されることはありません。パラメータの最新の変更を DB インスタンスに適用するには、DB インスタンスを手動で再起動します。パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

- トラブルシューティング — パフォーマンスやその他の運用上の問題が発生し、再起動が必要になる場合があります。例えば、DB インスタンスが応答しない可能性があります。

DB インスタンスの再起動の仕組み

Amazon RDS で DB インスタンスが再起動されると、次のシーケンシャルタスクが実行されます。

1. DB インスタンスのデータベースサービスを停止する
2. DB インスタンスでデータベースサービスを開始するす。

再起動プロセスにより、短時間の停止が発生します。この停止中、DB インスタンスのステータスは再起動中になります。シングル AZ 配置とマルチ AZ DB インスタンス配置の両方で、フェイルオーバーで再起動した場合でも停止します。

マルチ AZ 配置での DB インスタンスの再起動の仕組み

RDS DB インスタンスがマルチ AZ 配置にある場合は、フェイルオーバーで再起動できます。このオペレーションは、フェイルオーバー後に DB インスタンスの障害をシミュレートしたり、元のアベイラビリティーゾーンにオペレーションを復元したりするのに役立ちます。

フェイルオーバーによる再起動中、Amazon RDS は以下を実行します。

- データベースを突然中断します。そのため、DB インスタンスとそのクライアントセッションが正常にシャットダウンする時間がない場合があります。

> **Warning**
>
> データが損失する可能性を回避するため、フェイルオーバーで再起動する前に DB インスタンスでトランザクションを停止することをお勧めします。

- 必要に応じてデータベースのクラッシュリカバリを実行します。
- 別の AZ のスタンバイレプリカに自動的に切り替えます。AZ の変更は AWS マネジメントコンソール、および AWS CLI と RDS API への呼び出しに数分反映されない場合があります。
- スタンバイ DB インスタンスを指すように DB インスタンスの DNS レコードを更新します。したがって、DB インスタンスへの既存の接続のクリーンアップと再確立が必要になります。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。
- 再起動後に Amazon RDS イベントを作成します。

RDS for Microsoft SQL Server では、フェイルオーバーではプライマリ DB インスタンスのみが再起動されます。フェイルオーバー後、プライマリ DB インスタンスは新しいセカンダリ DB インスタンスになります。マルチ AZ インスタンスのパラメータは更新されない可能性があります。フェイルオーバーなしの再起動の場合、プライマリ DB インスタンスとセカンダリ DB インスタンスの両方が再起動し、再起動後にパラメータが更新されます。DB インスタンスが応答しない場合は、フェイルオーバーなしで再起動することをお勧めします。

DB インスタンスを再起動する際の考慮事項

インスタンスを再起動する前に、次の点を考慮してください。

- リードレプリカを持つ DB インスタンスの場合、ソース DB インスタンスとそのリードレプリカを個別に再起動できます。再起動が完了すると、レプリケーションが自動的に再開されます。
- 再起動にかかる時間は、クラッシュ回復プロセス、再起動時のデータベースアクティビティ、および特定の DB エンジンの動作によって異なります。再起動時間を短くするには、再起動中のデータベースアクティビティをできる限り減らすことをお勧めします。この手法は、転送中のトランザクションのロールバックアクティビティを減らします。

DB インスタンスを再起動するための前提条件

次の前提条件を満たしていることを確認します。

- DB インスタンスは `available` の状態である必要があります。データベースは、バックアップが進行中または、以前の要求による変更、メンテナンス時間のオペレーションなど、いくつかの理由で使用できない場合があります。
- 別の AZ へのフェイルオーバーを強制的に実行する場合は、DB インスタンスをマルチ AZ 用に設定する必要があります。
- 別の AZ へのフェイルオーバーを強制的に実行する場合は、データ損失の可能性を防ぐため、まず DB インスタンスのトランザクションを停止することをお勧めします。

DB インスタンスの再起動: 基本的な手順

DB インスタンスの再起動には、AWS マネジメントコンソール、AWS CLI または RDS API を使用します。

コンソール

DB インスタンスを再起動するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、再起動する DB インスタンスを選択します。
3. [アクション] で、[再起動] を選択します。

   [DB インスタンスを再起動] ページが表示されます。
4. (省略可能) [フェイルオーバーし再起動しますか?] を選択し、別の AZ へのフェイルオーバーを強制的に実行します。
5. [Reboot] を選択して DB インスタンスを再起動します。

   または、[Cancel] を選択します。

AWS CLI

AWS CLI を使用して DB インスタンスを再起動するには、`reboot-db-instance` コマンドを呼び出します。

Exampleシンプルな再起動

Linux、macOS、Unix の場合:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

Windows の場合:

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

Exampleフェイルオーバーによる再起動

マルチ AZ DB クラスターで、ある AZ から別の AZ へのフェイルオーバーを強制的に実行するには、`--force-failover` パラメータを使用します。

Linux、macOS、Unix の場合:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance \
    --force-failover
```

Windows の場合:

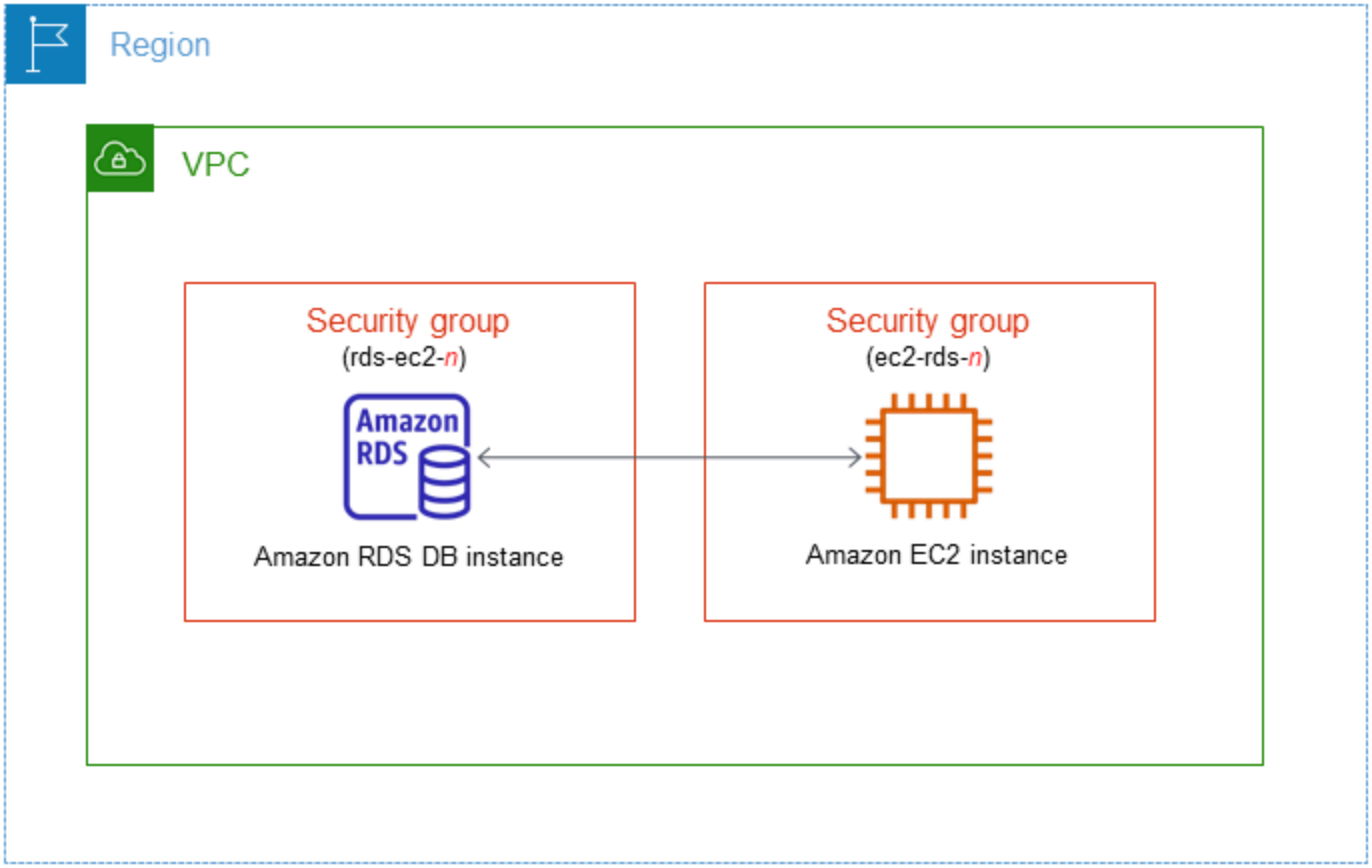
```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance ^
    --force-failover
```

RDS API

Amazon RDS API を使用して DB インスタンスを再起動するには、`RebootDBInstance` オペレーションを呼び出します。

EC2 インスタンスと DB インスタンスを自動的に接続する

RDS コンソールを使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスと DB インスタンスとの接続を簡単に設定できます。多くの場合、DB インスタンスはプライベートサブネットにあり、EC2 インスタンスは VPC 内のパブリックサブネットにあります。EC2 インスタンスの SQL クライアントを使用して、DB インスタンス に接続できます。EC2 インスタンスは、プライベート DB インスタンス にアクセスするウェブサーバーやアプリケーションを実行することもできます。EC2 インスタンスとマルチ AZ DB クラスター間の接続をセットアップする方法については、「the section called “EC2 インスタンスとマルチ AZ DB クラスターの接続”」を参照してください。



DB インスタンスと同じ VPC にない EC2 インスタンスに接続する場合は、「VPC の DB インスタンスにアクセスするシナリオ」のシナリオを参照してください。

トピック

- EC2 インスタンスとの自動接続の概要
- EC2 インスタンスと RDS データベースを自動的に接続する
- 接続中のコンピューティングリソースを表示する

- 特定の DB エンジンを実行している DB インスタンスに接続する

EC2 インスタンスとの自動接続の概要

EC2 インスタンスと RDS データベース間の接続を設定すると、Amazon RDS は EC2 インスタンスと RDS データベース の VPC セキュリティグループを自動的に設定します。

EC2 インスタンスと RDS データベース を接続するための要件は次のとおりです。

- EC2 インスタンスは RDS データベース と同じ VPC に存在する必要があります。

  同じ VPC に EC2 インスタンスが存在しない場合、コンソールには EC2 インスタンス作成用のリンクが表示されます。
- 接続を設定するユーザーには、以下の Amazon EC2 オペレーションを実行するアクセス許可が必要です。
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

DB インスタンスと EC2 インスタンスが異なるアベイラビリティーゾーンにある場合、アベイラビリティーゾーン間のコストが発生する可能性があります。

EC2 インスタンスへの接続を設定すると、次の表で示されているように、Amazon RDS は、RDS データベースと EC2 インスタンスに関連付けられているセキュリティグループの現在の設定に基づいて動作します。

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
RDS データベース に関連付けられたセキュリティグループ	パターン `ec2-rds-`*n* (*n* は数字) に一致する名前の EC2	RDS は何のアクションも実行しません。

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
には、パターン `rds-ec2-`*n* (*n* は数字) に一致する名前のセキュリティグループが 1 つまたは複数あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	インスタンスに関連付けられたセキュリティグループが 1 つまたは複数存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、RDS データベース の VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	EC2 インスタンスと RDS データベース との接続は、既に自動で設定されています。EC2 インスタンスと RDS データベースの間には既に接続が存在するため、セキュリティグループは変更されません。

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
次の条件のいずれかが適用されます。 • RDS データベース には、パターン `rds-ec2-`*n* と名前が一致する関連付けられたセキュリティグループはありません。 • RDS データベース に関連付けられたセキュリティグループには、パターン `rds-ec2-`*n* に一致する名前のセキュリティグループが 1 つまたは複数あります。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、EC2 インスタンスとの接続には使用できません。Amazon RDS は、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。変更の例としては、ルールの追加や、既存ルールのポート変更などがあります。	次の条件のいずれかが適用されます。 • EC2 インスタンスに関連付けされた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループは存在しません。 • EC2 インスタンスに関連付けられた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、RDS データベースとの接続には使用できません。Amazon RDS は、RDS データベース の VPC セキュリティグループをソースとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
RDS データベース に関連付けられたセキュリティグループには、パターン `rds-ec2-`*n* に一致する名前のセキュリティグループが 1 つまたは複数あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	EC2 インスタンスに関連付けられた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、RDS データベースとの接続には使用できません。Amazon RDS は、RDS データベース の VPC セキュリティグループをソースとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	RDS action: create new security groups
RDS データベース に関連付けられたセキュリティグループには、パターン `rds-ec2-`*n* に一致する名前のセキュリティグループが 1 つまたは複数あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	接続に有効な EC2 セキュリティグループは存在しますが、EC2 インスタンスに関連付けられていません。このセキュリティグループには、パターン `ec2-rds-`*n* に一致する名前が付いています。これは変更されていません。また、これには RDS データベース の VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	RDS action: associate EC2 security group

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
次の条件のいずれかが適用されます。 • RDS データベース には、パターン `rds-ec2-`*n* と名前が一致する関連付けられたセキュリティグループはありません。 • RDS データベース に関連付けられたセキュリティグループには、パターン `rds-ec2-`*n* に一致する名前のセキュリティグループが 1 つまたは複数あります。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、EC2 インスタンスとの接続には使用できません。Amazon RDS は、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	EC2 インスタンスに関連付けられた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、RDS データベース の VPC セキュリティグループをソースとするアウトバウンドルールが 1 つのみ存在します。	RDS action: create new security groups

RDS アクション: 新しいセキュリティグループを作成する

Amazon RDS は以下のアクションを実行します。

- パターン `rds-ec2-n` に一致する新しいセキュリティグループを作成します。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが存在します。このセキュリティグループでは、RDS データベース に関連付けられており、EC2 インスタンスが RDS データベースへのアクセスを許可します。
- パターン `ec2-rds-n` に一致する新しいセキュリティグループを作成します。このセキュリティグループには、RDS データベースの VPC セキュリティグループをターゲットとするアウトバウンドルールが存在します。このセキュリティグループには EC2 インスタンスに関連付けられ、EC2 インスタンスが RDS データベース にトラフィックの送信を許可します。

RDS アクション: EC2 セキュリティグループを関連付ける

Amazon RDS は、有効な既存の EC2 セキュリティグループを EC2 インスタンスに関連付けます。このセキュリティグループにより、EC2 インスタンスは RDS データベース にトラフィックの送信を許可します。

EC2 インスタンスと RDS データベースを自動的に接続する

EC2 インスタンスと RDS データベース との接続を設定する前に、「EC2 インスタンスとの自動接続の概要」で説明されている要件を満たしていることを確認してください。

接続の設定後にこれらのセキュリティグループを変更すると、EC2 インスタンスと RDS データベース との接続に影響する可能性があります。

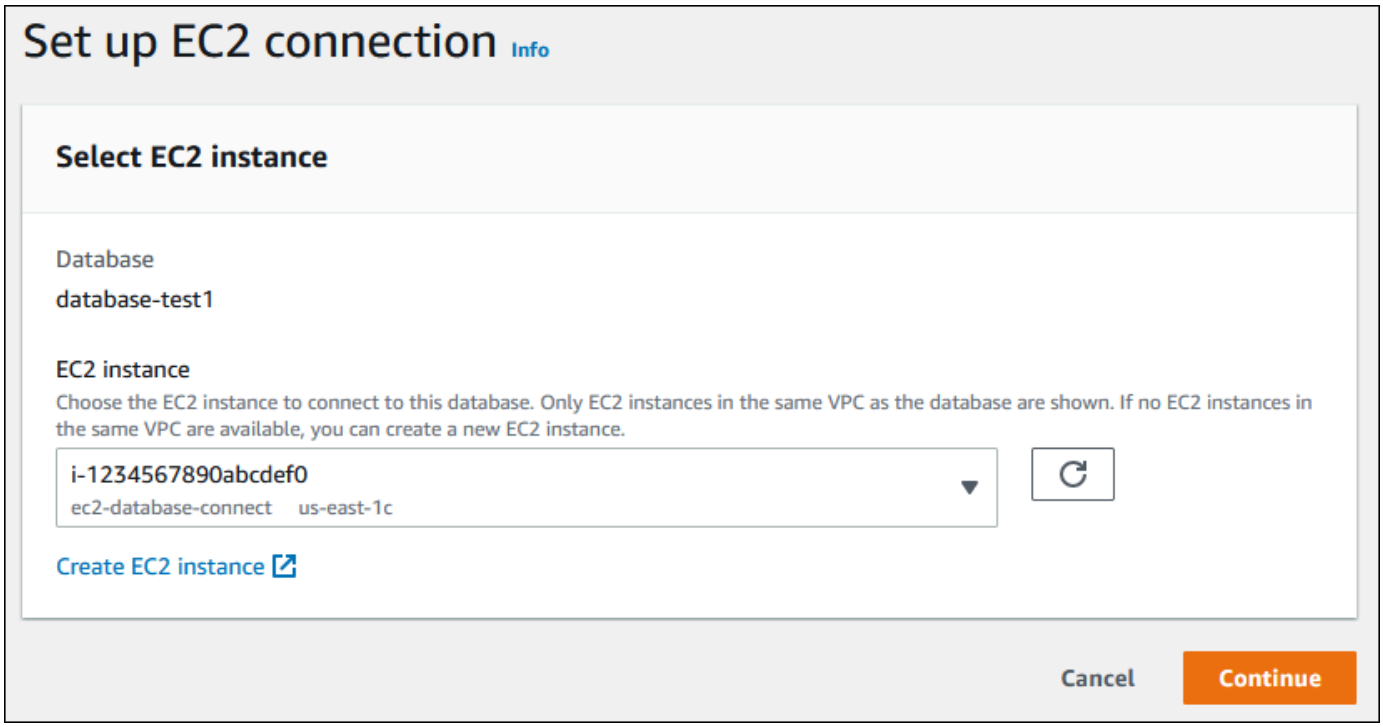
> **Note**
>
> AWS マネジメントコンソール を使用することでのみ、EC2 インスタンスと RDS データベース との接続を自動で設定できます。AWS CLI または RDS API を使用して自動で接続を設定することはできません。

EC2 インスタンスと RDS データベース を自動的に接続するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、RDS データベース のリンクを選択します。
3. [アクション] から [EC2 接続の設定] を選択します。

[Set up EC2 connection] (EC2 接続の設定) ページが表示されます。

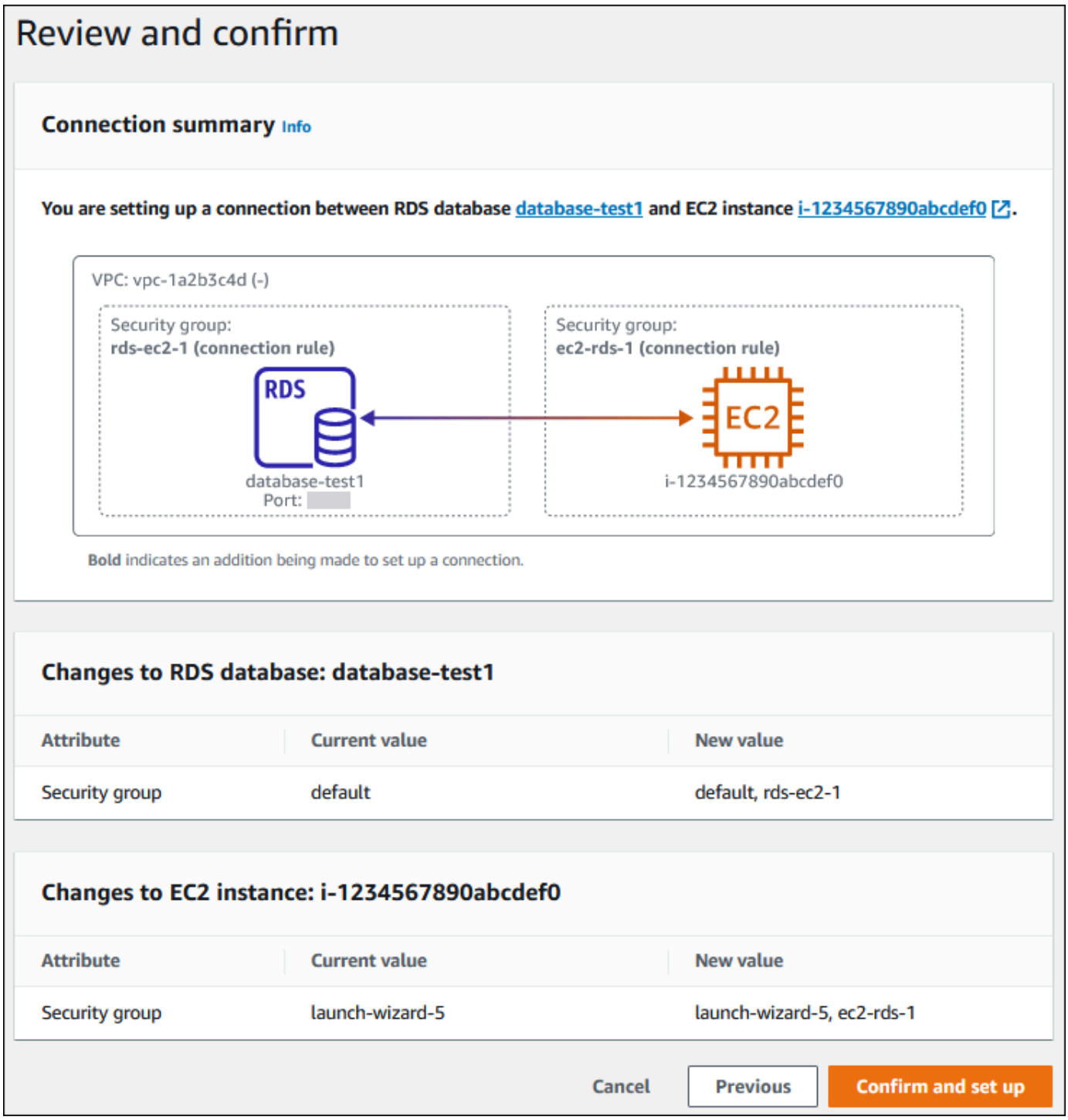
4. [Set up EC2 connection] (EC2 接続の設定) ページで、[EC2 instance] (EC2 インスタンス) を選択します。



同じ VPC に EC2 インスタンスが存在しない場合は、[Create EC2 instance] (EC2 インスタンスの作成) を選択します。この場合、新しい EC2 インスタンスが RDS データベース と同じ VPC にあることを確認してください。

5. [続行] をクリックしてください。

[Review and confirm] (確認と確定) ページが表示されます。



6. [Review and confirm] (確認と確定) ページで、EC2 インスタンスとの接続を設定するために RDS が行う変更を確認します。

   変更が正しければ、[確認とセットアップ] を選択します。

   変更内容が正しくない場合は、[Previous] (前へ) または [Cancel] (キャンセル) を選択します。

接続中のコンピューティングリソースを表示する

AWS マネジメントコンソール を使用して、RDS データベースに接続されているコンピューティングリソースを表示できます。表示されるリソースには、自動的に設定されたコンピューティングリソース接続が含まれます。コンピューティングリソースとの接続は、次の方法で自動的に設定できます。

- データベースを作成するときに、コンピューティングリソースを選択できます。

  詳細については、「Amazon RDS DB インスタンスの作成」および「Amazon RDS 用のマルチ AZ DB クラスターの作成」を参照してください。
- 既存のデータベースとコンピューティングリソース間の接続を設定できます。

  詳細については、「EC2 インスタンスと RDS データベースを自動的に接続する」を参照してください。

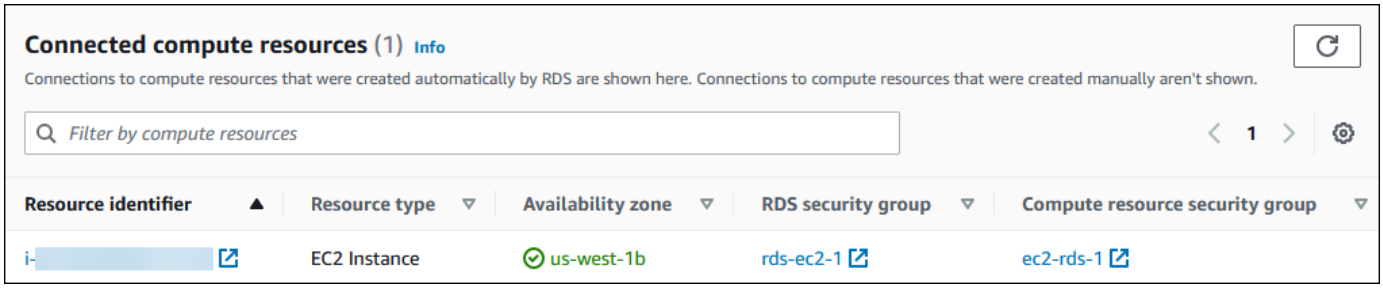
コンピューティングリソースリストには、手動でデータベースに接続されたものは含まれていません。例えば、データベースに関連付けられた VPC セキュリティグループにルールを追加することで、コンピューティングリソースがデータベースに手動でアクセスできるようになります。

コンピューティングリソースをリスト化するには、次の条件を満たしている必要があります。

- コンピューティングリソースに関連付けられているセキュリティグループの名前がパターン `ec2-rds-`*n* (*n* は数字) と一致する。
- コンピューティングリソースに関連付けられたセキュリティグループには、ポート範囲が RDS データベースが使用するポートに設定されたアウトバウンドルールがあります。
- コンピューティングリソースに関連付けられたセキュリティグループには、ソースが RDS データベース に関連付けられたセキュリティグループに設定されたアウトバウンドルールがある。
- RDS データベースに関連付けられたセキュリティグループの名前が、パターン `rds-ec2-`*n* (*n* は数字) に一致する。
- RDS データベースに関連付けられたセキュリティグループには、ポート範囲が RDS データベースが使用するポートに設定されたインバウンドルールがあります。
- RDS データベース に関連付けられたセキュリティグループには、ソースがコンピューティングリソースに関連付けられたセキュリティグループに設定されたインバウンドルールがある。

RDS データベースに接続されているコンピューティングリソースを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、RDS データベース の名前を選択します。
3. [Connectivity & security] (接続とセキュリティ) タブの [Connected compute resources] (接続されたコンピューティングリソース) にコンピューティングリソースが表示されます。



特定の DB エンジンを実行している DB インスタンスに接続する

特定の DB エンジンを実行している DB インスタンスへの接続については、DB エンジンの指示に従ってください。

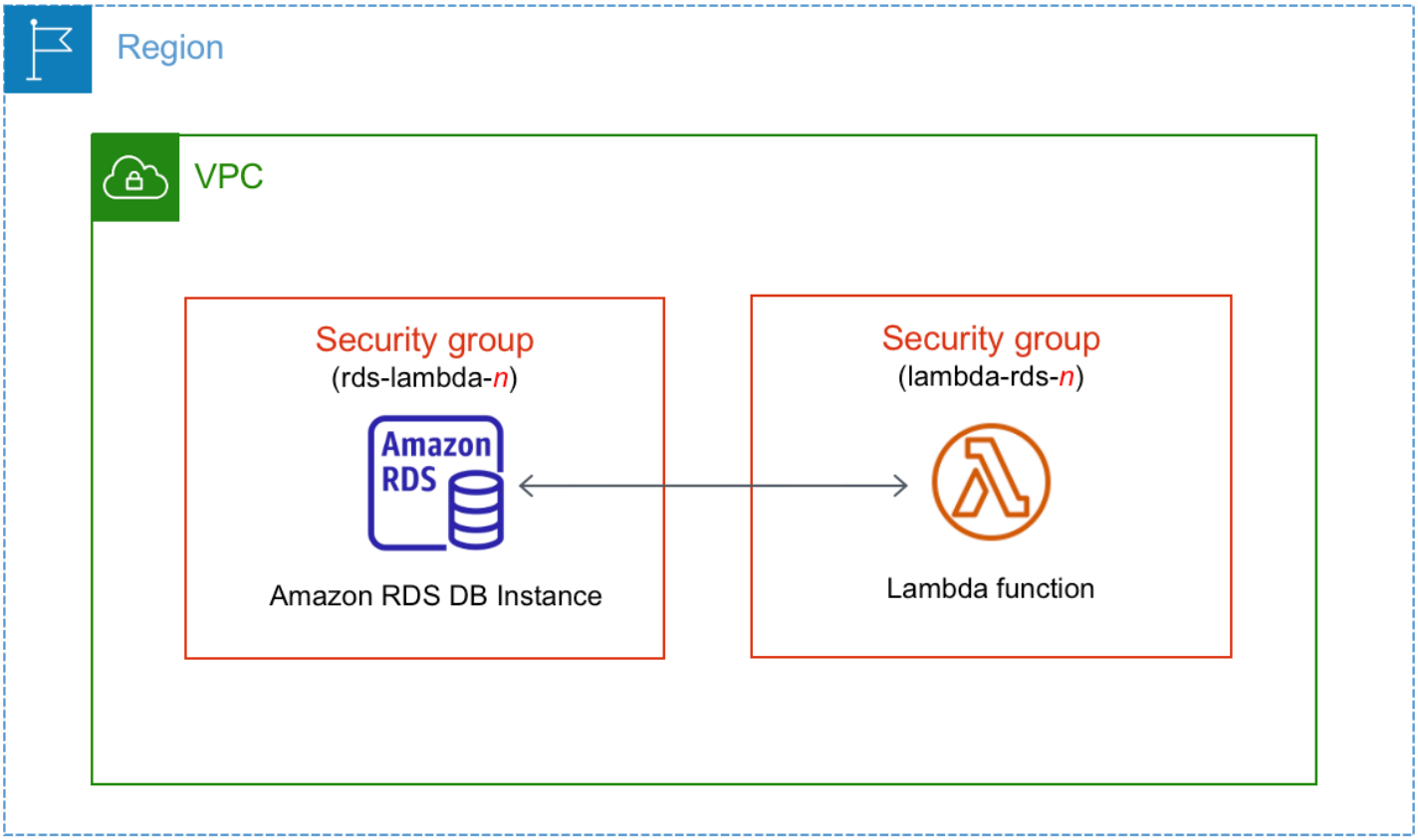
- MariaDB DB インスタンスへの接続
- SQL Server DB インスタンスへの接続
- MySQL DB インスタンスへの接続
- Oracle DB インスタンスへの接続
- PostgreSQL データベースエンジンを実行する DB インスタンスへの接続

Lambda 関数と DB インスタンスを自動的に接続する

Amazon RDS コンソールを使用すると、Lambda 関数と DB インスタンスとの接続を簡単に設定できます。多くの場合、DB インスタンスは VPC 内のプライベートサブネットにあります。アプリケーションで Lambda 関数を使用すると、プライベート DB インスタンスにアクセスできます。

Lambda 関数とマルチ AZ DB クラスター間の接続をセットアップする方法については、「the section called “Lambda 関数とマルチ AZ DB クラスターを接続する”」を参照してください。

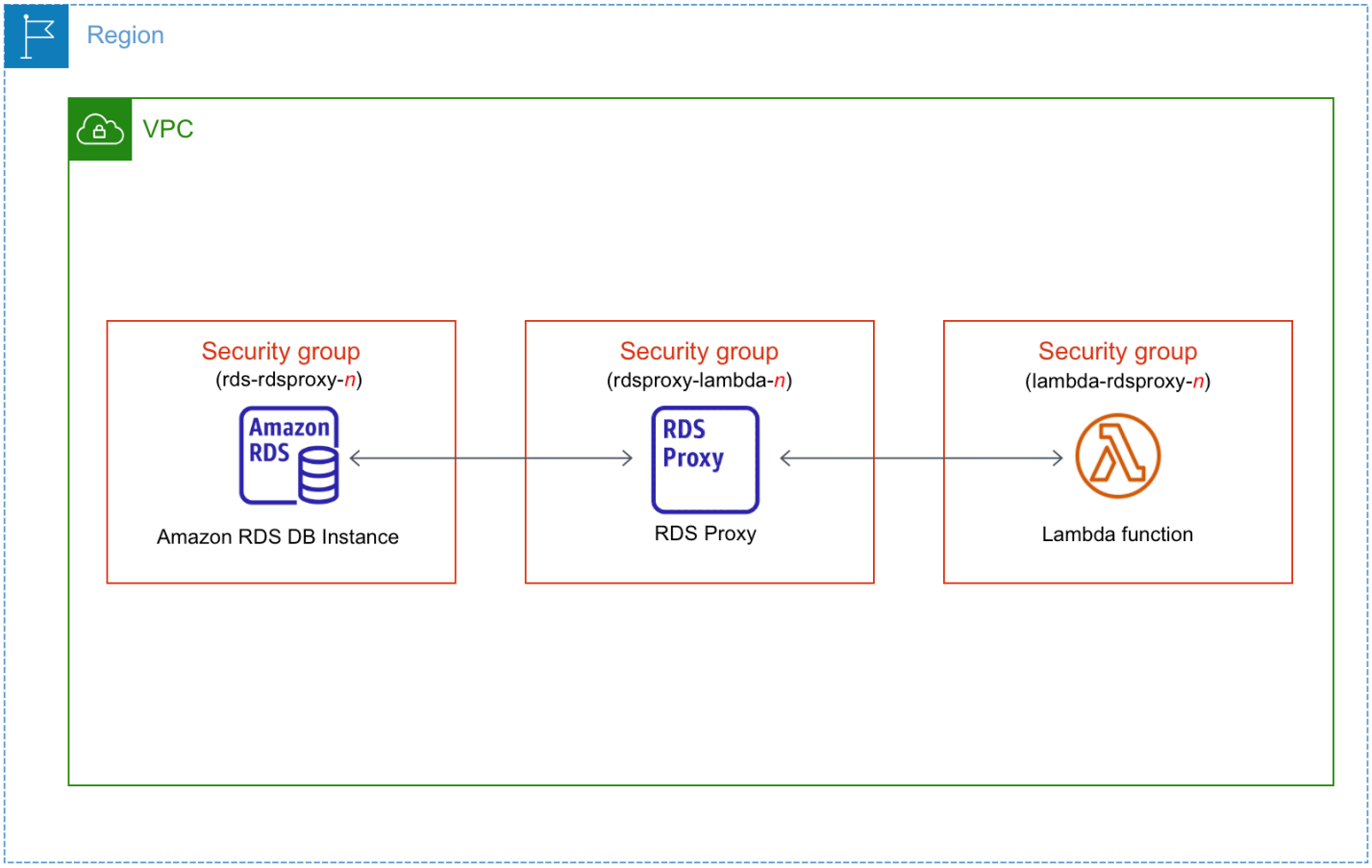
次の画像は、DB インスタンスと Lambda 関数の間の直接接続を示しています。



Lambda 関数と DB インスタンス間の RDS プロキシ経由の接続を設定して、データベースのパフォーマンスと耐障害性を改善できます。多くの場合、Lambda 関数は短いデータベース接続を頻繁に行い、RDS プロキシが提供する接続プールを使用することで利点を得られます。データベース認証情報を Lambda アプリケーションコードで管理する代わりに、Lambda 関数に設定済みの AWS Identity and Access Management IAM 認証を利用できます。詳細については、「Amazon RDS Proxy」を参照してください。

コンソールを使用して既存のプロキシに接続すると、Amazon RDS は、DB インスタンスと Lambda 関数からの接続を許可するように、プロキシセキュリティグループを更新します。

同じコンソールページから新しいプロキシを作成することもできます。コンソールでプロキシを作成するときに、DB インスタンスにアクセスするには、データベースの認証情報を入力するか、AWS Secrets Manager シークレットを選択する必要があります。



Tip

Lambda 関数を DB インスタンスにすばやく接続するには、コンソール内のガイド付きウィザードを使用することもできます。ウィザードを開くには、次の操作を行います。

1. Lambda コンソールの [関数] ページを開きます。
2. データベースを接続する関数を選択します。
3. [設定] タブで [RDS データベース] を選択します。
4. [RDS データベースに接続] を選択します。

関数をデータベースに接続したら、[プロキシの追加] を選択してプロキシを作成できます。

トピック

- Lambda 関数との自動接続の概要
- Lambda 関数と RDS データベース を自動的に接続する
- 接続中のコンピューティングリソースを表示する

Lambda 関数との自動接続の概要

Lambda 関数と RDS DB インスタンス を接続するための要件は次のとおりです。

- Lambda 関数は、DB インスタンスと同じ VPC に存在する必要があります。
- 接続を設定するユーザーには、以下の Amazon RDS、Amazon EC2、Lambda、Secrets Manager、および IAM 操作を実行するアクセス許可が必要です。
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBInstances`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBInstance`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`
    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
  - Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
  - Secrets Manager

    - `secretsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
- IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
- AWS KMS
    - `kms:describeKey`

> **Note**
>
> DB インスタンスと Lambda 関数が異なるアベイラビリティーゾーンにある場合、アベイラビリティーゾーン間のコストが発生する可能性があります。

Lambda 関数 RDS データベース間の接続を設定すると、Amazon RDS は、関数と DB インスタンスの VPC セキュリティグループを設定します。RDS プロキシを使用する場合、Amazon RDS はプロキシの VPC セキュリティグループも設定します。Amazon RDS は、次の表で説明されているように、DB インスタンス、Lambda 関数、およびプロキシに関連付けられたセキュリティグループの現在の設定に従って動作します。

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
DB インスタンスには 1 つ以上のセキュリティグループが関連付けられ、その名前は、パターン `rds-lambda-` *n* に一致します。または、プロキシが既に DB インスタンスに接続している場合、RDS	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* (*n* は数字) に一致します。	プロキシに関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `rdsproxy-lambda-` *n* (*n* は数字) に一致します。 パターンに一致するセキュリティグルー	Amazon RDS は何もしません。 Lambda 関数、プロキシ (オプション)、および DB インスタンス間の接続は既に自動的に設定されています。関数、プロキシ、およびデータベー

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
は関連するプロキシの TargetHealth が AVAILABLE であるかどうかを確認します。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、DB インスタンスまたはプロキシのいずれかの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つだけあります。	プは変更されていません。このセキュリティグループには、Lambda 関数と DB インスタンス の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。	スの間には既に接続が存在するため、セキュリティグループは変更されません。

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
次の条件のいずれかが適用されます。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つ 1 つ以上のセキュリティグループがあります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。ただし、これらのセキュリティグループは、いずれも Lambda 関数との接続には使用できません。	次の条件のいずれかが適用されます。 • Lambda 関数に関連付けられ、パターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致する名前のセキュリティグループはありません。 • Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、DB インスタンスとの接続には使用できません。 Amazon RDS は、DB インスタンスまたは	次の条件のいずれかが適用されます。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループはありません。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。ただし、Amazon RDS は、これらのセキュリティグループのいずれも、DB インスタンスまたは Lambda 関数との接続には使用できません。 Amazon RDS は、DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウト	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
Amazon RDS は、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。変更の例としては、ルールの追加や、既存ルールのポート変更などがあります。	プロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	バウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つ 1 つ以上のセキュリティグループがあります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。 ただし、Amazon RDS は、これらのセキュリティグループのいずれも、DB インスタンスとの接続には使用できません。Amazon RDS は、DB インスタンスまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。 ただし、Amazon RDS は、これらのセキュリティグループのいずれも、DB インスタンスまたは Lambda 関数との接続には使用できません。Amazon RDS は、DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つ 1 つ以上のセキュリティグループがあります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	接続用の有効な Lambda セキュリティグループが存在しますが、Lambda 関数に関連付けられていません。このセキュリティグループには、パターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致する名前が付いています。このセキュリティグループには変更が加えられていません。DB インスタンスまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つだけあります。	接続に有効なプロキシセキュリティグループは存在しますが、プロキシに関連付けられていません。このセキュリティグループには、パターン `rdsproxy-lambda-` *n* に一致する名前が付いています。このセキュリティグループには変更が加えられていません。DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。	RDS action: associate Lambda security group

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
次の条件のいずれかが適用されます。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つ 1 つ以上のセキュリティグループがあります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。ただし、Amazon RDS は、これらのセキュリティグループを Lambda 関数またはプロキシと	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、DB インスタンスまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つだけあります。	プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
の接続に使用できません。 Amazon RDS は、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。			

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
次の条件のいずれかが適用されます。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 • DB インスタンスに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つ 1 つ以上のセキュリティグループがあります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。ただし、Amazon RDS は、これらのセキュリティグループを Lambda 関数またはプロキシと	次の条件のいずれかが適用されます。 • Lambda 関数に関連付けられ、パターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致する名前のセキュリティグループはありません。 • Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。ただし、Amazon RDS は、これらのセキュリティグループを DB インスタンスとの接続に使用できません。 Amazon RDS は、DB インスタンスまたはプロキシの VPC セ	次の条件のいずれかが適用されます。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループはありません。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。ただし、Amazon RDS は、これらのセキュリティグループを DB インスタンスまたは Lambda 関数との接続に使用することはできません。 Amazon RDS は、DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウト	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
の接続に使用できません。 Amazon RDS は、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	キュリティグループをソースとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	バウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	

RDS アクション: 新しいセキュリティグループを作成する

Amazon RDS は以下のアクションを実行します。

- パターン `rds-lambda-`*n* または `rds-rdsproxy-`*n* (RDS プロキシを使用する場合) に一致する新しいセキュリティグループを作成します。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールがあります。このセキュリティグループは、DB インスタンスに関連付けられ、関数またはプロキシが DB インスタンスにアクセスすることを許可します。
- パターン `lambda-rds-`*n* または `lambda-rdsproxy-`*n* に一致する新しいセキュリティグループを作成します。このセキュリティグループには、DB インスタンスまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールがあります。このセキュリティグループは Lambda 関数に関連付けられ、関数が DB インスタンス にトラフィックを送信するか、プロキシ経由でトラフィックを送信することを許可します。

- パターン `rdsproxy-lambda-`*n* に一致する新しいセキュリティグループを作成します。このセキュリティグループには、DB インスタンスおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。

RDS アクション: Lambda セキュリティグループを関連付ける

Amazon RDS は、有効な既存の Lambda セキュリティグループを Lambda 関数に関連付けます。このセキュリティグループは、関数が DB インスタンス にトラフィックを送信するか、プロキシ経由でトラフィックを送信することを許可します。

Lambda 関数と RDS データベース を自動的に接続する

Amazon RDS コンソールを使用して、Lambda 関数を DB インスタンスに自動的に接続することができます。これにより、これらのリソース間の接続を設定するプロセスが簡単になります。

RDS プロキシを使用して、接続にプロキシを含めることもできます。Lambda 関数は短いデータベース接続を頻繁に行うため、RDS プロキシが提供する接続プールを使用することで利点を得られます。Lambda アプリケーションコードでデータベース認証情報を管理する代わりに、Lambda 関数用に設定済みの IAM 認証を使用することもできます。

[Lambda 接続の設定] ページを使用して、既存の DB インスタンスを新規および既存の Lambda 関数に接続できます。セットアッププロセスでは、必要なセキュリティグループが自動的にセットアップされます。

Lambda 関数と DB インスタンスの間の接続を設定する前に、次のことを確認してください。

- Lambda 関数と DB インスタンスが同じ VPC にあります。
- ユーザーアカウントに適切なアクセス許可があります。要件の詳細については、「Lambda 関数との自動接続の概要」を参照してください。

接続の設定後にセキュリティグループを変更すると、Lambda 関数と DB インスタンスとの接続に影響する可能性があります。

> **Note**
>
> AWS マネジメントコンソール でのみ、DB インスタンスと Lambda 関数の間の接続を自動的に設定できます。Lambda 関数を接続するには、DB インスタンスが使用可能状態である必要があります。

Lambda 関数と DB インスタンスを自動的に接続するには

<result>

設定を確認すると、Amazon RDS は、Lambda 関数、RDS プロキシ (プロキシを使用した場合)、および DB インスタンスの接続プロセスを開始します。コンソールに [接続の詳細] ダイアログボックスが表示され、リソース間の接続を許可するセキュリティグループの変更が一覧表示されます。
</result>

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、Lambda 関数に接続するDB インスタンスを選択します。
3. [アクション] として、[Lambda 接続の設定] を選択します。
4. [Lambda 接続の設定] ページの [Lambda 関数の選択] で、次のいずれかを実行します。
   - DB インスタンスと同じ VPC に既存の Lambda 関数がある場合は、[既存の関数を選択] を選択し、関数を選択します。
   - 同じ VPC に Lambda 関数がない場合は、[新しい関数を作成] を選択し、[関数名] を入力します。デフォルトのランタイムは Nodejs.18 に設定されています。接続設定が完了した後、Lambda コンソールで新しい Lambda 関数の設定を変更できます。
5. (オプション) [RDS プロキシ] で、[RDS プロキシを使用して接続] を選択し、次のいずれかを実行します。
   - 使用する既存のプロキシがある場合は、[既存のプロキシを選択] を選択し、プロキシを選択します。
   - プロキシがなく、Amazon RDS にプロキシを自動的に作成させる場合は、[新しいプロキシの作成] を選択します。次に、[データベース認証情報] として、次のいずれかを実行します。
     a. [データベースのユーザー名とパスワード] を選択し、DB インスタンスの [ユーザー名] と [パスワード] を入力します。
     b. [Secrets Manager シークレット] を選択します。次に、[シークレットを選択] で、AWS Secrets Manager シークレットを選択します。Secrets Manager シークレットがない場合は、[新しい Secrets Manager シークレットを作成] を選択して、新しいシークレットを作成します。シークレットを作成した後、[シークレットを選択] で、新しいシークレットを選択します。

新しいプロキシを作成した後、[既存のプロキシを選択] を選択し、プロキシを選択します。プロキシが接続可能になるまでに時間がかかる場合があることに注意してください。

6. (オプション) [接続の概要] を展開し、強調表示されているリソースの最新情報を確認します。
7. [設定] を選択します。

接続中のコンピューティングリソースを表示する

AWS マネジメントコンソール を使用して、DB インスタンスに接続されている Lambda 関数を確認できます。表示されるリソースには、Amazon RDS が自動的に設定したコンピューティングリソース接続が含まれます。

一覧表示されるコンピューティングリソースには、DB インスタンスに手動で接続されたリソースは含まれていません。例えば、データベースに関連付けられた VPC セキュリティグループにルールを追加することで、コンピューティングリソースが DB インスタンス に手動でアクセスするのを許可できます。

コンソールに Lambda 関数を一覧表示するには、以下の条件が適用される必要があります。

- コンピューティングリソースに関連付けられているセキュリティグループの名前がパターン `lambda-rds-`*n* または `lambda-rdsproxy-`*n* (*n* は数字) と一致します。
- コンピューティングリソースに関連付けられているセキュリティグループに、DB インスタンスまたは該当するプロキシが使用するポートにポート範囲が設定されたアウトバウンドルールがあります。アウトバウンドルールのデスティネーションは、DB インスタンスまたは関連するプロキシに関連付けられているセキュリティグループに設定されている必要があります。
- 構成にプロキシが含まれる場合、データベースに関連付けられたプロキシにアタッチされたセキュリティグループの名前は、パターン `rdsproxy-lambda-`*n* (*n* は数字) と一致します。
- 関数に関連付けられているセキュリティグループに、DB インスタンスまたは該当するプロキシが使用するポートにポート範囲が設定されたアウトバウンドルールがあります。デスティネーションは、DB インスタンスまたは関連するプロキシに関連付けられているセキュリティグループに設定されている必要があります。

DB インスタンスに自動的に接続されたコンピューティングリソースを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[データベース] を選択し、DB クラスターを選択します。
3. [接続とセキュリティ] タブの [接続されたコンピューティングリソース] にコンピューティングリソースが表示されます。

Amazon RDS DB インスタンスを変更する

ストレージの追加や DB インスタンスクラスの変更などのタスクを完了するために、DB インスタンスの設定を変更できます。このトピックでは、Amazon RDS DB インスタンスを変更する方法と、DB インスタンスの設定について説明します。

本稼働インスタンスの修正前に、各変更の影響を完全に把握できるように、テストインスタンスでの変更のテストをお勧めします。これにより、各変更の影響を完全に理解できます。このようなテストは特に、データベースのバージョンをアップグレードするときに重要です。

DB インスタンスに対するほとんどの変更は、すぐに適用することも、次のメンテナンスウィンドウまで延期することもできます。一部の変更 (パラメータグループの変更など) を適用するには、手動による DB インスタンスの再起動が必要になる場合があります。

> **Important**
>
> また、一部の変更を適用するために Amazon RDS で DB インスタンスを再起動する必要があるため、ダウンタイムが発生する場合もあります。DB インスタンスの設定を変更する前に、データベースとアプリケーションに対する影響を考慮してください。

コンソール

DB インスタンスを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] (変更) を選択します。Modify DB instance ページが表示されます。
4. 必要に応じて任意の設定を変更してください。各設定の詳細については、「DB インスタンスの設定」を参照してください。
5. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
6. (省略可能) 変更をすぐに適用するには、[すぐに適用] を選択します。このオプションを選択すると、ダウンタイムを発生させる場合があります。詳細については、「スケジュール変更設定の使用」を参照してください。
7. 確認ページで、変更内容を確認します。正しい場合は、[Modify DB Instance (DB インスタンスを変更)] を選択して変更を保存します。

または、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更をキャンセルします。

AWS CLI

AWS CLI を使用して DB インスタンスを変更するには、modify-db-instance コマンドを呼び出します。DB インスタンス識別子と、変更するオプションの値を指定します。各オプションの詳細については、「DB インスタンスの設定」を参照してください。

Example

次のコードでは、バックアップ保存期間を 1 週間 (7 日間) に設定して、mydbinstance を変更します。このコードは、--deletion-protection を使用して削除保護を有効にします。削除保護を無効にするには、--no-deletion-protection を使用します。変更は、--no-apply-immediately を使用して次のメンテナンスウィンドウ中に適用されます。今すぐ変更を適用するには、--apply-immediately を使用します。詳細については、「スケジュール変更設定の使用」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
    --deletion-protection \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --deletion-protection ^
    --no-apply-immediately
```

Example

次の例では、RDS for Oracle DB インスタンスにストレージボリュームを追加します。追加ボリュームは、割り当て済みストレージが 5000 GiB、12000 IOPS の gp3 ストレージを使用します。

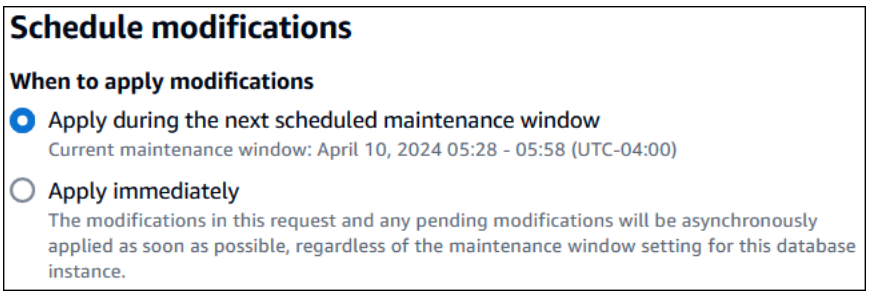
```
aws rds modify-db-instance \
     --db-instance-identifier my-oracle-instance \
     --additional-storage-volumes '[{ \
             "VolumeName": "rdsdbdata2", \
             "StorageType": "gp3", 
             "AllocatedStorage": 5000, \
             "IOPS": 12000 \
         }]'
```

RDS API

Amazon RDS API を使用して DB インスタンスを変更するには、ModifyDBInstance オペレーションを呼び出します。DB インスタンス識別子と、変更する設定のパラメータを指定します。各パラメータの詳細については、「DB インスタンスの設定」を参照してください。

スケジュール変更設定の使用

DB インスタンスを変更するときは、変更を行うタイミングを決定します。



次回のメンテナンス中ではなく、ただちに変更を適用する場合は、AWS マネジメントコンソールの [すぐに適用] オプションを選択します。または、AWS CLI を呼び出す際に `--apply-immediately` パラメータを使用するか、Amazon RDS API を使用する際に `ApplyImmediately` パラメータを `true` に設定します。

変更の即時適用を選択しない場合、RDS は、この変更を保留中の変更キューに配置します。次回のメンテナンスウィンドウで、RDS は、キュー内のすべての保留中の変更を適用します。変更の即時適用を選択した場合は、新しい変更と、保留中の変更キューにあるすべての変更が適用されます。

次のメンテナンスウィンドウに向けて保留中の変更を確認するには、describe-db-instances AWS CLI コマンドを使用して `PendingModifiedValues` フィールドを確認します。

> **Important**
>
> DB インスタンスを一時的に使用できないようにすること (ダウンタイム) を要する保留中の変更がある場合、[Apply Immediately (すぐに適用)] オプションを選択すると、予想外のダウンタイムが発生することがあります。
> 変更をすぐに適用することを選択した場合、保留中の変更も、次のメンテナンス時間中ではなく、すぐに適用されます。
> 次のメンテナンスウィンドウで保留中の変更を適用しない場合は、変更を元に戻すように DB インスタンスを変更できます。これを行うには、AWS CLI を使用し、`--apply-immediately` オプションを指定します。
> `modify-db-instance` コマンドを使用してストレージボリュームを追加すると、RDS は `--no-apply-immediately` パラメータに関係なくストレージボリュームをすぐに追加します。リクエストに他の変更がある場合は、スケジュールの変更に基づいて適用されます。

変更の延期を選択した場合でも、一部のデータベース設定に対する変更は即時に適用されます。さまざまなデータベース設定が [Apply immediately (すぐに適用)] の設定とどのように相互作用するかについては、「DB インスタンスの設定」を参照してください。

DB インスタンスの設定

次のテーブルは、変更できる設定と変更できない設定の詳細を示しています。また、変更を適用できるタイミングや、変更によって DB インスタンスのダウンタイムが発生するかどうかを確認することもできます。マルチ AZ などの Amazon RDS の機能を使用すると、後で DB インスタンスを変更した場合にダウンタイムを最小限に抑えることができます。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

コンソール、CLI の `modify-db-instance` コマンド、または RDS API の `ModifyDBInstance` オペレーションを使用して DB インスタンスを変更できます。

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ストレージ割り当て
 	
DB インスタンスに割り当てるストレージ (ギビバイト単位)。割り当てたストレージは増やすことだけができます。割り当てられたストレージを減らすことはできません。
 	
古い DB インスタンスのストレージ、または古い DB スナップショットから復元された DB インスタンスのストレージは変更できない場合があります。[Allocated Storage (割り当てられたストレージ)] 設定は、DB インスタンスが対象にならない場合、コンソールで無効になります。CLI コマンド describe-valid-db-instance-modifications を使用して、より多くのストレージを割り当てることができるかどうかを確認できます。このコマンドは、DB インスタンスの有効なストレージオプションを返します。
 	
DB インスタンスのステータスが [storage-optimization] (スト
	CLI オプション:
	
	`--allocated-storage`
	
	RDS API パラメータ:
	
	`AllocatedStorage`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。パフォーマンスは変更時に低下する可能性があります。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
レージの最適化) に設定されていると、割り当てられたストレージを変更することはできません。DB インスタンスでは、任意の 24 時間以内に最大 4 回のストレージ変更を実行できます。ボリュームごとの最適化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。 許容される最大ストレージは、DB エンジンとストレージタイプによって異なります。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。				

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
アーキテクチャ設定 DB インスタンスに複数のテナントデータベースを配置できるようにする設定。現在、この設定をサポートしているのは RDS for Oracle コンテナデータベース (CDB) のみです。 CDB がシングルテナント設定の場合は、マルチテナント設定を使用するように変更できます。この設定では、RDS API を使用して、データベースのエディションと必要なオプションライセンスに応じて、1 ~ 30 個のテナントデータベースを作成できます。アプリケーション PDB とプロキシ PDB はサポートされていません。マルチテナント設定は永続的です。つまり、後で CDB をシングルテナント設定に戻すことはできません。 Note Amazon RDS 設定は、単なる Oracle DB エン	CLI オプション: `--multi-tenant` (CDB アーキテクチャのマルチテナント設定) `--no-multi-tenant` (CDB アーキテクチャのシングルテナント設定) API パラメータ: `MultiTenant`	変更はただちに発生します。	この変更時にダウンタイムは発生しません。	Oracle

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
ジンではなく、Amazon RDS の機能であるため、「multitenant」ではなく「multi-tenant」と呼ばれています。同様に、RDS の用語「テナント」は、Oracle PDB だけでなく、RDS 設定のすべてのテナントを指します。RDS のドキュメントの「Oracle マルチテナント」という用語は、オンプレミスデプロイと RDS デプロイの両方に対応する Oracle データベース CDB アーキテクチャのみを指します。 詳細については、「RDS for Oracle CDB の概要」を参照してください。				

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
アーキテクチャ設定 Oracle データベースのアーキテクチャ: CDB または非 CDB。[Oracle マルチテナントアーキテクチャ] を選択した場合、RDS for Oracle は非 CDB を、シングルテナント設定を使用する CDB に変換します。 この設定は、ご使用のデータベースが、April 2021 以降の RU で Oracle Database 19c を実行している非 CDB である場合にのみサポートされます。変換後、CDB には 1 つの初期プラグイン可能データベース (PDB) が含まれます。アーキテクチャの変更は永続的です。つまり、CDB を非 CDB に戻すことはできません。 Note シングルテナント設定の CDB をマルチテナント設定に変換するには、CDB インスタンスを再度変更し、アーキテクチャ設定	CLI オプション: `--engine oracle-ee-cdb` (Oracle マルチテナント) `--engine oracle-se2-cdb` (Oracle マルチテナント) API パラメータ: `Engine`	変更をすぐに適用するように選択した場合は、直ちに適用されます。 変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。	この変更中に、ダウンタイムが発生します。	Oracle

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
にマルチテナント設定を選択します。 詳細については、「CDB アーキテクチャのシングルテナント設定」を参照してください。				

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

マイナーバージョン自動アップグレード
 	
[マイナーバージョン自動アップグレードの有効化] を選択すると、希望する DB エンジンのマイナーバージョンのアップグレードをリリースと同時に自動的に DB インスタンスに適用できます。これがデフォルトの動作です。Amazon RDS では、メンテナンスウィンドウでマイナーバージョンの自動アップグレードが実行されます。[マイナーバージョン自動アップグレードの有効化] を選択しなかった場合、新しいマイナーバージョンが利用可能になっても DB インスタンスは自動的にアップグレードされません。
 	
詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。
	CLI オプション:
	
	`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`
	
	RDS API パラメータ:
	
	`AutoMinorVersionUpgrade`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
バックアップレプリケーション [別の AWS リージョンへのレプリケーションを有効にする] を選択して、災害復旧用の追加リージョンにバックアップを作成します。 次に、追加バックアップ先リージョンを選択します。	DB インスタンスの変更時には使用できません。AWS CLI または RDS API でクロスリージョンバックアップを有効にする方法については、「Amazon RDS のクロスリージョン自動バックアップの有効化」を参照してください。	変更は、可能な限り早く非同期的に適用されます。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされているDB エンジン

バックアップの保存期間
 	
自動バックアップが保持される日数。自動バックアップを無効にするには、バックアップ保持期間を 0 に設定します。
 	
詳細については、「バックアップの概要」を参照してください。
 	
Note
AWS Backup を使用してバックアップを管理する場合、このオプションは適用されません。AWS Backup については、「AWS Backup デベロッパーガイド」を参照してください。
	CLI オプション:
	
	`--backup-retention-period`
	
	RDS API パラメータ:
	
	`BackupRetentionPeriod`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しないで、設定を 0 以外の値から別の 0 以外の値に変更した場合、変更は可能な限り早く非同期的に適用されます。そうでない場合、変更は次のメンテナンス時間中に行われます。
			0 から 0 以外の値、0 以外の値から 0 に変更した場合、ダウンタイムが発生します。
			
			これは、シングル AZ とマルチ AZ DB インスタンスの両方に適用されます。
				すべてのDB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

バックアップウィンドウ
 	
データベースの自動バックアップが実行される期間。バックアップウィンドウは、協定世界時 (UTC) の開始時間で、時間単位での実行期間です。
 	
詳細については、「バックアップの概要」を参照してください。
 	
Note
AWS Backup を使用してバックアップを管理する場合、このオプションは表示されません。AWS Backup の詳細については、「AWS Backup デベロッパーガイド」を参照してください。
	CLI オプション:
	
	`--preferred-backup-window`
	
	RDS API パラメータ:
	
	`PreferredBackupWindow`
		変更は、可能な限り早く非同期的に適用されます。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

認証局
 	
DB インスタンスによって使用されるサーバー証明書の認定機関 (CA)。
 	
詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
	CLI オプション:
	
	`--ca-certificate-identifier`
	
	RDS API パラメータ:
	
	`CACertificateIdentifier`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			ダウンタイムは、DB エンジンが再起動なしのローテーションをサポートしていない場合にのみ発生します。describe-db-engine-versions AWS CLI コマンドを使用すると、DB エンジンが再起動なしのローテーションをサポートしているかどうかを判断できます。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

Copy tags to snapshots
 	
DB インスタンスタグがある場合、このオプションはを有効にすると DB スナップショットを作成する際にそれらをコピーすることができます。
 	
詳細については、「Amazon RDS リソースのタグ付け」を参照してください。
	CLI オプション:
	
	`--copy-tags-to-snapshot` 、、または `--no-copy-tags-to-snapshot`
	
	RDS API パラメータ:
	
	`CopyTagsToSnapshot`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

データベースポート
 	
DB インスタンスへのアクセスに使用するポート。
 	
ポート値は、DB インスタンスに関連するオプショングループのオプションに指定されているポート値と一致しないようにしてください。
 	
詳細については、「Amazon RDS DB インスタンスへの接続」を参照してください。
	CLI オプション:
	
	`--db-port-number`
	
	RDS API パラメータ:
	
	`DBPortNumber`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	DB インスタンスはすぐに再起動されます。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

DB エンジンバージョン
 	
使用する DB エンジンのバージョン。本番稼働 DB インスタンスをアップグレードする前に、テスト DB インスタンスでアップグレードプロセスをテストすることをお勧めします。これを行うと、その期間の検証とアプリケーションの検証に役立ちます。
 	
詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。
	CLI オプション:
	
	`--engine-version`
	
	RDS API パラメータ:
	
	`EngineVersion`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中に、ダウンタイムが発生します。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

DB インスタンスクラス
 	
使用する DB インスタンスクラス。
 	
詳細については、「DB インスタンスクラス」を参照してください。
	CLI オプション:
	
	`--db-instance-class`
	
	RDS API パラメータ:
	
	`DBInstanceClass`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中に、ダウンタイムが発生します。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

DB インスタンス識別子:
 	
新しい DB インスタンス識別子。この値は小文字で保存されます。
 	
DB インスタンスの名前の変更に伴う影響の詳細については、「DB インスタンスの名前を変更する」を参照してください。
	CLI オプション:
	
	`--new-db-instance-identifier`
	
	RDS API パラメータ:
	
	`NewDBInstanceIdentifier`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			DB エンジンのバージョンが動的 SSL アップロードをサポートしていない限り、この変更中にダウンタイムが発生します。バージョンが再起動を必要とするかどうかを判断するには、次の AWS CLI コマンドを実行します。
			
			`aws rds describe-db-engine-versions \ --default-only \ --engine your-db-engine \`
				すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
			`--query 'DBEngineVersions[*].SupportsCertificateRotationWithoutRestart'`	

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

DB パラメータ グループ
 	
DB インスタンスに関連付ける DB パラメータグループ。
 	
詳細については、「Amazon RDS のパラメータグループ」を参照してください。
	CLI オプション:
	
	`--db-parameter-group-name`
	
	RDS API パラメータ:
	
	`DBParameterGroupName`
		新しい DB パラメータグループと DB インスタンスの関連付けはすぐに行われます。	新しい DB パラメータグループを DB インスタンスに関連付けると、ダウンタイムは発生しません。
			
			DB パラメータグループの関連付けは、パラメータグループ内のパラメータ変更の適用とは異なります。RDS は、DB インスタンスを手動で再起動した後にのみ、変更された静的パラメータと動的パラメータを新しく関連付けられたパラメータグルー
				すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
			プに適用します。ただし、DB インスタンスに関連付けた後に DB パラメータグループの動的パラメータを変更すると、これらの変更は再起動せずに直ちに適用されます。 詳細については、「Amazon RDS のパラメータグループ」および「DB インスタンスの再起動」を参照してください。	

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

専用ログボリューム
 	
専用ログボリューム (DLV) を使用して、データベーステーブルを含むボリュームとは別のストレージボリュームにデータベーストランザクションログを保存します。
 	
詳細については、「専用ログボリューム (DLV) を使用する」を参照してください。
	CLI オプション:
	
	`-dedicated-log-volume`
	
	RDS API パラメータ:
	
	`DedicatedLogVolume`
		変更は、DB インスタンスの再起動後に適用されます。	DB インスタンスの再起動中にダウンタイムが発生します。	MariaDB、MySQL、PostgreSQL

削除保護
 	
DB インスタンスが削除されないようにするには [Enable deletion protection (削除保護の有効化)] を選択します。
 	
詳細については、「DB インスタンスを削除する」を参照してください。
	CLI オプション:
	
	`--deletion-protection|--no-deletion-protection`
	
	RDS API パラメータ:
	
	`DeletionProtection`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

拡張モニタリング
 	
[拡張モニタリングを有効にする] を選択すると、DB インスタンスが実行されているオペレーティングシステムに対してリアルタイムでのメトリクスの収集が有効になります。
 	
詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。
	CLI オプション:
	
	`--monitoring-interval` 、、および `--monitoring-role-arn`
	
	RDS API パラメータ:
	
	`MonitoringInterval` 、、および `MonitoringRoleArn`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされているDB エンジン

IAM DB authentication
 	
[Enable IAM DB authentication] (IAM DB 認証を有効にする)、ユーザーとロールを通じてデータベースユーザーを認証します。
 	
詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。
	CLI オプション:
	
	`--enable-iam-database-authentication|--no-enable-iam-database-authentication`
	
	RDS API パラメータ:
	
	`EnableIAMDatabaseAuthentication`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。	MariaDB、MySQL、PostgreSQL のみ

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

Kerberos 認証
 	
DB インスタンスの移動先となる Active Directory を選択します。この操作の前にディレクトリが存在する必要があります。ディレクトリが既に選択されている場合は、[None (なし)] を指定して DB インスタンスを現在のディレクトリから削除できます。
 	
詳細については、「Kerberos 認証」を参照してください。
	CLI オプション:
	
	`--domain`、、および `--domain-iam-role-name`
	
	RDS API パラメータ:
	
	`Domain`、、および `DomainIAMRoleName`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中は、短いダウンタイムが発生します。	Microsoft SQL Server、MySQL、Oracle、PostgreSQL のみ

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ライセンスモデル
 	
Db2 および Oracle のライセンスを使用するには、[bring-your-own-license] を選択します。
 	
Microsoft SQL Server の一般ライセンス契約が使用するには、[license-included] を選択します。
 	
詳細については、Amazon RDS for Db2 のライセンスオプション、Amazon RDS での Microsoft SQL Server のライセンス、および RDS for Oracle のライセンスオプション を参照してください。
	CLI オプション:
	
	`--license-model`
	
	RDS API パラメータ:
	
	`LicenseModel`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中に、ダウンタイムが発生します。	Microsoft SQL Server と Oracle のみ

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ログのエクスポート
 	
Amazon CloudWatch Logs に発行するデータベースログファイルのタイプ。
 	
詳細については、「Amazon CloudWatch Logs へのデータベースログの発行」を参照してください。
	CLI オプション:
	
	`--cloudwatch-logs-export-configuration`
	
	RDS API パラメータ:
	
	`CloudwatchLogsExportConfiguration`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

メンテナンスウィンドウ
 	
システムメンテナンスを実行する時間帯。該当する場合は、システムメンテナンスにはアップグレードが含まれます。メンテナンス時間は、協定世界時 (UTC) の開始時間で、時間単位での実行期間です。
 	
そのウィンドウを現在の時刻に設定した場合、現在の時刻からウィンドウの終わりまで 30 分以上必要です。このタイミングにより、保留中の変更が確実に適用されるようになります。
 	
詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。
	CLI オプション:
	
	`--preferred-maintenance-window`
	
	RDS API パラメータ:
	
	`PreferredMaintenanceWindow`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	ダウンタイムを引き起こす保留中のアクションが 1 つ以上あり、現在の時刻を含むようにメンテナンス時間を変更した場合、それらの保留中のアクションはすぐに適用され、ダウンタイムが発生します。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

でマスター認証情報を管理する AWS Secrets Manager
 	
[Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択して、Secrets Manager でユーザーのパスワードをシークレットに管理します。
 	
オプションで、シークレットを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
 	
RDS によって既に DB インスタンスのマスターユーザーのパスワードを管理している場合は、Rotate secret immediately (すぐにシークレットをローテーションする) を選択してマスターユーザーパスワードをすぐにローテーションできます。
 	
詳細については、「Amazon RDS および AWS Secrets Manager に
	CLI オプション:
	
	`--manage-master-user-password | --no-manage-master-user-password`
	
	`--master-user-secret-kms-key-id`
	
	`--rotate-master-user-password | --no-rotate-master-user-password`
	
	RDS API パラメータ:
		マスターユーザーパスワードの自動管理をオンまたはオフにすると、すぐに変更が行われます。この変更は、[Apply immediately] (すぐに適用) の設定を無視します。
		
		マスターユーザーのパスワードをローテーションする場合は、変更がすぐに適用されるように指定する必要があります。
			この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

よるパスワード管理」を参照してください。	`ManageMasterUserPassword`
	
	`MasterUserSecretKmsKeyId`
	
	`RotateMasterUserPassword`			

マルチ AZ 配置
 	
複数のアベイラビリティーゾーンにある DB インスタンスをデプロイする場合、[はい] をクリックします。それ以外の場合は、[いいえ] をクリックします。
 	
詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。
	CLI オプション:
	
	`--multi-az|--no-multi-az`
	
	RDS API パラメータ:
	
	`MultiAZ`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。ただし、パフォーマンスに影響する可能性があります。詳細については、「Amazon RDS の DB インスタンスをマルチ AZ 配置に変換する」を参照してください。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ネットワークの種類
 	
DB インスタンスでサポートされている IP アドレス設定プロトコル。
 	
リソースが、インターネットプロトコルバージョン 4 (IPv4) アドレス設定プロトコル経由でのみ DB インスタンスと通信できるように指定する IPv4。
 	
リソースが IPv4、インターネットプロトコルバージョン 6 (IPv6)、またはその両方で DB インスタンスと通信できるように指定する デュアルスタックモード。IPv6 アドレス設定プロトコルで DB インスタンスと通信する必要があるリソースがある場合は、デュアルスタックモードを使用します。また、IPv6 CIDR ブロックを、指定した DB サブネットグループ内のすべてのサブネットに関連付けてください。
	CLI オプション:
	
	`--network-type`
	
	RDS API パラメータ:
	
	`NetworkType`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中、ダウンタイムが発生する可能性があります。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
詳細については、「Amazon RDS IP アドレス指定」を参照してください。				

新しいマスターパスワード
 	
マスターユーザーのパスワード。パスワードには、8–41 個の英数字を使用する必要があります。
	CLI オプション:
	
	`--master-user-password`
	
	RDS API パラメータ:
	
	`MasterUserPassword`
		変更は、可能な限り早く非同期的に適用されます。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

オプショングループ
 	
DB インスタンスに関連付けるオプショングループ。
 	
詳細については、「オプショングループを使用する」を参照してください。
	CLI オプション:
	
	`--option-group-name`
	
	RDS API パラメータ:
	
	`OptionGroupName`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。1 つの例外は MariaDB Audit Plugin を RDS for MariaDB または RDS for MySQL DB インスタンスに追加することです。これにより、停止が発生する可能性があります。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされているDB エンジン

Performance Insights
 	
[Performance Insights の有効化] を選択すると、DB インスタンスの負荷をモニタリングし、データベースパフォーマンスの分析とトラブルシューティングを行うことができます。
 	
Performance Insights は、一部の DB エンジンのバージョンと DB インスタンスクラスでは利用できません。DB インスタンスで利用できない場合、[Performance Insights] セクションはコンソールに表示されません。
 	
詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」および「Amazon RDS DB エンジンとインスタンスクラスでサポートされている Performance Insights」を参照してください。
	CLI オプション:
	
	`--enable-performance-insights| --no-enable-performance-insights`
	
	RDS API パラメータ:
	
	`EnablePerformanceInsights`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	Db2 を除くすべて

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

Performance Insights AWS KMS key
 	
Performance Insights データを暗号化するための AWS KMS key の AWS KMS キー識別子。キー識別子は、KMS キーの Amazon リソースネーム (ARN)、AWS KMS キー識別子、またはキーエイリアスです。
 	
詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。
	CLI オプション:
	
	`--performance-insights-kms-key-id`
	
	RDS API パラメータ:
	
	`PerformanceInsightsKMSKeyId`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	Db2 を除くすべて

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

Performance Insights の保持期間
 	
Performance Insights データを保持する期間 (日数)。保持期間の設定は、デフォルト (7 日間) です。パフォーマンスデータをさらに長期間保持するには、1～24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。
 	
詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。
	CLI オプション:
	
	`--performance-insights-retention-period`
	
	RDS API パラメータ:
	
	`PerformanceInsightsRetentionPeriod`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	Db2 を除くすべて

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

プロセッサの機能
 	
DB インスタンスの DB インスタンスクラスの CPU コア数およびコアごとのスレッド数。
 	
詳細については、「RDS for Oracle で DB インスタンスクラスのプロセッサを設定する」を参照してください。
	CLI オプション:
	
	`--processor-features` 、、および `--use-default-processor-features | --no-use-default-processor-features`
	
	RDS API パラメータ:
	
	`ProcessorFeatures` 、、および `UseDefaultProcessorFeatures`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中に、ダウンタイムが発生します。	Oracle のみ

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

プロビジョンド IOPS
 	
DB インスタンスのプロビジョンド IOPS (毎秒ごとの I/O オペレーション) の値。この設定は、[Storage type] (ストレージタイプ) で次のいずれかを選択した場合にのみ使用できます。
 	
• 汎用 SSD (gp3)
• プロビジョンド IOPS SSD (io1)
• プロビジョンド IOPS SSD (io2)
 	
詳細については、「the section called “プロビジョンド IOPS ストレージ”」および「the section called “gp3 ストレージ (推奨)”」を参照してください。
	CLI オプション:
	
	`--iops`
	
	RDS API パラメータ:
	
	`Iops`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

パブリックアクセス
 	
パブリック IP アドレスを DB インスタンスに付与する場合は [パブリックアクセス可能] を選択します。これにより、DB インスタンスは VPC 外からアクセス可能になります。パブリックにアクセス可能となるよう、DB インスタンスは、VPC のパブリックサブネット内にある必要があります。
 	
VPC 内からのみ DB インスタンスにアクセス可能にするには、[パブリックアクセス不可] を選択します。
 	
詳細については、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。
 	
VPC の外部から DB インスタンスに接続するには、DB インスタンスがパブリックにアクセスできる必要があります。また、DB インスタンスのセキュリティグループのインバウンドルールを使用してアクセスを許可する必要があります。さらに、他の要件を満た
	CLI オプション:
	
	`--publicly-accessible | --no-publicly-accessible`
	
	RDS API パラメータ:
	
	`PubliclyAccessible`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
す必要があります。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。 DB インスタンスがパブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続または Direct Connect 接続を使用してプライベートネットワークからアクセスすることもできます。詳細については、「ネットワーク間のトラフィックのプライバシー」を参照してください。				

セキュリティグループ
 	
DB インスタンスに関連付ける VPC セキュリティグループ。
 	
詳細については、「セキュリティグループによるアクセス制御」を参照してください。
	CLI オプション:
	
	`--vpc-security-group-ids`
	
	RDS API パラメータ:
	
	`VpcSecurityGroupIds`
		変更は、可能な限り早く非同期的に適用されます。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ストレージの自動スケーリング
 	
[Enable storage autoscaling (ストレージのオートスケーリングを有効にする)] - DB インスタンスのストレージスペースが不足しないように、必要に応じて Amazon RDS のストレージを自動的に増やせるようにします。
 	
[Maximum storage threshold (ストレージの最大しきい値)] を使用して、Amazon RDS で DB インスタンスのストレージを自動的に増やすための上限を設定します。デフォルトは 1,000 GiB です。
 	
詳細については、「Amazon RDS ストレージの自動スケーリングによる容量の自動管理」を参照してください。
	CLI オプション:
	
	`--max-allocated-storage`
	
	RDS API パラメータ:
	
	`MaxAllocatedStorage`
		変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ストレージスループット
 	
DB インスタンスの新しいストレージスループット値。この設定は、[Storage type] (ストレージタイプ) に汎用 SSD (gp3) を選択した場合にのみ使用できます。
 	
詳細については、「the section called “gp3 ストレージ (推奨)”」を参照してください。
	CLI オプション:
	
	`--storage-throughput`
	
	RDS API パラメータ:
	
	`StorageThroughput`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更時にダウンタイムは発生しません。	すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

ストレージタイプ
 	
使用するストレージのタイプ。
 	
汎用 SSD (gp3) を選択すると、[Advanced settings] (詳細設定) で追加のプロビジョンド IOPS とストレージスループットをプロビジョニングできます。
 	
プロビジョンド IOPS SSD (io1) または プロビジョンド IOPS SSD (io2) を選択した場合は、プロビジョンド IOPS 値を入力します。
 	
Amazon RDS が DB インスタンスでストレージのサイズまたはタイプを変更するための変更を開始すると、インスタンスでストレージの最適化が完了するまで、ストレージのサイズ、パフォーマンス、またはタイプを変更するための別のリクエストを送信できません。任意の 24 時間以内に最大 4 回のストレージ変更を実行できます。
	CLI オプション:
	
	`--storage-type`
	
	RDS API パラメータ:
	
	`StorageType`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			以下の変更を行うと、プロセスが開始される間に短いダウンタイムが発生します。その後は、変更が実行されている間もデータベースを通常どおりに使用できます。
			
			• [General Purpose (SSD)] (汎用 (SSD)) または [Provisioned IOPS (SSD)] (プロビジョンド IOPS (SSD)) から [Magnetic] (マグネティック)。
				すべての DB エンジン

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
詳細については、「Amazon RDS ストレージタイプ」を参照してください。			• [Magnetic] (マグネティック) から [General Purpose (SSD)] (汎用 (SSD)) または [Provisioned IOPS (SSD)] (プロビジョンド IOPS (SSD))。	

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

追加のストレージボリューム	CLI オプション:
	
	`--additional-storage-volumes`
	
	RDS API パラメータ:
	
	`AdditionalStorageVolumes`
		• ストレージボリュームを追加すると、変更はすぐに行われます。このアクションは、[すぐに適用] の設定を無視します。
		• 追加のストレージボリュームを変更するときに、変更をすぐに適用することを選択した場合は、すぐに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテ
			RDS for Oracle または RDS for SQL Server DB インスタンスには、最大 3 つのストレージボリュームを追加できます。gp3 または io2 ストレージタイプを使用するように各追加ボリュームを設定できます。また、割り当てられたストレージ、IOPS、スループットのさまざまな設定を指定して、ワークロードの要件に合わせて最適化することもできます。	RDS for Oracle および RDS for SQL Server

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン
		ナンス期間中に適用されます。 • 追加のストレージボリュームを削除すると、変更はすぐに行われます。このアクションは、[すぐに適用] の設定を無視します。	ボリューム名は、rdsdbdata2 、rdsdbdata3 、または rdsdbdata4 として指定してください。 プライマリボリュームと追加ボリュームの合計ストレージは 256 TiB を超えることはできません。	

コンソールの設定と説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意	サポートされている DB エンジン

DB サブネットグループ
 	
DB インスタンスの DB サブネットグループ。この設定を使用して、DB インスタンスを別の VPC に移動できます。
 	
詳細については、「Amazon VPC と Amazon RDS」を参照してください。
	CLI オプション:
	
	`--db-subnet-group-name`
	
	RDS API パラメータ:
	
	`DBSubnetGroupName`
		変更をすぐに適用するように選択した場合は、直ちに適用されます。
		
		変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
			この変更中に、ダウンタイムが発生します。	すべての DB エンジン

DB インスタンスのメンテナンス

Amazon RDS では、Amazon RDS リソースのメンテナンスを定期的に実行します。以下のトピックで、こうしたメンテナンスアクションとその適用方法について説明します。

DB インスタンスのメンテナンス更新の概要

メンテナンスでは、ほとんどの場合、DB インスタンスの以下のリソースの更新が行われます。

- 基盤となるハードウェア
- 基盤となるオペレーティングシステム (OS)
- データベースエンジンのバージョン

通常、オペレーティングシステムのアップデートはセキュリティ問題に関連しています。できるだけ早く行うことをお勧めします。オペレーティングシステムのアップデートの詳細については、「the section called “アップデートの適用”」を参照してください。

トピック

- メンテナンス更新中のオフラインリソース
- 遅延 DB インスタンスの変更
- DescribePendingMaintenanceActions API の結果整合性

メンテナンス更新中のオフラインリソース

一部のメンテナンス項目では、Amazon RDS が DB インスタンスを少しの間オフラインにする必要があります。リソースをオフラインにする必要があるメンテナンス項目には、必要なオペレーティングシステムやデータベースのパッチが含まれます。セキュリティやインスタンスの信頼性に関連するパッチのみ、必須のパッチ適用として自動的にスケジューリングされます。そのようなパッチ適用はまれであり、通常は数か月に一度です。メンテナンスウィンドウのごくわずかしか必要としないことがほとんどです。

遅延 DB インスタンスの変更

すぐに適用しないことを選択した遅延 DB インスタンスの変更は、メンテナンス期間中に適用されます。例えば、メンテナンス期間中に DB インスタンスクラスまたはパラメータグループを変更することを選択できます。保留中の再起動設定を使用して指定した変更は、[保留中のメンテナンス] リス

トに表示されません。DB インスタンスの変更については、「Amazon RDS DB インスタンスを変更する」を参照してください。

次のメンテナンスウィンドウに向けて保留中の変更を確認するには、describe-db-instances AWS CLI コマンドを使用して PendingModifiedValues フィールドを確認します。

DescribePendingMaintenanceActions API の結果整合性

Amazon RDS DescribePendingMaintenanceActions API は結果整合性モデルに従います。つまり、DescribePendingMaintenanceActions コマンドの結果が、それ以降のすべての RDS コマンドにすぐには表示されない場合があります。以前の API コマンドを使用した直後に DescribePendingMaintenanceActions を使用する場合は、この点に注意してください。

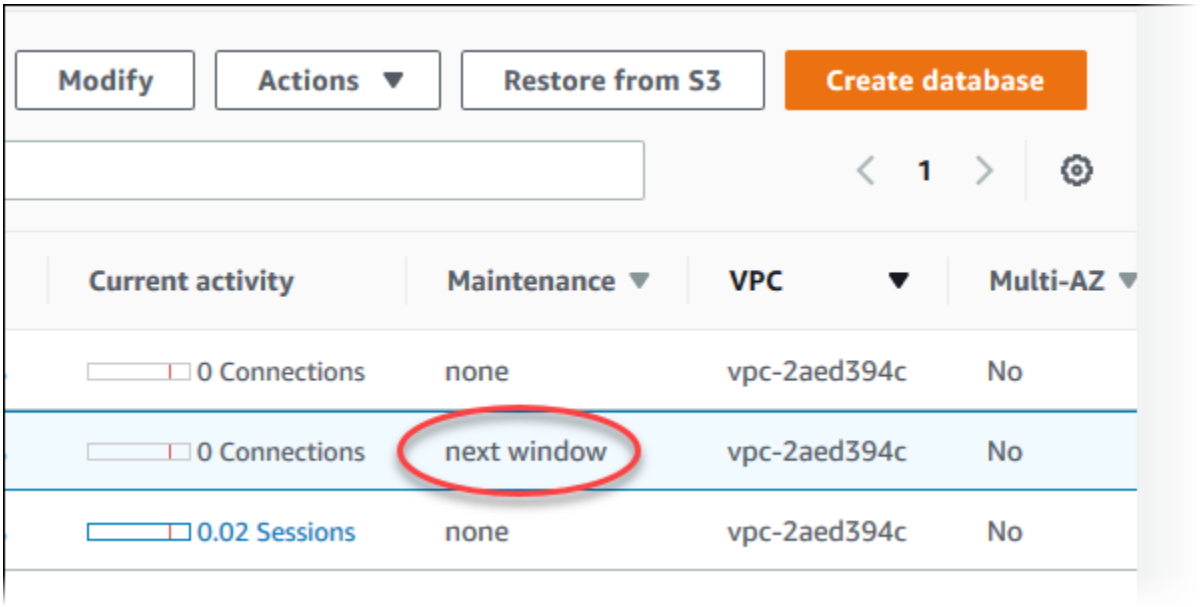
結果整合性は、メンテナンス更新の管理方法に影響を与える可能性があります。例えば、ApplyPendingMaintenanceActions コマンドを実行して DB インスタンスのデータベースエンジンバージョンを更新すると、最終的に DescribePendingMaintenanceActions に表示されます。このシナリオでは、DescribePendingMaintenanceActions はメンテナンスアクションが適用済みであっても適用されていないと示す場合があります。

結果整合性を管理するには、以下を実行します。

- コマンドを実行して変更する前に、DB インスタンスの状態を確認します。エクスポネンシャルバックオフアルゴリズムを使用して適切な DescribePendingMaintenanceActions コマンドを実行し、前のコマンドがシステム内を伝播するのに十分な時間を確保できるようにします。これを行うには、DescribePendingMaintenanceActions コマンドを繰り返し実行し、数秒の待機時間から始めて、最大 5 分間の待機時間まで徐々に増やします。
- DescribePendingMaintenanceActions コマンドが正確なレスポンスを返した場合でも、後続のコマンド間の待機時間を追加します。数秒の待機時間から始めて、エクスポネンシャルバックオフアルゴリズムを適用し、最大約 5 分間の待機時間まで徐々に増やします。

保留中のメンテナンス更新の表示

DB インスタンスでメンテナンスによるアップデートが利用可能かどうかは、RDS コンソール、AWS CLI、または Amazon RDS API を使用して確認します。アップデートが利用できる場合は、次の図に示すように、Amazon RDS コンソールで DB インスタンスの [メンテナンス] 列に表示されます。



DB インスタンスのメンテナンスアップデートが利用できない場合は、列の値が [なし] になります。

DB インスタンスのメンテナンスアップデートが利用できる場合は、列の値が以下のようになります。

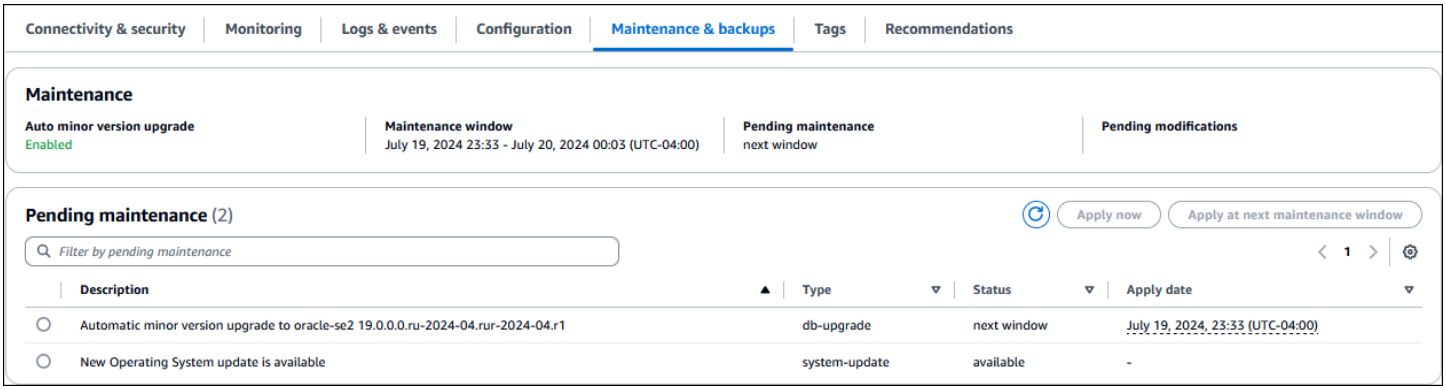
- 必須 - メンテナンスアクションはリソースに適用され、無期限に延期することはできません。
- 利用可能 - メンテナンスアクションは利用可能ですが、自動的にはリソースに適用されません。手動で適用できます。
- 次のウィンドウ - メンテナンスアクションは次回のメンテナンスウィンドウ中にリソースに適用されます。
- 進行中 - メンテナンスアクションはリソースに適用中です。

更新が利用可能な場合は、以下のいずれかの操作を行います。

- メンテナンスの値が [次のウィンドウ] である場合は、[アクション] から [アップグレードを延期する] を選択してメンテナンスアクションを延期します。既にスタートしているメンテナンスアクションは延期できません。
- メンテナンスアクションをすぐに適用する。
- 次回のメンテナンスウィンドウ中にメンテナンスアクションを適用します。
- 何のアクションも実行しません。

AWS マネジメントコンソールを使用してアクションを実行するには

1. DB インスタンスを選択して、その詳細を表示します。
2. [メンテナンスとバックアップ] を選択します。保留中のメンテナンスアクションが表示されます。
3. 実行するアクションを選択し、適用するタイミングを選択します。



メンテナンスウィンドウは、保留中のオペレーションをスタートする時刻を決定しますが、オペレーションの総実行時間を制限しません。メンテナンスオペレーションは、メンテナンスウィンドウが終了するまでに完了するかどうかは保証されておらず、指定終了時間を超える場合もあります。詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。

また、DB インスタンスでメンテナンスによるアップデートが利用可能かどうかは、AWS CLI コマンドの describe-pending-maintenance-actions を使用して確認することもできます。

メンテナンスによるアップデートの適用については、「DB インスタンスへの更新の適用」を参照してください。

Amazon RDS のメンテナンスアクション

RDS DB インスタンスには、次のメンテナンスアクションが適用されます。

- `server-certificate-rotation` – DB インスタンスの Amazon RDS サーバー証明書をローテーションします。

> **Note**
>
> 再起動なしのローテーションをサポートするエンジンは、この通知を受信しません。Amazon RDS for Oracle DB インスタンスはこの通知を受け取りますが、再起動は必要ありません。サーバー証明書のローテーション中にデータベースリスナーのみが再起動

されます。既存のデータベース接続は影響を受けませんが、新しい接続ではリスナーが再起動されるまで短期間だけエラーが発生します。詳細については、「サーバー証明書の自動ローテーション」を参照してください。

- db-upgrade – DB インスタンスの DB エンジンバージョンをアップグレードします。
- hardware-maintenance – DB インスタンスの基盤となるハードウェアでメンテナンスを実行します。
- system-update – DB インスタンスのオペレーティングシステムを更新します。

マルチ AZ 配置のメンテナンス

DB インスタンスをマルチ AZ 配置として実行すると、メンテナンスイベントの影響をさらに軽減できます。この結果は、Amazon RDS が以下の手順に従ってオペレーティングシステムの更新を適用するためです。

1. スタンバイに対してメンテナンスを実行する。
2. スタンバイをプライマリに昇格させる。
3. 旧プライマリでメンテナンスを実行し、その旧プライマリが新しいスタンバイになる。

マルチ AZ 配置で DB インスタンスのデータベースエンジンをアップグレードする場合、Amazon RDS は、プライマリおよびセカンダリ DB インスタンスを両方同時に変更します。この場合、マルチ AZ 配置のプライマリ DB インスタンスとセカンダリ DB インスタンスの両方が、アップグレード中に利用できません。このオペレーションにより、アップグレードが完了するまでダウンタイムが発生します。ダウンタイムの時間は、DB インスタンスのサイズによって異なります。

適用する必要がある基盤となるオペレーティングシステムパッチが存在する場合、プライマリ DB インスタンスにパッチを適用するには、短時間のマルチ AZ フェイルオーバーが必要です。このフェイルオーバーの所要時間は通常 1 分未満です。

DB インスタンスが RDS for MySQL、RDS for PostgreSQL または RDS for MariaDB を実行している場合、ブルー/グリーンデプロイを使用することで、アップグレードに必要なダウンタイムを最小限に抑えることができます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。マルチ AZ 配置で RDS for SQL Server または RDS Custom for SQL Server DB インスタンスをアップグレードすると、Amazon RDS は、ローリングアップグレードを実行するため、停止が発生が発生するのは、フェイルオーバーが発生する場合のみです。詳細については、「マルチ AZ の考慮事項」を参照してください。

マルチ AZ 配置については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

Amazon RDS メンテナンスウィンドウ

メンテナンスウィンドウは、システムの変更を適用する週単位の時間間隔です。各 DB インスタンスには、週ごとのメンテナンスウィンドウがあります。メンテナンスウィンドウは、変更やソフトウェアのパッチなどが実行されるタイミングをコントロールする機会です。メンテナンスウィンドウの調整については、「DB インスタンスの適切なメンテナンスウィンドウの調整」を参照してください。

メンテナンスの適用中は、RDS で DB インスタンス のリソースの一部が使用されます。わずかながらパフォーマンスに影響が出る場合があります。DB インスタンスでは、まれに、メンテナンスによるアップデートを完了するためにマルチ AZ フェイルオーバーが必要になる場合があります。

メンテナンスイベントを特定の週に予定した場合、そのイベントはユーザーが指定した 30 分のメンテナンスウィンドウ中にスタートされます。ほとんどのメンテナンスイベントは 30 分のメンテナンスウィンドウ中に完了しますが、大規模なメンテナンスイベントは 30 分以上かかる場合があります。DB インスタンスが停止すると、メンテナンスウィンドウは一時停止されます。

30 分のメンテナンスウィンドウは、リージョンごとに決められた 8 時間の中でランダムに選択されます。DB インスタンスの作成時にメンテナンスウィンドウを指定しないと、RDS でランダムに選択された曜日に 30 分のメンテナンスウィンドウが割り当てられます。

次の表に、デフォルトのメンテナンスウィンドウを割り当てる時間ブロックを AWS リージョン別に示します。

リージョン名	リージョン	時間ブロック
米国東部 (バージニア北部)	us-east-1	03:00 ~ 11:00 UTC
米国東部 (オハイオ)	us-east-2	03:00 ~ 11:00 UTC
米国西部 (北カリフォルニア)	us-west-1	06:00 ~ 14:00 UTC
米国西部 (オレゴン)	us-west-2	06:00 ~ 14:00 UTC

リージョン名	リージョン	時間ブロック
アフリカ (ケープタウン)	af-south-1	03:00 ~ 11:00 UTC
アジアパシフィック (香港)	ap-east-1	06:00 ~ 14:00 UTC
アジアパシフィック (ハイデラバード)	ap-south-2	06:30 ~ 14:30 UTC
アジアパシフィック (ジャカルタ)	ap-southeast-3	08:00 ~ 16:00 UTC
アジアパシフィック (マレーシア)	ap-southeast-5	09:00 ~ 17:00 UTC
アジアパシフィック (メルボルン)	ap-southeast-4	11:00 ~ 19:00 UTC
アジアパシフィック (ムンバイ)	ap-south-1	06:00 ~ 14:00 UTC
アジアパシフィック (ニュージーランド)	ap-southeast-6	13:00 ~ 21:00 UTC
アジアパシフィック (大阪)	ap-northeast-3	22:00 ~ 23:59 UTC
アジアパシフィック (ソウル)	ap-northeast-2	13:00 ~ 21:00 UTC
アジアパシフィック (シンガポール)	ap-southeast-1	14:00 ~ 22:00 UTC
アジアパシフィック (シドニー)	ap-southeast-2	12:00 ~ 20:00 UTC
アジアパシフィック (台北)	ap-east-2	9:00 ~ 17:00 UTC

リージョン名	リージョン	時間ブロック
アジアパシフィック (タイ)	ap-southeast-7	8:00 ~ 16:00 UTC
アジアパシフィック (東京)	ap-northeast-1	13:00 ~ 21:00 UTC
カナダ (中部)	ca-central-1	03:00 ~ 11:00 UTC
カナダ西部 (カルガリー)	ca-west-1	18:00 ~ 02:00 UTC
中国 (北京)	cn-north-1	06:00 ~ 14:00 UTC
中国 (寧夏)	cn-northwest-1	06:00 ~ 14:00 UTC
欧州 (フランクフルト)	eu-central-1	21:00 ~ 05:00 UTC
欧州 (アイルランド)	eu-west-1	22:00 ~ 06:00 UTC
欧州 (ロンドン)	eu-west-2	22:00 ~ 06:00 UTC
欧州 (ミラノ)	eu-south-1	02:00 ~ 10:00 UTC
欧州 (パリ)	eu-west-3	23:59 ~ 07:29 UTC
欧州 (スペイン)	eu-south-2	13:00 ~ 21:00 UTC
欧州 (ストックホルム)	eu-north-1	23:00 ~ 07:00 UTC
欧州 (チューリッヒ)	eu-central-2	02:00 ~ 10:00 UTC
イスラエル (テルアビブ)	il-central-1	03:00 ~ 11:00 UTC
メキシコ (中部)	mx-central-1	19:00 ~ 3:00 UTC
中東 (バーレーン)	me-south-1	06:00 ~ 14:00 UTC

リージョン名	リージョン	時間ブロック
中東 (アラブ首長国連邦)	me-central-1	05:00 ~ 13:00 UTC
南米 (サンパウロ)	sa-east-1	00:00 ~ 08:00 UTC
AWS GovCloud (米国東部)	us-gov-east-1	17:00 ~ 01:00 UTC
AWS GovCloud (米国西部)	us-gov-west-1	06:00 ~ 14:00 UTC

トピック

- DB インスタンスの適切なメンテナンスウィンドウの調整

DB インスタンスの適切なメンテナンスウィンドウの調整

メンテナンスウィンドウは使用率の最も低い時間帯に設定する必要があります。このため、場合によっては変更が必要になります。この間、DB インスタンスは、システムの変更が適用されていて停止が必要な場合に限り、使用できません。必要な変更を加えるのに必要な最小限の時間のみ使用不可となります。

> ⚠ メンテナンスウィンドウを変更しても、必須のアップグレードは妨げられません。
>
> 保留中の必須アップグレードを避けるために DB インスタンスのメンテナンスウィンドウを繰り返し変更したとしても、必須適用日を過ぎた後は、Amazon RDS が希望するメンテナンスウィンドウ外でアップグレードを適用する場合があります。これは、エンジンバージョンの廃止やインスタンスタイプの廃止キャンペーンなど、すべての強制アップグレードに適用されます。予期しないダウンタイムを回避するには、アップグレードを遅らせるためにメンテナンスウィンドウのスケジュールに依存するのではなく、必須適用日より前に DB インスタンスを事前にアップグレードすることをお勧めします。

次の例では、DB インスタンスの適切なメンテナンスウィンドウを調整します。

この例では、mydbinstance という名前の DB インスタンスが存在しており、必要なメンテナンス時間は "日曜日 05:00 ~ 日曜日 06:00" (UTC) であることを前提としています。

コンソール

必要なメンテナンスウィンドウを調整するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。Modify DB instance ページが表示されます。
4. [メンテナンス] セクションで、メンテナンスウィンドウをアップデートします。

   > **Note**
   >
   > DB インスタンスのメンテナンスウィンドウとバックアップ時間は重複できません。バックアップ時間に重複するメンテナンスウィンドウを入力した場合、エラーメッセージが表示されます。

5. [続行] を選択します。

   確認ページで、変更内容を確認します。
6. 直ちにメンテナンスウィンドウに変更を適用するには、[すぐに適用] を選択します。
7. [DB インスタンスを変更] を選択して、変更を保存します。

   または、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更をキャンセルします。

AWS CLI

必要なメンテナンスウィンドウを調整するには、以下のパラメータを指定して AWS CLI `modify-db-instance` コマンドを使用します。

- `--db-instance-identifier`
- `--preferred-maintenance-window`

Example

次のコード例は、メンテナンス時間を火曜日の午前 4:00 から 4:30 UTC に設定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Windows の場合:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

RDS API

必要なメンテナンスウィンドウを調整するには、以下のパラメータを指定して Amazon RDS API の ModifyDBInstance オペレーションを使用します。

- DBInstanceIdentifier
- PreferredMaintenanceWindow

DB インスタンスへの更新の適用

Amazon RDS を使用すると、メンテナンスオペレーションを適用するタイミングを選択できます。AWS マネジメントコンソール、AWS CLI、または RDS API を使用して Amazon RDS で更新を適用するタイミングを指定できます。

コンソール

DB インスタンスのアップデートを管理するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. アップデートが必要な DB インスタンスを選択します。
4. [アクション] で、以下のいずれかのオプションを選択します。

   - 今すぐパッチを適用
   - 次のウィンドウでパッチを適用

   > **Note**
   >
   > [次のウィンドウでパッチを適用] を選択して、アップデートを延期する場合は、[アップグレードを延期する] を選択します。既にスタートしているメンテナンスアクションは延期できません。
   > メンテナンスアクションをキャンセルするには、DB インスタンスを変更し、[マイナーバージョン自動アップグレード] を無効にします。

AWS CLI

保留中のアップデートを DB インスタンスに適用するには、AWS CLI コマンドの apply-pending-maintenance-action を使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
```

```
    --apply-action system-update \
    --opt-in-type immediate
```

Windows の場合:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> メンテナンスアクションを延期するには、--opt-in-type として undo-opt-in を指定します。メンテナンスアクションが既にスタートされている場合は、--opt-in-type として undo-opt-in を指定できません。
> メンテナンスアクションをキャンセルするには、modify-db-instance AWS CLI コマンドを実行し、--no-auto-minor-version-upgrade を指定します。

少なくとも 1 つの保留中のアップデートがあるリソースのリストを取得するには、describe-pending-maintenance-actions AWS CLI コマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Windows の場合:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

AWS CLI コマンド describe-pending-maintenance-actions の --filters パラメータを指定して、DB インスタンスのリソースのリストを取得することもできます。--filters コマンドの形式は、Name=*filter-name*,Value=*resource-id*,... です。

以下は、フィルターの Name パラメータの許容値です。

- db-instance-id - DB インスタンス識別子または Amazon リソースネーム (ARN) のリストが許容されます。返されるリストには、これらの ID または ARN で識別された DB インスタンスの保留中のメンテナンスアクションのみが含まれます。
- db-cluster-id - DB クラスター識別子または Amazon Aurora 用の ARN のリストが許容されます。返されるリストには、これらの ID または ARN で識別された DB クラスターの保留中のメンテナンスアクションのみが含まれます。

次の例では、sample-instance1 および sample-instance2 DB インスタンスの保留中のメンテナンスアクションが返されます。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

Windows の場合:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-instance-id,Values=sample-instance1,sample-instance2
```

RDS API

アップデートを DB インスタンスに適用するには、Amazon RDS API の ApplyPendingMaintenanceAction オペレーションを呼び出します。

少なくとも 1 つの保留中のアップデートがあるリソースのリストを返すには、Amazon RDS API の DescribePendingMaintenanceActions オペレーションを呼び出します。

RDS DB インスタンスのオペレーティングシステムの更新

RDS for Db2、RDS for MariaDB、RDS for MySQL、RDS for PostgreSQL、RDS for SQL Server、RDS for Oracle および RDS Custom for SQL Server DB インスタンスでは、オペレーティングシステムのアップデートが必要になることがあります。Amazon RDS は、データベースパフォーマンスと顧客の全体的なセキュリティ体制改善のために、OS を新しいバージョンにアップグレードします。通常、アップデートには約 10 分かかります。オペレーティングシステムのアップデートでは、DB インスタンスの DB エンジンのバージョンまたは DB インスタンスクラスは変更されません。

オペレーティングシステムのアップデートは、オプションの場合も必須の場合もあります。

- オプションのアップデートは、随時適用できます。これらのアップデートはオプションですが、RDS フリートを最新の状態に保つために定期的に適用することをお勧めします。RDS は、これらのアップデートを自動的に適用しません。

  新しいオプションのオペレーティングシステムパッチが利用可能になったときに通知を受けるには、セキュリティパッチイベントカテゴリの RDS-EVENT-0230 をサブスクライブできます。RDS イベントにサブスクライブする方法については、「Amazon RDS イベント通知にサブスクライブする」を参照してください。

  > **Note**
  >
  > RDS-EVENT-0230 は、オペレーティングシステムのディストリビューションのアップグレードに適用されません。

- 必須アップデートが必要で、適用日があります。この適用日付より前にアップデートをスケジュールするように計画してください。指定した適用日後、割り当てられたメンテナンスウィンドウ中に、Amazon RDS は DB インスタンスのオペレーティングシステムを最新バージョンに自動的にアップグレードします。

> **Note**
>
> さまざまなコンプライアンス義務を果たすためには、すべてのオプションおよび必須のアップデートを最新の状態に保つことが必要になる場合があります。RDS によって提供されるすべてのアップデートは、メンテナンス期間中に定期的に適用することをお勧めします。

AWS マネジメントコンソール または AWS CLI を使用すると、オペレーティングシステムのアップグレードの種類に関する情報を取得できます。

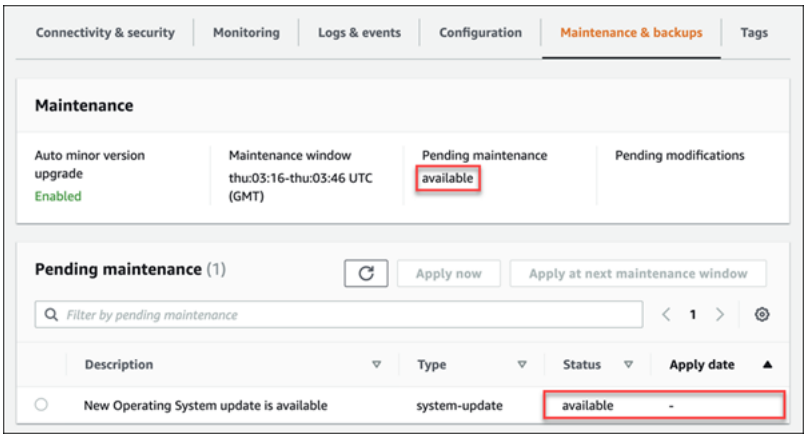
コンソール

AWS マネジメントコンソール を使用してアップデート情報を取得するには

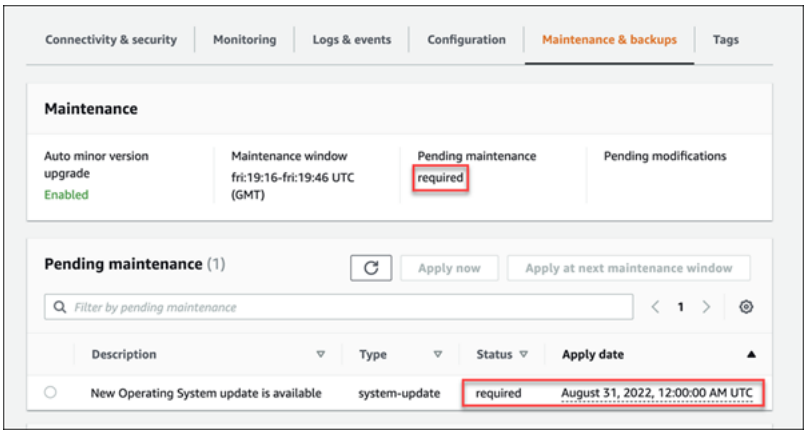
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、DB インスタンスを選択します。

3. [メンテナンスとバックアップ] を選択します。
4. [Pending maintenance] (保留中のメンテナンス) セクションで、オペレーティングシステムのアップデートを検索し、[Status] (ステータス) 値をチェックします。

オプションのアップデートの場合、AWS マネジメントコンソールで、メンテナンス [ステータス] が [利用可能] と設定されており、次の画像のように [日付を適用] がありません。



必須のアップデートの場合、メンテナンス[Status] (ステータス) が [必須] と設定されており、次の画像のように[Apply date] (適用日) があります。



AWS CLI

AWS CLI からアップデート情報を取得するには、describe-pending-maintenance-actions コマンドを使用します。

```
aws rds describe-pending-maintenance-actions
```

オペレーティングシステムの必須アップデートには、AutoAppliedAfterDate 値と CurrentApplyDate 値が含まれます。オペレーティングシステムのオプションのアップデートには、これらの値は含まれません。

次の出力は、必須のオペレーティングシステムのアップデートを示しています。

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb1",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "AutoAppliedAfterDate": "2022-08-31T00:00:00+00:00",
      "CurrentApplyDate": "2022-08-31T00:00:00+00:00",
      "Description": "New Operating System update is available"
    }
  ]
}
```

次の出力は、オプションのオペレーティングシステムのアップデートを示しています。

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System update is available"
    }
  ]
}
```

オペレーティングシステムのアップデートの可用性

オペレーティングシステムのアップデートは、DB エンジンのバージョンと DB インスタンスクラスに固有です。したがって、DB インスタンスは、異なる時間にアップデートを受信または要求します。そのエンジンのバージョンとインスタンスクラスに基づいた DB インスタンスにオペレーティングシステムのアップデートがある場合は、アップデートがコンソールに表示されます。describe-pending-maintenance-actions AWS CLI コマンドを実行するか、または DescribePendingMaintenanceActions RDS API オペレーションを呼び出すことによっても表示できます。インスタンスでアップデートが利用可能である場合、DB インスタンスへの更新の適用 の手順に従って OS をアップデートできます。

自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するための AWS Organizations アップグレードロールアウトポリシーをサポートしています。このポリシーは、以下の方法でインスタンスの制御されたアップグレード戦略を実装するのに役立ちます。

アップグレードロールアウトポリシーの仕組み

新しいマイナーエンジンバージョンが自動アップグレードの対象になると、ポリシーは定義された順序に基づいてアップグレードシーケンスを制御します。

- [first] とマークされたリソース (通常は開発環境) は、メンテナンスウィンドウ中にアップグレードの対象となります。
- 指定されたベイク時間が経過すると、[second] とマークされたリソースが対象になります。
- 別の指定ベイク時間が過ぎると、[last] とマークされたリソース (通常は本番環境) が対象になります。
- AWS Health 通知によるアップグレードの進行状況のモニタリング。

アップグレードの順序は、次の方法で定義できます。

- アカウントレベルのポリシー – 指定されたアカウントのすべての対象リソースに適用されます。
- リソースタグ — タグに基づいて特定のリソースに適用します。

> **Note**
>
> アップグレードポリシーが設定されていないリソース、またはポリシーから除外されているリソースは、自動的に [second] のアップグレード順序を受け取ります。

前提条件

- AWS アカウントは、アップグレードロールアウトポリシーが有効になっている Organizations の組織の一部である必要があります
- インスタンスのマイナーバージョン自動アップグレードを有効にします

- タグは、アップグレードロールアウトポリシーに厳密には必要ありません。さまざまな環境 (開発、テスト、QA、本番稼働など) に特定のアップグレード順序を定義する場合は、タグを使用できます。ポリシーにタグ設定を含めない場合、そのポリシーのすべてのリソースはデフォルトのアップグレード順序に従います。

前提条件

- AWS アカウントは、アップグレードロールアウトポリシーが有効になっている Organizations の組織の一部である必要があります
- インスタンスのマイナーバージョン自動アップグレードを有効にします
- リソースにタグ付けして環境 (開発、テスト、本番稼働など) を特定する

リソースにタグを付けるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. タグを付加するインスタンスを選択します。
4. [アクション] を選択して [タグを管理] を選択します。
5. [タグを追加] を選択します。
6. タグキー (「環境」など) と値 (「開発」など) を入力します。
7. [タグを追加] を選択し、[保存] を選択します。

AWS CLI を使用してタグを追加することもできます。

```
aws rds add-tags-to-resource \
    --resource-name arn:aws:rds:region:account-number:db:instance-name \
    --tags Key=Environment,Value=Development
```

アップグレードの順序とフェーズ

アップグレードロールアウトポリシーでは、次の 3 つのアップグレード順序がサポートされています。

- [first] - 通常、開発環境またはテスト環境に使用されます

- [second] - 通常、QA 環境に使用されます。特に設定されていないリソースのデフォルトの順序
- [last] - 通常、本番環境用に予約されています

新しいマイナーエンジンバージョンが自動アップグレードの対象になった場合:

- アップグレード順序が [first] のリソースは、設定されたメンテナンスウィンドウ中にアップグレードの対象となります。
- 指定されたベイク時間の後、アップグレード順序 [second] のリソースは、メンテナンスウィンドウ中にアップグレードの対象となります。
- 別の指定ベイク時間の後、アップグレード順序 [last] のリソースは、メンテナンスウィンドウ中にアップグレードの対象となります。
- 自動マイナーバージョンアップグレードキャンペーンは、アップグレード順序 [first]、[second]、および [last] の対象となるすべてのリソースがアップグレードされた後、またはキャンペーンがスケジュールされた終了日のいずれか早い方に達した後に終了します。

> **Note**
>
> すべてのマイナーバージョンの自動アップグレードは、アプリケーションへの潜在的な影響を最小限に抑えるために、各インスタンスで設定されたメンテナンスウィンドウ中に実行されます。

オブザーバビリティ

AWS Health とモニタリング

AWS Health 通知を受け取ります:

- 自動マイナーバージョンアップグレードキャンペーンを開始する前
- アップグレードの進行状況の追跡とモニタリングに役立つ各フェーズの移行の間
- AWS Health コンソールでフリート全体でアップグレードされたリソースの数を示す進捗状況の更新

Amazon RDS イベント通知

- 自動マイナーバージョンアップグレードが有効になっているリソースの通知。以下が含まれます。

  - リソースがアップグレード順序 ([first]、[second]、または [last]) に基づいてアップグレードの対象となる場合
  - メンテナンスウィンドウ中のスケジュールされたアップグレードタイムライン
  - 個々のデータベースのアップグレードの開始ステータスと完了ステータス
- 自動モニタリングのために Amazon EventBridge0 を使用してこれらのイベントをサブスクライブする

考慮事項

留意すべき考慮事項:

- このポリシーは、アクティブなキャンペーン中に行われたポリシーの変更を含む、今後のすべての自動マイナーバージョンアップグレードキャンペーンに適用されます。
- 進行中のアップグレードキャンペーンに参加する場合、リソースは現在実行中のアップグレード順序に従い、設定されたポリシーを待機しません。
- アップグレードポリシーが設定されていないリソースは、自動的に [second] のアップグレード順序を受け取ります。
- このポリシーは、次のフェーズに進む前に、アップグレードフェーズ間の検証期間を提供します。
- ポリシータグまたはリソースタグの変更を加えると、新しいアップグレード順序が適用される前に反映される時間が必要になります。
- このポリシーは、自動マイナーバージョンアップグレードが有効になっている Amazon RDS リソースにのみ適用されます。
- 環境内で問題を検出した場合は、後続の環境の自動マイナーバージョンアップグレードを無効にするか、アップグレードが次のアップグレード順序に進む前に検証期間を使用して問題を解決できます。

> **Note**
>
> この機能は、2026 年 1 月以降にリリースされた Oracle Database エンジンバージョンのマイナーバージョン自動アップグレードをサポートします。

RDS リソースのタグ付けの詳細については、「Amazon RDS リソースのタグ付け」を参照してください。アップグレードロールアウトポリシーの設定と使用の詳細な手順については、「AWS Organizations ユーザーガイド」の「AWS Organizations の開始方法」を参照してください。

DB インスタンスのエンジンバージョンのアップグレード

Amazon RDS では、サポート対象の各データベースエンジンの新しいバージョンを提供しているため、DB インスタンスを最新の状態に維持することができます。新しいバージョンには、データベースエンジンのバグ修正、セキュリティの強化、およびその他の改善が含まれます。Amazon RDS がデータベースエンジンの新しいバージョンをサポートすると、DB インスタンスをアップグレードする方法とタイミングを選択できます。

アップグレードには、メジャーバージョンのアップグレードとマイナーバージョンのアップグレードの 2 種類があります。一般的に、メジャーエンジンバージョンのアップグレードは、既存のアプリケーションと互換性のない変更を導入する場合があります。それに対して、マイナーバージョンのアップグレードには、既存のアプリケーションとの下位互換性がある変更のみが含まれます。

マルチ AZ DB クラスターの場合、RDS for PostgreSQL のみがサポートされています。マイナーバージョンアップグレードは、マルチ AZ DB クラスターをサポートするすべてのエンジンでサポートされています。詳細については、「the section called “マルチ AZ DB クラスターのアップグレード”」を参照してください。

バージョン番号付けの順序は、各データベースエンジンに固有です。例えば、RDS for MySQL 5.7 および 8.0 は、メジャーエンジンバージョンで、5.7 から 8.0 バージョンへのアップグレードは、メジャーバージョンのアップグレードです。RDS for MySQL 5.7.22 および 5.7.23 は、マイナーエンジンバージョンで、5.7.22 から 5.7.23 へのアップグレードは、マイナーバージョンのアップグレードです。

> **Important**
>
> DB インスタンスをアップグレードしているときは、変更できません。アップグレード中、DB インスタンスのステータスは `upgrading` です。

特定の DB エンジンのメジャーバージョンおよびマイナーバージョンのアップグレードの詳細については、以下の DB エンジンに関するドキュメントを参照してください。

- MariaDB DB エンジンのアップグレード
- Microsoft SQL Server DB エンジンのアップグレード
- RDS for MySQL DB エンジンのアップグレード
- RDS for Oracle DB エンジンのアップグレード

- RDS for PostgreSQL DB エンジンのアップグレード

メジャーバージョンのアップグレードの場合は、AWS マネジメントコンソール、AWS CLI、または RDS API を通じて、DB エンジンバージョンを手動で変更する必要があります。マイナーバージョンのアップグレードの場合は、エンジンバージョンを手動で変更することも、マイナーバージョン自動アップグレードオプションを有効にすることもできます。

> **Note**
>
> データベースエンジンをアップグレードするには、ダウンタイムが必要です。ブルー/グリーンデプロイを使用することで、DB インスタンスのアップグレードに必要なダウンタイムを最小限に抑えることができます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

トピック

- エンジンバージョンの手動アップグレード
- マイナーエンジンバージョンの自動アップグレード

エンジンバージョンの手動アップグレード

DB インスタンスのエンジンバージョンを手動でアップグレードするには、AWS マネジメントコンソール、AWS CLI または RDS API を使用することができます。

コンソール

コンソールを使用して DB インスタンスのエンジンバージョンをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、アップグレードする DB インスタンスを選択します。
3. [Modify] を選択します。Modify DB instance ページが表示されます。
4. [DB engine version] で、新しいバージョンを選択します。
5. [Continue] を選択して、変更の概要を確認します。
6. アップグレードをスケジュールするタイミングを決定します。

- 変更を保留中の変更キューに入れるには、[次の定期メンテナンス期間中に適用] を選択します。次回のメンテナンスウィンドウで、RDS は、キュー内のすべての保留中の変更を適用します。
- 変更をすぐに反映させるには、[Apply immediately] を選択します。このオプションを選択すると、停止状態になる場合があります。詳細については、「スケジュール変更設定の使用」を参照してください。

7. 確認ページで、変更内容を確認します。正しい場合は、[Modify DB Instance (DB インスタンスを変更)] を選択して変更を保存します。

   または、[Back] を選択して変更を編集するか、[Cancel] を選択して変更をキャンセルします。

AWS CLI

DB インスタンスのエンジンバージョンをアップグレードするには、CLI の modify-db-instance コマンドを使用します。以下のパラメータを指定します。

- `--db-instance-identifier` – DB インスタンスの名前です。
- `--engine-version` – アップグレード先のデータベースエンジンのバージョン番号です。

  有効なエンジンバージョンの詳細については、AWS CLI の describe-db-engine-versions コマンドを参照してください。
- `--allow-major-version-upgrade` – メジャーバージョンをアップグレードします。
- `--no-apply-immediately` – 次のメンテナンス時間中に変更を適用します。今すぐ変更を適用するには、`--apply-immediately` を使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

RDS API

DB インスタンスのエンジンバージョンをアップグレードするには、ModifyDBInstance アクションを使用します。以下のパラメータを指定します。

- DBInstanceIdentifier – DB インスタンスの名前、例えば *mydbinstance* です。
- EngineVersion – アップグレード先のデータベースエンジンのバージョン番号です。有効なエンジンバージョンについては、DescribeDBEngineVersions オペレーションを使用します。
- AllowMajorVersionUpgrade – メジャーバージョンのアップグレードを許可するかどうか。そのためには、値を true に設定します。
- ApplyImmediately – 変更をすぐに適用するか、次のメンテナンスウィンドウ中に適用するかを指定します。今すぐ変更を適用するには、値を true に設定します。次のメンテナンスウィンドウ中に変更を適用するには、値を false に設定します。

マイナーエンジンバージョンの自動アップグレード

マイナーバージョン自動アップグレードは、データベースを定期的に最新のデータベースエンジンのバージョンに更新します。ただし、アップグレードに最新のデータベースエンジンのバージョンが含まれているとは限りません。特定の時間に特定のバージョンでデータベースを維持する必要がある場合は、必要なスケジュールに従って必要なデータベースのバージョンに手動でアップグレードすることをお勧めします。重大なセキュリティ問題が発生した場合、またはバージョンがサポート終了日に達すると、[マイナーバージョン自動アップグレード] オプションを有効にしていない場合でも、Amazon RDS はマイナーバージョンのアップグレードを適用します。詳細については、お使いのデータベースエンジンのアップグレードドキュメントを参照してください。

- RDS for PostgreSQL のマイナーバージョンの自動アップグレード
- RDS for MySQL のマイナーバージョンの自動アップグレード
- RDS for MariaDB のマイナーバージョンの自動アップグレード
- Oracle のマイナーバージョンのアップグレード
- Microsoft SQL Server DB エンジンのアップグレード

- Amazon RDS での Db2 のバージョン

トピック

- 自動マイナーバージョンアップグレードの仕組み
- 自動マイナーバージョンアップグレードの実行
- メンテナンスアップデートの提供状況を確認する
- 自動マイナーバージョンアップグレードの対象を検索する

自動マイナーバージョンアップグレードの仕組み

アップグレードターゲットは、Amazon RDS がデータベースをアップグレードする DB エンジンバージョンです。マイナーエンジンバージョンは、次の条件が満たされた場合にアップグレードターゲットとして指定されます。

- データベースは、ターゲットマイナーエンジンバージョンより低い DB エンジンのマイナーバージョンを実行中です。

  DB インスタンスの現在のエンジンバージョンは、データベース詳細ページの [設定] タブで、または CLI コマンド `describe-db-instances` を実行することで確認できます。
- データベースは、自動マイナーバージョンアップグレードを有効にしています。

RDS では、メンテナンスウィンドウ中にアップグレードが自動で実行されるようスケジュールされます。アップグレード中、RDS は以下を実行します。

1. システムの事前チェックを実行して、データベースが正常でアップグレードの準備ができていることを確認する
2. DB エンジンをターゲットのマイナーエンジンバージョンにアップグレードする
3. アップグレード後のチェックを実行する
4. データベースのアップグレードを完了とマークする

自動アップグレードにはダウンタイムが伴います。ダウンタイムの長さは、DB エンジンの種類やデータベースのサイズなど、さまざまな要因によって異なります。

自動マイナーバージョンアップグレードの実行

以下のタスクを実行すると、DB インスタンスのマイナーバージョン自動アップグレードを有効にするかどうかを制御できます。

- DB インスタンスの作成
- DB インスタンスを変更する
- リードレプリカの作成
- スナップショットからの DB インスタンスの復元
- 特定の時点への DB インスタンスの復元
- Amazon S3 からの DB インスタンのインポート (Amazon S3 の MySQL バックアップの場合)

これらのタスクを実行すると、次の方法で DB インスタンスのマイナーバージョン自動アップグレードを有効にするかどうかを制御できます。

- コンソールを使用して、[マイナーバージョン自動アップグレード] を設定します。
- AWS CLI を使用して、`--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` オプションを設定します。
- RDS API を使用して、`AutoMinorVersionUpgrade` パラメータを設定します。

メンテナンスアップデートの提供状況を確認する

DB エンジンバージョンのアップグレードなど、メンテナンス更新が DB インスタンスで利用可能かどうかを確認するには、コンソール、AWS CLI、または RDS API を使用できます。DB エンジンバージョンを手動でアップグレードして、メンテナンス期間を調整することもできます。詳細については、「DB インスタンスのメンテナンス」を参照してください。

自動マイナーバージョンアップグレードの対象を検索する

次の AWS CLI コマンドを使用すると、特定の AWS リージョン で指定されたマイナー DB エンジンバージョンの自動マイナーアップグレードターゲットの現在のバージョンを確認できます。CreateDBInstance の `Engine` パラメータの説明で、このコマンドに使用可能な `--engine` 値を見つけることができます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
```

```
--engine engine \
--engine-version minor-version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine engine ^
--engine-version minor-version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

例えば、次のAWS CLIコマンドにより、米国東部 (オハイオ)AWSリージョン (us-east-2) の MySQL マイナーバージョン 8.0.11 の自動マイナーアップグレードターゲットを特定できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version 8.0.11 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version 8.0.11 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

以下のような出力が生成されます。

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  8.0.15         |
|  False       |  8.0.16         |
|  False       |  8.0.17         |
|  False       |  8.0.19         |
|  False       |  8.0.20         |
|  False       |  8.0.21         |
|  True        |  8.0.23         |
|  False       |  8.0.25         |
+--------------+-----------------+
```

この例では、MySQL バージョン 8.0.23 の`AutoUpgrade`値は`True`です。したがって、自動マイナーアップグレードターゲットは MySQL バージョン 8.0.23 であり、出力でハイライトされています。

> **Important**
>
> すぐに RDS for PostgreSQL DB インスタンスを Aurora PostgreSQL DB クラスターに移行する予定がある場合は、移行計画の早い段階で DB インスタンスのマイナーバージョンの自動アップグレードをオフにすることを強くお勧めします。RDS for PostgreSQL バージョンが Aurora PostgreSQL でサポートされていない場合、Aurora PostgreSQL への移行が遅れる可能性があります。Aurora PostgreSQL バージョンについては、Amazon Aurora PostgreSQL のエンジンのバージョンを参照してください。

DB インスタンスの名前を変更する

AWS マネジメントコンソール、AWS CLI `modify-db-instance` コマンド、または Amazon RDS API `ModifyDBInstance` アクションを使用して、DB インスタンスの名前を変更できます。DB インスタンスの名前を変更すると、広範囲に影響が及びます。以下は、DB インスタンスの名前を変更する前の考慮事項の一覧です。

- DB インスタンスの名前を変更すると、DB インスタンスに割り当てた名前が URL に含まれているので、DB インスタンスのエンドポイントが変わります。必ず古い URL のトラフィックを新しい URL にリダイレクトする必要があります。
- DB インスタンスの名前を変更すると、DB インスタンスによって使用されていた古い DNS 名はすぐに削除されますが、数分間キャッシュされたままの場合があります。名前を変更した DB インスタンスの新しい DNS 名は、10 分程度で有効になります。名前を変更した DB インスタンスは、新しい名前が有効になるまで使用できません。
- インスタンスの名前を変更する際に既存の DB インスタンスの名前を使用することはできません。
- DB インスタンスに関連付けられたすべてのリードレプリカは、名前を変更した後もインスタンスに関連付けられたままです。例えば、本稼働データベースを提供する DB インスタンスと、リードレプリカが複数関連付けられているインスタンスがあるとします。DB インスタンスの名前を変更し、本稼働環境で DB スナップショットを置き換えた場合、名前を変更した DB インスタンスにはリードレプリカが関連付けられたままです。
- DB インスタンスの名前に関連付けられたメトリクスとイベントは、DB インスタンス名を再利用する場合も維持されます。例えば、リードレプリカを昇格させ、前のプライマリ DB インスタンスの名前となるように名前を変更した場合、プライマリ DB インスタンスに関連付けられたイベントとメトリクスは、名前が変更されたインスタンスに関連付けられます。
- DB インスタンスのタグは、名前を変更するかどうかにかかわらず DB インスタンスに残ります。
- 名前を変更した DB インスタンスの DB スナップショットは保持されます。

> **Note**
>
> DB インスタンスはクラウドで実行される独立したデータベース環境です。DB インスタンスは、複数のデータベース、または複数のスキーマを使用する 1 つの Oracle データベースをホストすることができます。データベース名の変更の詳細については、DB エンジンのドキュメントを参照してください。

名前を変更して既存の DB インスタンスを置き換える

DB インスタンスの名前を変更する最も一般的な理由は、リードレプリカを昇格させること、または DB スナップショットまたは ポイントインタイムリカバリ (PITR) からデータを復元することです。データベースの名前を変更することにより、DB インスタンスを参照するアプリケーションコードを変更せずに DB インスタンスを置き換えることができます。このような場合、次の作業を行います。

1. プライマリ DB インスタンスに向かうトラフィックを停止します。これを行うには、DB インスタンスでデータベースにアクセスするトラフィックをリダイレクトしたり、他の方法を使用してトラフィックが DB インスタンスでデータベースにアクセスしないようにする必要があります。
2. このトピックで後述するように、プライマリ DB インスタンスの名前を、プライマリ DB インスタンスではなくなったことを示す名前に変更します。
3. DB スナップショットから復元するか、リードレプリカを昇格させることにより、新しいプライマリ DB インスタンスを作成し、新しいインスタンスに以前のプライマリ DB インスタンスの名前を付けます。
4. リードレプリカに新しいプライマリ DB インスタンスを関連付けます。

古いプライマリ DB インスタンスを削除した場合、古いプライマリ DB インスタンスの不要な DB スナップショットをすべて削除する必要があります。

リードレプリカの昇格については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

> **Important**
>
> DB インスタンスは名前が変更されると再起動されます。常時オプションやミラーリングオプションが有効になっている RDS for SQL Server のマルチ AZ インスタンスの場合、名前変更オペレーション後にインスタンスを再起動すると、フェイルオーバーが予想されます。

コンソール

DB インスタンスの名前を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

3. 名前を変更する DB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. [設定] で、[DB インスタンス識別子] に新しい名前を入力します。
6. [続行] をクリックしてください。
7. 変更をすぐに反映させるには、[Apply immediately] を選択します。このオプションを選択すると、停止状態になる場合があります。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
8. 確認ページで、変更内容を確認します。正しい場合は、[Modify DB Instance (DB インスタンスを変更)] を選択して変更を保存します。

   または、[Back] を選択して変更を編集するか、[Cancel] を選択して変更をキャンセルします。

AWS CLI

DB インスタンスの名前を変更するには、AWS CLI コマンド `modify-db-instance` を使用します。DB インスタンスの新しい名前とともに、現在の `--db-instance-identifier` 値および `--new-db-instance-identifier` パラメータを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier DBInstanceIdentifier \
    --new-db-instance-identifier NewDBInstanceIdentifier
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier DBInstanceIdentifier ^
    --new-db-instance-identifier NewDBInstanceIdentifier
```

RDS API

DB インスタンスの名前を変更するには、以下のパラメータを指定して Amazon RDS API オペレーション `ModifyDBInstance` を呼び出します。

- `DBInstanceIdentifier` — インスタンスの既存の名前

- NewDBInstanceIdentifier — インスタンスの新しい名前

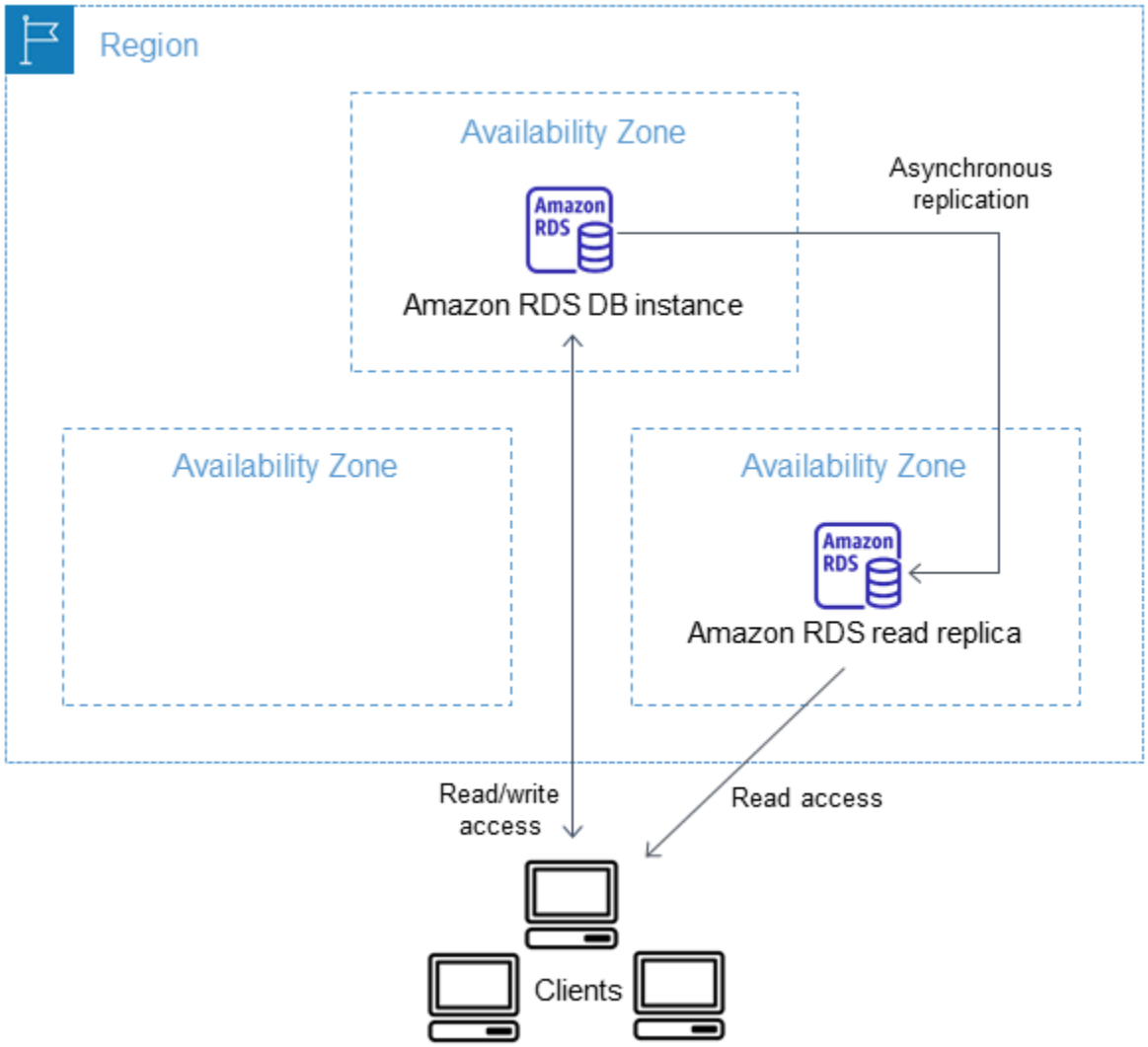
DB インスタンスのリードレプリカの操作

リードレプリカは、DB インスタンスの読み取り専用コピーです。クエリをアプリケーションからリードレプリカにルーティングすることにより、プライマリ DB インスタンスの負荷を軽減できます。こうすることにより、単一 DB インスタンスの容量制約にとらわれることなく伸縮自在にスケールアウトし、読み取り負荷の高いデータベースワークロードに対応できます。

ソース DB インスタンスからリードレプリカを作成するには、Amazon RDS では、DB エンジン組み込みのレプリケーション機能を使用します。特定のエンジンでのリードレプリカの使用については、以下のセクションを参照してください。

- Amazon RDS for Db2 のレプリカの使用
- MariaDB リードレプリカの使用
- Amazon RDS での Microsoft SQL Server 用のリードレプリカの使用
- MySQL リードレプリカの使用
- Amazon RDS for Oracle でのリードレプリカの使用
- Amazon RDS for PostgreSQL でのリードレプリカの使用

ソース DB インスタンスからリードレプリカを作成すると、ソースがプライマリ DB インスタンスになります。プライマリ DB インスタンスに対して更新を行うと、Amazon RDS はリードレプリカに非同期的にコピーします。次の図は、ソース DB インスタンスが別のアベイラビリティーゾーン (AZ) のリードレプリカにレプリケートされる様子を示しています。クライアントにはプライマリ DB インスタンスへの読み取り/書き込みアクセス権、およびレプリカへの読み取り専用アクセス権があります。



リードレプリカは、レプリカに使用される DB インスタンスクラスと同じレートで標準 DB インスタンスとして請求されます。同じ AWS リージョン 内のソース DB インスタンスとリードレプリカ間のデータのレプリケーションで発生したデータ転送に対しては課金されません。詳細については、「クロスリージョンレプリケーションのコスト」および「Amazon RDS DB インスタンスの請求」を参照してください。

トピック

- Amazon RDS リードレプリカの概要
- DB エンジンのリードレプリカ間の違い
- リードレプリカの作成
- リードレプリカをスタンドアロン DB インスタンスに昇格させる
- リードレプリケーションのモニタリング

- 別の でのリードレプリカの作成AWS リージョン

Amazon RDS リードレプリカの概要

次のセクションでは、DB インスタンスのリードレプリカについて説明します。マルチ AZ DB クラスターのリードレプリカについては、「the section called “マルチ AZ DB クラスターのリードレプリカの操作”」を参照してください。

トピック

- リードレプリカのユースケース
- リードレプリカの仕組み
- マルチ AZ 配置のリードレプリカ
- クロスリージョンリードレプリカ
- リードレプリカのストレージタイプ
- レプリカからレプリカを作成する場合の制限事項
- レプリカを削除する際の注意事項

リードレプリカのユースケース

以下のような状況では、ソース DB インスタンスに対する 1 つまたは複数のリードレプリカのデプロイが適している可能性があります。

- 読み込みが多いデータベースに対して 1 つの DB インスタンスの処理または I/O 機能を拡張します。このような過度の読み込みトラフィックを 1 つまたは複数のリードレプリカに割り振ることができます。
- ソース DB インスタンスが利用可能でない場合に読み込みトラフィックを誘導します。場合によっては、バックアップまたは予定される保守による I/O 停止により、ソース DB インスタンスで I/O リクエストを取得できないことがあります。このような場合は、リードトラフィックをリードレプリカに誘導することができます。このようなユースケースの場合、ソース DB インスタンスを利用できないため、リードレプリカのデータは「古い」場合があるので注意が必要です。
- ビジネスレポーティングまたはデータウェアハウジングでは、本稼働 DB インスタンスではなく、ビジネスレポーティングクエリをリードレプリカに対して実行します。
- 災害復旧の実装。プライマリ DB インスタンスで障害が発生した場合、災害対策ソリューションとして、リードレプリカをスタンドアロンインスタンスに昇格させることができます。

リードレプリカの仕組み

リードレプリカを作成したら、既存の DB インスタンスをソースとして指定します。次に、Amazon RDS でソースインスタンスのスナップショットを作成し、スナップショットから読み取り専用インスタンスを作成します。Amazon RDS では、DB エンジン用の非同期レプリケーションの方法を使用することで、プライマリ DB インスタンスに変更がある場合は必ずリードレプリカが更新されます。

リードレプリカは、読み取り専用接続のみ許可される DB インスタンスとして動作します。例外は RDS for Db2 エンジンと RDS for Oracle DB エンジンで、それぞれスタンバイモードとマウントされたモードでレプリカデータベースをサポートしています。スタンバイレプリカとマウントされたレプリカは、ユーザー接続を受け付けないため、読み取り専用のワークロードに対応できません。スタンバイレプリカとマウントされたレプリカの主な用途は、クロスリージョンのディザスタリカバリです。詳細については、「Amazon RDS for Db2 のレプリカの使用」および「Amazon RDS for Oracle でのリードレプリカの使用」を参照してください。

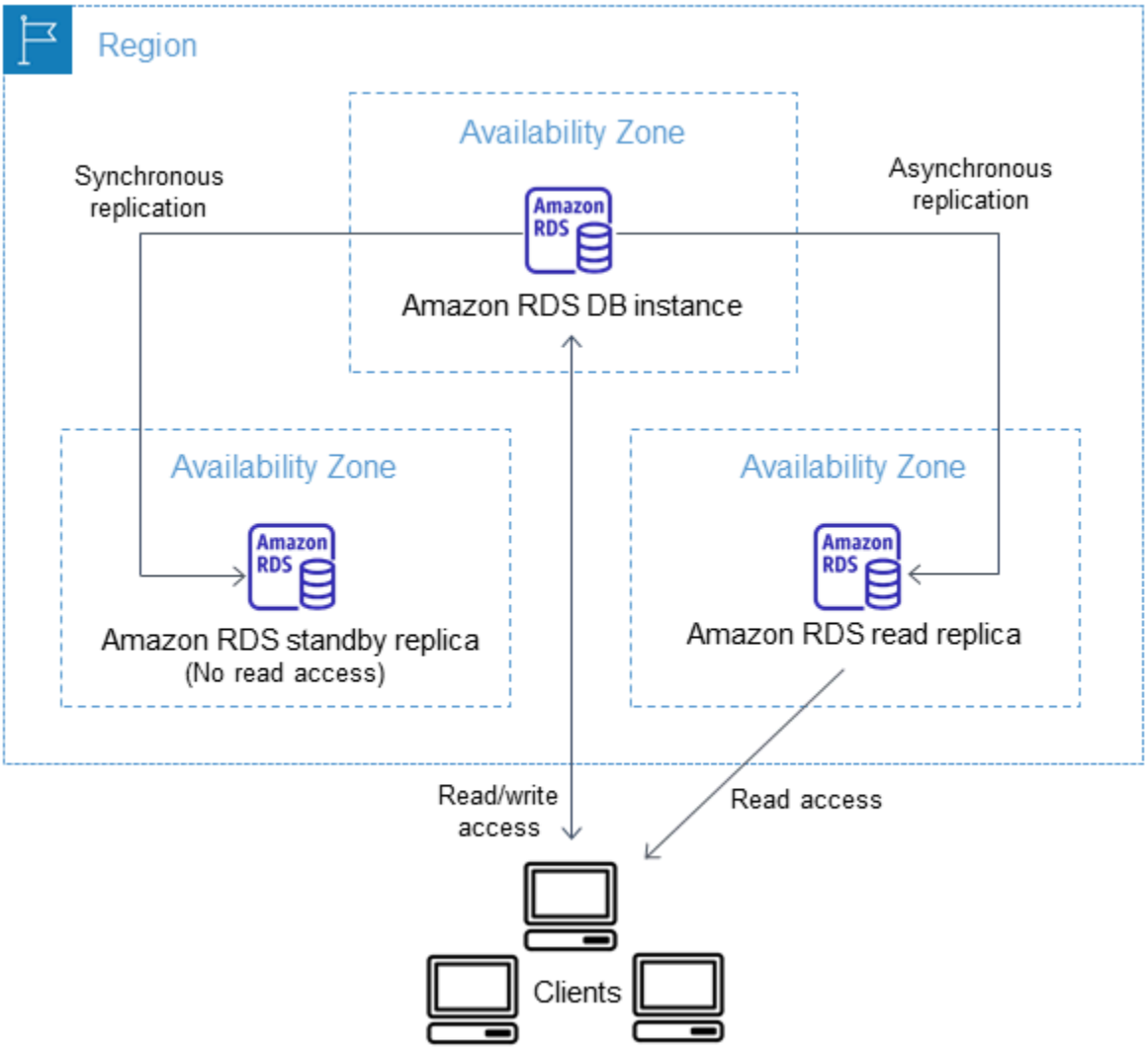
アプリケーションは、DB インスタンスの場合と同じ方法でリードレプリカに接続します。Amazon RDS では、ソース DB インスタンスからのすべてのデータベースがレプリケートされます。

リードレプリカは手動で作成する必要があります。RDS は、リードレプリカの自動スケーリングをサポートしていません。リードレプリカは、リード需要の変化に応じて自動的に追加または削除されます。

マルチ AZ 配置のリードレプリカ

マルチ AZ 配置で高可用性を持つように設定されたスタンバイレプリカもある DB インスタンスのリードレプリカを設定できます。スタンバイレプリカとのレプリケーションは同期的です。リードレプリカとは異なり、スタンバイレプリカは読み取りトラフィックを処理できません。

次のシナリオでは、クライアントは 1 つの AZ のプライマリ DB インスタンスへの読み取り/書き込みアクセス権を持っています。プライマリインスタンスは、更新を 2 番目の AZ のリードレプリカに非同期でコピーし、さらに 3 番目の AZ のスタンバイレプリカにも同期的にコピーします。クライアントはリードレプリカに対してのみ読み取りアクセス権を持ちます。



高可用性レプリカとスタンバイレプリカの詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

クロスリージョンリードレプリカ

リードレプリカは、そのプライマリ DB インスタンスとは異なる AWS リージョン に存在する場合があります。このような場合、Amazon RDS によりプライマリ DB インスタンスとリードレプリカ間の安全な通信チャネルが設定されます。Amazon RDS では、セキュリティグループエントリの追加など、安全なチャネルを有効にするために必要な AWS のセキュリティ設定を確立できます。クロスリージョンリードレプリカの詳細については、「別の でのリードレプリカの作成AWS リージョン」を参照してください。

この章の情報は、ソースの DB インスタンスと同じ AWS リージョン または別の AWS リージョンにおける Amazon RDS リードレプリカの作成に適用されます。以下の情報は、Amazon EC2 インス

タンスまたはオンプレミスで実行されているインスタンスでレプリケーションをセットアップする場合には適用されません。

リードレプリカのストレージタイプ

デフォルトでは、ソース DB インスタンスと同じストレージタイプのリードレプリカが作成されます。ただし、次の表に示すオプションに基づいて、ソース DB インスタンスと別のストレージタイプを持つリードレプリカを作成することもできます。

ソース DB インスタンスストレージのタイプ	ソース DB インスタンスストレージの割り当て	リードレプリカのストレージタイプのオプション
プロビジョンド IOPS	100 GiB - 64 TiB	プロビジョンド IOPS、汎用、マグネティック
汎用	100 GiB - 64 TiB	プロビジョンド IOPS、汎用、マグネティック
汎用	100 GiB 未満	汎用、マグネティック
マグネティック	100 GiB ~ 6 TiB	プロビジョンド IOPS、汎用、マグネティック
マグネティック	100 GiB 未満	汎用、マグネティック

> **Note**
>
> リードレプリカの割り当て済みストレージを増やす場合は、少なくとも 10% にする必要があります。10 パーセントに満たない単位で増やそうとすると、エラーになります。

レプリカからレプリカを作成する場合の制限事項

Amazon RDS では、循環レプリケーションはサポートされません。既存の DB インスタンスのレプリケーションソースとして機能するように DB インスタンスを設定することはできません。新しいリードレプリカは、既存の DB インスタンスからのみ作成することができます。例えば、**MySourceDBInstance** が **ReadReplica1** にレプリケートされる場合、**ReadReplica1** に再度レプリケートされるように **MySourceDBInstance** を設定することはできません。

MariaDB および RDS for MySQL、そして RDS for PostgreSQL の特定のバージョンでは、既存のリードレプリカからリードレプリカを作成することができます。例えば、既存のレプリカ **ReadReplica1** から新しいリードレプリカ **ReadReplica2** を作成できます。RDS for Db2、RDS for Oracle、RDS for SQL Server では、既存のリードレプリカからリードレプリカを作成することはできません。

レプリカを削除する際の注意事項

RDS はリードレプリカの自動スケーリングをサポートしていません。したがって、RDS は需要が増えてもレプリカの数を増やしたり、需要が減ってもレプリカの数を減らしたりすることはありません。リードレプリカが不要になった場合は、DB インスタンスを削除するのと同じメカニズムを使用して、リードレプリカを手動で削除します。同じ AWS リージョンのリードレプリカを削除せずにソース DB インスタンスを削除すると、各レプリカはスタンドアロン DB インスタンスに昇格されます。

DB インスタンスの削除については、「DB インスタンスを削除する」を参照してください。リードレプリカの昇格の詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。クロスリージョンリードレプリカのソース DB インスタンスの削除の関連情報については、「クロスリージョンレプリケーションに関する考慮事項」を参照してください。

DB エンジンのリードレプリカ間の違い

Amazon RDS DB エンジンごとにレプリケーションの実装方法は異なるため、いくつかの大きな違いを知っておく必要があります。

Db2

RDS for Db2 のレプリカには、以下の機能と動作があります。

- レプリケーション方法 – 物理レプリケーション。
- トランザクションログの消去 – 以下の条件が満たされると、RDS for Db2 はプライマリ DB インスタンスからログを消去します。
  - ログは 2 時間以上経過している。
  - アーカイブログに設定した保持時間が経過している。
  - RDS for Db2 は、すべてのレプリカ DB インスタンスにログを正常にレプリケートした。

  これは、同じ AWS リージョンの DB インスタンスとクロスリージョンの DB インスタンスの両方に適用されます。アーカイブログの保持時間の設定については、「rdsadmin.set_archive_log_retention」を参照してください。

- 書き込み可能なレプリカ – Db2 レプリカは物理コピーであり、Db2 はレプリカへの書き込みを許可しません。リードレプリカを昇格させることで、書き込み可能にすることができます。昇格させたレプリカには、昇格をリクエストした時点までにレプリケートされたデータが含まれます。
- バックアップ – RDS for Db2 リードレプリカでは、自動バックアップと手動スナップショットがサポートされています。
- 並列レプリケーション – アーカイブログデータは、常にプライマリデータベースからそのすべてのレプリカに並列に送信されます。
- スタンバイ状態 – スタンバイレプリカの主な用途は、クロスリージョンのディザスタリカバリです。詳細については、「Amazon RDS for Db2 のレプリカの使用」を参照してください。

MariaDB と MySQL

RDS for MariaDB および RDS for MySQL のリードレプリカには、次の機能と動作があります。

- レプリケーション方法 – 論理レプリケーション。
- トランザクションログの消去 – RDS for MariaDB および RDS for MySQL は、適用されていないバイナリログを保持します。
- 書き込み可能なレプリカ – MariaDB または MySQL のリードレプリカは書き込み可能にすることができます。
- バックアップ – RDS for MariaDB または RDS for MySQL のリードレプリカでは、自動バックアップと手動スナップショットがサポートされています。
- 並列レプリケーション – すべてのサポートされている MariaDB バージョンおよび MySQL のバージョンでは、並列レプリケーションのスレッドが許可されます。
- マウント状態 – サポートされていません。

Oracle

RDS for Oracle のリードレプリカには、次の機能と動作があります。

- レプリケーション方法 – 物理レプリケーション。
- トランザクションログの消去 – プライマリ DB インスタンスにクロスリージョンのリードレプリカがない場合、Amazon RDS for Oracle は、ソース DB インスタンスで最低 2 時間のトランザクションログを保持します。2 時間後、またはアーカイブログの保持時間設定のいずれか長い方の時間が経過すると、ログはソース DB インスタンスから削除されます。ログがデータベースに正常に

適用された場合にのみ、アーカイブログの保持時間設定が経過すると、ログはリードレプリカから削除されます。

プライマリ DB インスタンスには、1 つ以上のクロスリージョンのリードレプリカが存在する場合があります。その場合 Amazon RDS for Oracle は、ソース DB インスタンスのトランザクションログが転送され、すべてのクロスリージョンリードレプリカに適用されるまで保持します。

アーカイブログの保持時間の設定については、「アーカイブ REDO ログの保持」を参照してください。

- 書き込み可能なレプリカ – Oracle のリードレプリカは物理コピーであり、Oracle ではリードレプリカの書き込みは許可されません。リードレプリカを昇格させて書き込み可能にすることができます。昇格したリードレプリカには、昇格をリクエストされた時点までのレプリケートされたデータがあります。
- バックアップ – RDS for Oracle のリードレプリカでは、自動バックアップと手動スナップショットがサポートされています。
- 並列レプリケーション – REDO ログデータは、常にプライマリデータベースからそのすべてのリードレプリカに並列に送信されます。
- マウント状態 – マウントされたレプリカの主な用途は、クロスリージョンのディザスタリカバリです。マウントされたレプリカには、Active Data Guard ライセンスは必要ありません。詳細については、「Amazon RDS for Oracle でのリードレプリカの使用」を参照してください。

PostgreSQL

RDS for PostgreSQL のリードレプリカには、次の機能と動作があります。

- レプリケーション方法 – 物理レプリケーション。
- トランザクションログの消去 – PostgreSQL にはパラメータ `wal_keep_segments` があり、これにより、データをリードレプリカに提供するために、先書き口グ (WAL) ファイルをいくつ保持するかを指示します。パラメータ値は、保持するログの数を指定します。
- 書き込み可能なレプリカ – PostgreSQL のリードレプリカは物理コピーであり、PostgreSQL ではリードレプリカを書き込み可能にすることは許可されません。
- バックアップ – RDS for PostgreSQL のリードレプリカでは、手動スナップショットがサポートされています。リードレプリカの自動バックアップは、RDS for PostgreSQL 14.1 以降のバージョンでのみサポートされます。RDS for PostgreSQL 14.1 より前のバージョンの PostgreSQL リードレプリカでは、自動バックアップをオンにすることはできません。RDS for PostgreSQL 13 以前のバージョンでバックアップが必要な場合は、リードレプリカのスナップショットを作成します。

- 並列レプリケーション – PostgreSQL では、単一プロセスでレプリケーションを処理します。
- マウント状態 – サポートされていません。

SQL Server

RDS for SQL Server のリードレプリカには、次の機能と動作があります。

- レプリケーション方法 – 物理レプリケーション。
- トランザクションログの消去 – プライマリレプリカのトランザクションログファイルの仮想ログファイル (VLF) は、セカンダリレプリカで不要になったら切り捨てることができます。

  VLF は、レプリカでログレコードがハードニングされている場合にのみ非アクティブとしてマークできます。プライマリレプリカ内でのディスクサブシステムの速度に関係なく、トランザクションログは、最も遅いレプリカで強化されるまで VLF を保持します。
- 書き込み可能なレプリカ – SQL Server のリードレプリカは物理コピーであり、書き込みは許可されません。リードレプリカを昇格させて書き込み可能にすることができます。昇格したリードレプリカには、昇格をリクエストされた時点までのレプリケートされたデータがあります。
- バックアップ – RDS for SQL Server のリードレプリカでは、自動バックアップと手動スナップショットはサポートされていません。
- 並列レプリケーション – REDO ログデータは、常にプライマリデータベースからそのすべてのリードレプリカに並列に送信されます。
- マウント状態 – サポートされていません。

リードレプリカの作成

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、既存の DB インスタンスからリードレプリカを作成できます。リードレプリカは、レプリケーション元のソース DB インスタンスの DB インスタンス識別子である `SourceDBInstanceIdentifier` を指定することで作成できます。

リードレプリカを作成すると、Amazon RDS はソース DB インスタンスの DB スナップショットを取得し、レプリケーションを開始します。DB スナップショットオペレーションが始まると、ソース DB インスタンスでごく短時間の I/O 停止が発生します。通常、I/O 停止は約 1 秒続きます。ソース DB インスタンスがマルチ AZ 配置の場合は、I/O 停止を回避できます。スナップショットがセカンダリ DB インスタンスから取得されるためです。

アクティブな長時間実行トランザクションの場合、リードレプリカの作成プロセスに時間がかかることがあります。長時間実行トランザクションが完了してから、リードレプリカを作成することをお勧めします。同じソース DB インスタンスから複数のリードレプリカを同時に作成する場合、Amazon RDS は初回の作成アクションの開始時にスナップショットを 1 つだけ取得します。

リードレプリカが利用可能になると、そのボリュームは Amazon S3 からのデータブロックをバックグラウンドでロードし続けます。この期間中、リードレプリカは完全に動作しますが、初期化が完了するまでパフォーマンスが最大にならない場合があります。ボリュームごとの初期化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。初期化中、`StorageOperationStatus` は `Initializing` に設定され、初期化が完了すると両方のフィールドは存在しなくなります。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

リードレプリカを作成するときは、いくつかの考慮事項があります。最初に、バックアップ保持期間を 0 以外の値に設定することで、ソース DB インスタンスで自動バックアップを有効にする必要があります。この要件は、別のリードレプリカのソース DB インスタンスであるリードレプリカにも適用されます。RDS for MySQL のリードレプリカで自動バックアップを有効にするには、まずリードレプリカを作成し、次にそのリードレプリカを変更して自動バックアップを有効にします。

> **Note**
>
> AWS リージョン 内では、Amazon VPC に基づくすべてのリードレプリカをソース DB インスタンスと同じ仮想プライベートクラウド (VPC) に作成することを強くお勧めします。リードレプリカをソース DB インスタンスとは異なる VPC に作成すると、クラスレスドメイン間ルーティング (CIDR) の範囲がレプリカと RDS システムとの間で重複する可能性があります。CIDR が重複すると、レプリカが不安定になり、レプリカに接続するアプリケーションに悪影響を及ぼす可能性があります。リードレプリカの作成時にエラーが発生した場合は、別のターゲット DB サブネットグループを選択します。詳細については、「VPC 内の DB インスタンスの使用」を参照してください。
>
> コンソールまたは AWS CLI を使用して別の AWS アカウント でリードレプリカを直接作成する方法はありません。

コンソール

ソース DB インスタンスからリードレプリカを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. リードレプリカのソースとして使用する DB インスタンスを選択します。
4. [アクション] で [リードレプリカの作成] を選択します。
5. [DB インスタンス識別子] に、リードレプリカの名前を入力します。
6. インスタンス設定を選択します。リードレプリカでもソース DB インスタンスと同等以上の DB インスタンスクラスとストレージタイプを使用することをお勧めします。
7. AWS リージョン では、リードレプリカの作成先のリージョンを指定します。
8. [ストレージ] では、割り当てられたストレージのサイズや、ストレージの自動スケーリングを使用するかどうかを指定します。

   ソース DB インスタンスが最新のストレージ設定になっていない場合は、[ストレージファイルシステム設定のアップグレード] オプションを使用できます。この設定を有効にすると、リードレプリカのファイルシステムが適切な設定にアップグレードされます。詳細については、「the section called “ストレージファイルシステムのアップグレード”」を参照してください。
9. [可用性] では、レプリカのフェイルオーバーをサポートするために別のアベイラビリティーゾーンにレプリカのスタンバイを作成するかどうかを選択します。

   > **Note**
   >
   > リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。
10. 他の DB インスタンスの設定を指定します。各可用性設定の詳細については、「DB インスタンスの設定」を参照してください。
11. 暗号化されたリードレプリカを作成するには、[その他の設定] を展開してから、次の設定を指定します。

    a. [暗号化の有効化] を選択します。

    b. [AWS KMS key] では、KMS キーの AWS KMS key 識別子を選択します。

> **Note**
>
> ソース DB インスタンスは暗号化する必要があります。DB インスタンスの暗号化については、「Amazon RDS リソースの暗号化」を参照してください。

12. [Create read replica] を選択します。

リードレプリカが作成されると、RDS コンソールの [Databases] (データベース) ページに表示されます。[Role] (ロール) 列に [Replica] (レプリカ) が表示されます。

AWS CLI

ソース DB インスタンスからリードレプリカを作成するには、AWS CLI コマンド create-db-instance-read-replica を使用します。この例では、割り当てられたストレージサイズを設定し、ストレージの自動スケーリングを有効にします。また、ファイルシステムを適切な設定にアップグレードします。

他の設定を指定できます。各設定の詳細については、「DB インスタンスの設定」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance \
    --allocated-storage 100 \
    --max-allocated-storage 1000 \
    --upgrade-storage-config
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000 ^
```

```
    --upgrade-storage-config
```

RDS API

ソースの Db2、MySQL、MariaDB、Oracle、PostgreSQL、または SQL Server DB インスタンスからリードレプリカを作成するには、以下の必須パラメータを指定して Amazon RDS API の CreateDBInstanceReadReplica オペレーションを呼び出します。

- `DBInstanceIdentifier`
- `SourceDBInstanceIdentifier`

> **Note**
>
> RDS for Db2 スタンバイレプリカを作成するには、オプションの `ReplicaMode` オペレーションを `mounted` に設定します。

リードレプリカをスタンドアロン DB インスタンスに昇格させる

リードレプリカをスタンドアロン DB インスタンスに昇格させることができます。ソース DB インスタンスに複数のリードレプリカがある場合、いずれかのリードレプリカを DB インスタンスに昇格させても、他のレプリカには影響が及びません。

リードレプリカを昇格させると、RDS によって DB インスタンスが使用可能になる前に DB インスタンスが再起動されます。リードレプリカのサイズによっては、昇格プロセスが完了するまで数分以上かかる場合があります。

リードレプリカを昇格させるユースケース

リードレプリカをスタンドアロン DB インスタンスに昇格させる理由には、次のようなものがあります。

- 障害復旧の実装 – プライマリ DB インスタンスに障害が発生した場合、データ復旧スキームとして、リードレプリカを昇格させることができます。このアプローチは、同期的なレプリケーション、自動障害検出、フェイルオーバーを補完します。

非同期レプリケーションの影響と制限を承知の上で、データ復旧にリードレプリカの昇格が必要と判断した場合に限り、昇格を行ってください。これを行うには、最初にリードレプリカを作成し、次にプライマリ DB インスタンスで障害をモニタリングします。障害が発生した場合、以下の作業を行います。

1. リードレプリカを昇格させます。
2. 昇格された DB インスタンスにデータベーストラフィックを向けます。
3. 昇格された DB インスタンスとの置き換えリードレプリカをソースとして作成します。

- ストレージ設定のアップグレード — ソース DB インスタンスが推奨ストレージ設定になっていない場合は、インスタンスのリードレプリカを作成し、ストレージファイルシステム設定をアップグレードできます。このオプションは、リードレプリカのファイルシステムを適切な設定に移行させます。次に、リードレプリカをスタンドアロンインスタンスに昇格させることができます。

  このオプションを使用すると、古い 32 ビットファイルシステムのストレージとファイルサイズのスケーリング制限を克服できます。詳細については、「the section called “ストレージファイルシステムのアップグレード”」を参照してください。

  このオプションは、ソース DB インスタンスが最新のストレージ設定になっていない場合、または同じリクエスト内で DB インスタンスクラスを変更する場合にのみ使用できます。

- シャーディング – シャーディングでは、「share-nothing」アーキテクチャを採用しており、基本的に大きいデータベースが複数の小さいデータベースに分割されます。データベースを分割する一般的な方法としては、同じクエリに結合されていないテーブルを切り離して別のホスト上に配置します。別の方法としては、複数のホスト間でテーブルを複製し、ハッシュアルゴリズムを使用してどのホストで特定の更新を受け取るかを指定します。各シャード (小さいデータベース) に対応するリードレプリカを作成し、スタンドアロンシャードに変換することを決定したら昇格させることができます。次に、要件に応じて、各シャードのテーブルのキースペース (行を分割する場合) または分布を分割できます。

- DDL 操作の実行 (MySQL および MariaDB のみ) – インデックスの作成や再構築などの DDL 操作には時間がかかることがあり、DB インスタンスのパフォーマンスが大幅に低下する可能性があります。これらのオペレーションは、MySQL リードレプリカや MariaDB リードレプリカがプライマリ DB インスタンスと同期されたら、これらのリードレプリカに対して実行できます。次に、リードレプリカを昇格させ、昇格されたインスタンスを使用するようにアプリケーションに指示できます。

> **Note**
>
> リードレプリカが RDS for Oracle DB インスタンスの場合は、昇格の代わりにスイッチオーバーを実行できます。スイッチオーバーでは、ソース DB インスタンスが新しいレプリカになり、レプリカが新しいソース DB インスタンスになります。詳細については、「Oracle Data Guard のスイッチオーバー操作の実行」を参照してください。

昇格されたリードレプリカの特徴

リードレプリカを昇格させると、リードレプリカとして機能しなくなり、スタンドアロン DB インスタンスになります。新しいスタンドアロン DB インスタンスには、次の特徴があります。

- スタンドアロン DB インスタンスは、昇格前のリードレプリカのオプショングループとパラメータグループを保持します。
- 新しい DB インスタンスからリードレプリカを作成して、ポイントインタイム復元オペレーションを実行できます。
- 昇格した DB インスタンスはリードレプリカではなくなったため、レプリケーションターゲットとしては使用できません。

リードレプリカを昇格させるための前提条件

リードレプリカを昇格する前に、次のことを実行してください。

- バックアップ戦略を確認します。
  - バックアップを有効にし、少なくとも 1 つのバックアップを完了することをお勧めします。バックアップ期間は、以前のバックアップ以降のデータベースに対する変更数の関数です。
  - リードレプリカでバックアップを有効にしている場合は、自動バックアップウィンドウを構成して、毎日のバックアップがリードレプリカの昇格を妨げないようにします。
  - リードレプリカが `backing-up` ステータスになっていないことを確認します。この状態にあるリードレプリカを昇格させることはできません。
- プライマリ DB インスタンスに対するトランザクションの書き込みを停止し、すべての更新がリードレプリカに反映されるまで待ちます。

  データベースの更新は、プライマイ DB インスタンスで行われた後にリードレプリカで行われます。レプリケーションのラグは大きく異なる可能性があります。`Replica Lag` メトリクスを使用して、リードレプリカに対するすべての更新の時期を確認できます。

- (MySQL および MariaDB のみ) MySQL または MariaDB のリードレプリカに変更を加えるには、リードレプリカの DB パラメータグループで `read_only` パラメータを `0` に設定する必要があります。次に、インデックスの作成など、必要なすべての DDL 操作をリードレプリカで実行します。リードレプリカに対するアクションの実行は、プライマリ DB インスタンスのパフォーマンスには影響しません。

リードレプリカの昇格: 基本的な手順

以下のステップは、DB インスタンスにリードレプリカを昇格させる一般的なプロセスを示しています。

1. Amazon RDS コンソールの [昇格] オプション、AWS CLI コマンドの `promote-read-replica`、または Amazon RDS API の `PromoteReadReplica` オペレーションを使用して、リードレプリカをプロモートします。

   > **Note**
   >
   > 昇格プロセスの完了までには数分かかります。リードレプリカを昇格させると、レプリケーションが停止され、リードレプリカが再起動されます。再起動が完了すると、リードレプリカは新しい DB インスタンスとして使用可能になります。

2. (オプション) 新しい DB インスタンスをマルチ AZ 配置に変更します。詳細については、「Amazon RDS DB インスタンスを変更する」および「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

コンソール

リードレプリカをスタンドアロン DB インスタンスに昇格させるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

   [Databases (データベース)] ペインが表示されます。各リードレプリカには、[Role (ロール)] 列に [Replica (レプリカ)] があります。
3. 昇格させるリードレプリカを選択します。
4. [アクション] で、[Promote (昇格)] を選択します。

5. [リードレプリカの昇格] ページで、新しく昇格された DB インスタンスのバックアップ保持期間とバックアップウィンドウを入力します。
6. すべての設定が正しいことを確認したら、[Continue] を選択します。
7. 確認ページで、[Promote Read Replica] を選択します。

AWS CLI

リードレプリカをスタンドアロン DB インスタンスに昇格させるには、AWS CLI `promote-read-replica` コマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds promote-read-replica \
    --db-instance-identifier myreadreplica
```

Windows の場合:

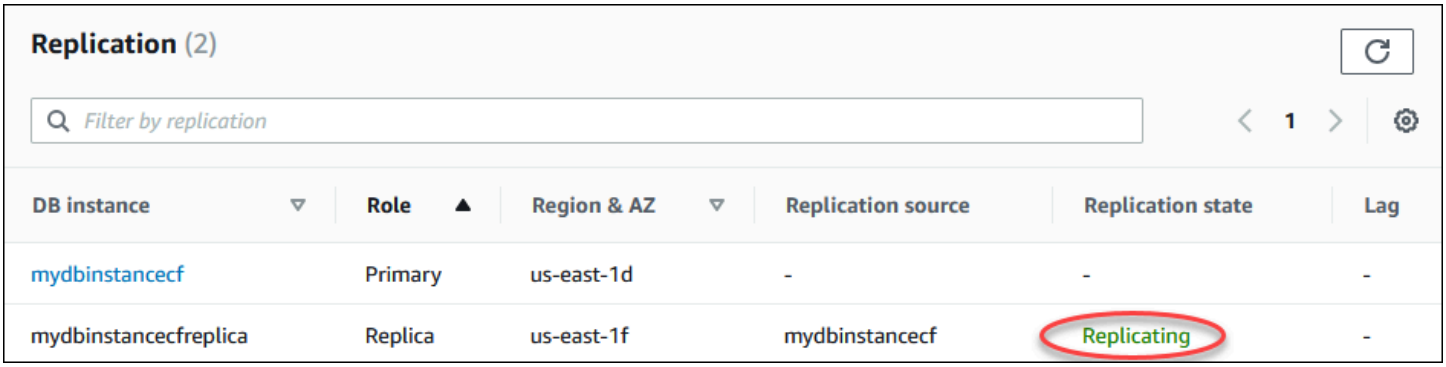
```
aws rds promote-read-replica ^
    --db-instance-identifier myreadreplica
```

RDS API

リードレプリカをスタンドアロン DB インスタンスに昇格するには、必須のパラメータ `DBInstanceIdentifier` を指定して Amazon RDS API の `PromoteReadReplica` オペレーションを呼び出します。

リードレプリケーションのモニタリング

リードレプリカのステータスは、さまざまな方法でモニタリングできます。Amazon RDS コンソールは、リードレプリカの詳細にある [Connectivity & security] (接続性とセキュリティ) タブの [Replication] (レプリケーション) セクションでリードレプリカのステータスを表示します。リードレプリカの詳細を表示するには、Amazon RDS コンソールの DB インスタンスのリストでリードレプリカの名前を選択します。



AWS CLI `describe-db-instances` コマンドまたは Amazon RDS API `DescribeDBInstances` オペレーションを使用して、リードレプリカのステータスを確認することもできます。

リードレプリカのステータスは、以下のいずれかです。

- replicating – リードレプリカが正常にレプリケーションされています。
- replication degraded (SQL Server および PostgreSQL のみ) – レプリカはプライマリインスタンスからデータを受信していますが、1 つ以上のデータベースが更新を取得していない可能性があります。これは、レプリカが新しく作成されたデータベースをセットアップしているときなどに発生します。また、ブルー/グリーンデプロイのブルー環境で、サポートされていない DDL やラージオブジェクトの変更が行われた場合にも発生する可能性があります。

  パフォーマンスが低下した状態でエラーが発生しない限り、ステータスは `replication degraded` から `error` に移行しません。
- error – レプリケーションでエラーが発生しました。Amazon RDS コンソールの [Replication Error] またはイベントを口グを確認して、正確なエラーについて調べます。レプリケーションエラーのトラブルシューティングの詳細については、「MySQL リードレプリカに関する問題のトラブルシューティング」を参照してください。
- terminated (MariaDB、MySQL、または PostgreSQL のみ) – レプリケーションが終了しました。これは、手動またはレプリケーションエラーによってレプリケーションが連続する 30 日超停止した場合に発生します。この場合、Amazon RDS はプライマリ DB インスタンスとすべてのリードレプリカ間のレプリケーションを終了します。Amazon RDS は、ソース DB インスタンスでのストレージ要件の増加と長いフェイルオーバー時間を防ぐためにこれを行います。

  レプリケーションが中断すると、ログに書き込まれるエラーメッセージの量が増えてログのサイズと数が増加するため、ストレージに影響が及ぶ可能性があります。さらに、レプリケーションが中断すると、Amazon RDS が復旧中に大量の口グを保持して処理するのに必要な時間が原因で、災害対策にも影響が及ぶ可能性があります。

- 終了 (Oracle のみ) – レプリケーションは終了します。これは、リードレプリカに十分なストレージが残っていないため、レプリケーションが 8 時間以上停止した場合に発生します。この場合、Amazon RDS はプライマリ DB インスタンスと影響を受けたリードレプリカ間のレプリケーションを終了します。このステータスは終了状態であり、リードレプリカを再作成する必要があります。
- stopped (MariaDB または MySQL のみ) – お客様がリクエストを開始したため、レプリケーションが停止しました。
- replication stop point set (MySQL のみ) – お客様が開始した停止ポイントが mysql.rds_start_replication_until ストアドプロシージャを使用して設定され、レプリケーションが進行中です。
- replication stop point reached (MySQL のみ) – お客様が開始した停止ポイントが mysql.rds_start_replication_until ストアドプロシージャを使用して設定され、レプリケーションは停止ポイントに到達したために停止しました。

DB インスタンスのレプリケート先の場所を確認でき、その場合は、そのレプリケーションステータスを確認できます。RDS コンソールの [Databases] (データベース) ページの [Role] (ロール) 列に [Primary] (プライマリ) と表示されます。その DB インスタンス名を選択します。詳細ページの [Connectivity & security] (接続とセキュリティ) タブの [Replication] (レプリケーション) の下に、レプリケーションの状態が表示されます。

レプリケーションラグのモニタリング

Amazon CloudWatch のレプリケーションのラグをモニタリングするには、Amazon RDS `ReplicaLag` メトリクスを表示します。

Db2 の場合、`ReplicaLag` メトリクスは、遅延しているデータベースの最大遅延 (秒単位) です。例えば、5 秒遅れているデータベースと 10 秒遅れているデータベースがある場合、`ReplicaLag` は 10 秒です。高可用性ディザスタリカバリ (HADR) の使用可能なステータスがないデータベースは、計算に含まれません。

MariaDB と MySQL の場合、`ReplicaLag` メトリクスでは、`Seconds_Behind_Master` コマンドの `SHOW REPLICA STATUS` フィールドの値がレポートされます。MySQL と MariaDB のレプリケーション遅延の一般的な原因は以下のとおりです。

- ネットワークが停止している。
- リードレプリカで、インデックスがあるテーブルに書き込んでいる。`read_only` パラメータがリードレプリカで 0 に設定されていない場合、レプリケーションが中断されることがあります。

- MyISAM などの非トランザクションストレージエンジンを使用している。レプリケーションは、MariaDB の MySQL および InnoDB ストレージエンジンの XtraDB ストレージエンジンでのみサポートされます。

> **Note**
>
> MariaDB の旧バージョンは、`SHOW SLAVE STATUS` ではなく `SHOW REPLICA STATUS` を使用していました。10.5 より前の MariaDB バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

`ReplicaLag` メトリクスが 0 に達すると、レプリカはプライマリ DB インスタンスに追いついています。`ReplicaLag` メトリクスにより `-1` が返された場合、レプリケーションは現在アクティブではありません。`ReplicaLag = -1` は `Seconds_Behind_Master = NULL` と同等です。

Oracle で、`ReplicaLag` メトリクスは、`Apply Lag` 値と、現在の時刻と適用するラグの `DATUM_TIME` 値の差異の合計です。`DATUM_TIME` 値は、リードレプリカがソース DB インスタンスから最後にデータを受信した時刻です。詳細については、Oracle ドキュメントの「V$DATAGUARD_STATS」を参照してください。

SQL Server の場合、`ReplicaLag` メトリクスは、遅延しているデータベースの最大ラグ (秒単位) です。例えば、5 秒遅れているデータベースと 10 秒遅れているデータベースがある場合、`ReplicaLag` は 10 秒です。`ReplicaLag` メトリクスは、次のクエリの値を返します。

```
SELECT MAX(secondary_lag_seconds) max_lag FROM sys.dm_hadr_database_replica_states;
```

詳細については、Microsoft ドキュメントの「secondary_lag_seconds」を参照してください。

`ReplicaLag` はレプリカのセットアップ中やリードレプリカが `-1` 状態にあるときなど、RDS でラグを確認できない場合は `error` を返します。

> **Note**
>
> 新しいデータベースは、リードレプリカでアクセス可能になるまでラグ計算に含まれません。

PostgreSQL では、`ReplicaLag` メトリクスは次のクエリの値を返します。

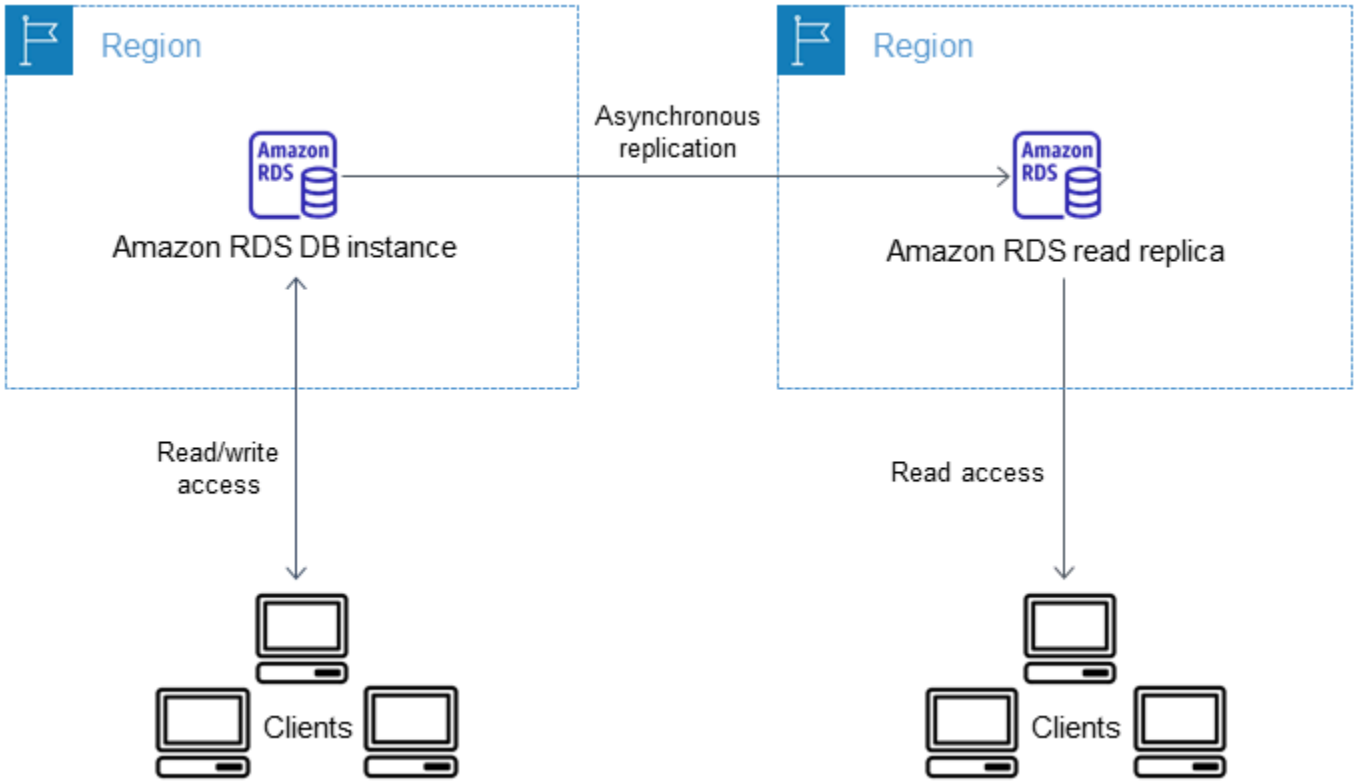
```
SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS reader_lag
```

PostgreSQL バージョン 9.5.2 以降では、ソースインスタンスで保持される先書きログ (WAL) の管理に物理レプリケーションスロットを使用します。クロスリージョンリードレプリカインスタンスごとに、Amazon RDS は物理レプリケーションスロットを作成し、インスタンスに関連付けます。2 つ Amazon CloudWatch メトリクス `Oldest Replication Slot Lag` と `Transaction Logs Disk Usage` では、WAL データの受信について最も長い遅延が発生しているレプリカまでの距離と、WAL データに使用されているストレージの量が表示されます。クロスリージョンリードレプリカで著しい遅延が発生しているときは、`Transaction Logs Disk Usage` の値を大幅に増やすことができます。

CloudWatch を使用した DB インスタンスのモニタリングの詳細については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

別の でのリードレプリカの作成AWS リージョン

Amazon RDS では、リードレプリカをソース DB インスタンスとは異なる AWS リージョン に作成できます。



別の AWS リージョン にリードレプリカを作成して、以下を実行します。

- 災害対策機能が向上します。
- ユーザーに近い AWS リージョン への読み取りオペレーションをスケールします。
- ある AWS リージョン のデータセンターから別の AWS リージョン のデータセンターへの移行が簡単になります。

ソースインスタンスとは異なる AWS リージョン でリードレプリカを作成する作業は、同じ AWS リージョン でレプリカを作成する作業と似ています。AWS マネジメントコンソール の使用、`create-db-instance-read-replica` コマンドの実行、または `CreateDBInstanceReadReplica` API オペレーションの呼び出しを行うことができます。

> **Note**
>
> ソース DB インスタンスとは異なる AWS リージョン に暗号化されたリードレプリカを作成するには、ソース DB インスタンスを暗号化する必要があります。

トピック

- リージョンとバージョンの可用性
- クロスリージョンリードレプリカの作成
- Amazon RDS でのクロスリージョンレプリケーションのしくみ
- クロスリージョンレプリケーションに関する考慮事項
- クロスリージョンレプリケーションのコスト

リージョンとバージョンの可用性

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。リージョン間レプリケーションによるバージョンとリージョンの可用性の詳細については、「Amazon RDS でのクロスリージョンリードレプリカでサポートされているリージョンと DB エンジン」を参照してください。

クロスリージョンリードレプリカの作成

異なる AWS リージョン でソース Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle、または PostgreSQL DB インスタンスからリードレプリカを作成する手順を以下に示します。

コンソール

AWS マネジメントコンソール を使用して、複数の AWS リージョン にわたるリードレプリカを作成できます。

複数の AWS リージョン にわたるリードレプリカをコンソールで作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. リードレプリカのソースとして使用する Db2、 MariaDB、Microsoft SQL Server、MySQL、Oracle、または PostgreSQL DB インスタンスを選択します。
4. [アクション] で [リードレプリカの作成] を選択します。
5. [DB インスタンス識別子] に、リードレプリカの名前を入力します。
6. [送信先リージョン] を選択します。
7. 使用するインスタンス仕様を選択します。リードレプリカでも同等以上の DB インスタンスクラスとストレージタイプを使用することをお勧めします。
8. 別の AWS リージョン で暗号化されたリードレプリカを作成するには

    a. [暗号化の有効化] を選択します。

    b. [AWS KMS key] では、作成先 AWS リージョン の KMS キーの AWS KMS key 識別子を選択します。

    > **Note**
    >
    > 暗号化されたリードレプリカを作成するには、ソース DB インスタンスを暗号化する必要があります。DB インスタンスの暗号化については、「Amazon RDS リソースの暗号化」を参照してください。

9. ストレージの自動スケーリングなど、他のオプションを選択します。
10. [Create read replica] を選択します。

AWS CLI

異なる AWS リージョンでソースの Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle、または PostgreSQL DB インスタンスからリードレプリカを作成するには、`create-db-instance-`

read-replica コマンドを使用できます。この場合、リードレプリカ (作成先のリージョン) が必要な AWS リージョン の create-db-instance-read-replica を使用し、ソース DB インスタンスの Amazon リソースネーム (ARN) を指定します。ARN は、Amazon Web Services で作成したリソースを一意に識別します。

例えば、ソース DB インスタンスが US East (N. Virginia) リージョンにある場合、ARN は次の例のようになります。

```
arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

ARN の詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

ソース DB インスタンスから異なる AWS リージョン にリードレプリカを作成するには、作成先の AWS リージョン で AWS CLI create-db-instance-read-replica コマンドを使用します。別の AWS リージョン でリードレプリカを作成するには、次のパラメータが必要です。

- --region – リードレプリカが作成される作成先の AWS リージョン。
- --source-db-instance-identifier – ソース DB インスタンスの DB インスタンス識別子です。この識別子は、コピー元 AWS リージョン の ARN 形式である必要があります。
- --db-instance-identifier – 作成先の のリードレプリカの識別子。AWS リージョン

Exampleクロスリージョンリードレプリカの

次のコードは、米国西部 (オレゴン) リージョン内のソース DB インスタンスから US East (N. Virginia) リージョン内にリードレプリカを作成します。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --region us-west-2 ^
```

```
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

別の AWS リージョン で暗号化されたリードレプリカを作成するには、次のパラメータも必要です。

- `--kms-key-id` - 作成先 AWS リージョン でリードレプリカの暗号化に使用する KMS キーの AWS KMS key 識別子。

Example暗号化されたクロスリージョンリードレプリカの

次のコードは、米国西部 (オレゴン) リージョン内のソース DB インスタンスから US East (N. Virginia) リージョン内に暗号化されたリードレプリカを作成します。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
 \
    --kms-key-id my-us-west-2-key
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:mydbinstance
 ^
    --kms-key-id my-us-west-2-key
```

`--source-region` オプションは、AWS GovCloud (米国東部) と AWS GovCloud (米国西部) リージョン間で暗号化されたリードレプリカを作成する場合に必要です。`--source-region` には、ソース DB インスタンスの AWS リージョン を指定します。

`--source-region` を指定しない場合、`--pre-signed-url` の値を指定する必要があります。署名付きの URL は、ソースの AWS リージョン で呼び出される `create-db-instance-read-replica` コマンドに対する、署名バージョン 4 で署名されたリクエストを含む URL です。`pre-`

`signed-url` オプションの詳細については、AWS CLI コマンドリファレンスの「create-db-instance-read-replica」を参照してください。

RDS API

異なる AWS リージョンでソースの Db2、MariaDB、MySQL、Microsoft SQL Server、Oracle、または PostgreSQL DB インスタンスからリードレプリカを作成するには、Amazon RDS API オペレーション CreateDBInstanceReadReplica を呼び出すことができます。この場合、リードレプリカが必要な AWS リージョン (作成先のリージョン) の CreateDBInstanceReadReplica を呼び出し、ソース DB インスタンスの Amazon リソースネーム (ARN) を指定します。ARN は、Amazon Web Services で作成したリソースを一意に識別します。

ソース DB インスタンスから異なる AWS リージョン に暗号化されたリードレプリカを作成するには、作成先の AWS リージョン で Amazon RDS API の `CreateDBInstanceReadReplica` オペレーションを使用できます。暗号化されたリードレプリカを別の AWS リージョンに作成するには、`PreSignedURL` の値を指定する必要があります。`PreSignedURL` には、リードレプリカが作成されるソース AWS リージョンで呼び出す `CreateDBInstanceReadReplica` オペレーションのリクエストが含まれている必要があります。`PreSignedUrl` の詳細については、「`CreateDBInstanceReadReplica`」を参照してください。

例えば、ソース DB インスタンスが US East (N. Virginia) リージョンにある場合、ARN は以下のような内容です。

```
arn:aws:rds:us-east-1:123456789012:db:mydbinstance
```

ARN の詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

Example

```
https://us-west-2.rds.amazonaws.com/
    ?Action=CreateDBInstanceReadReplica
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBInstanceReadReplica
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBInstanceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%123456789012%25253Adb%25253Amydbinstance
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4%2526SourceDBInstanceIdentifier%253Darn%25253Aaws
%25253Ards%25253Aus-west-2%25253A123456789012%25253Ainstance%25253Amydbinstance
```

```
        %2526Version%253D2014-10-31
        %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
        %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
        %2526X-Amz-Date%253D20161117T215409Z
        %2526X-Amz-Expires%253D3600
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &DBInstanceIdentifier=myreadreplica
    &SourceDBInstanceIdentifier=&region-arn;rds:us-east-1:123456789012:db:mydbinstance
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-20T22%3A06%3A23.624Z
    &AWSAccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

Amazon RDS でのクロスリージョンレプリケーションのしくみ

Amazon RDS は、次のプロセスを使用してクロスリージョンリードレプリカを作成します。関係する AWS リージョン とデータベースのデータ量によっては、このプロセスは完了までに数時間かかることがあります。クロスリージョンリードレプリカを作成する際、この情報を使用してプロセスの進行状況を確認できます。

1. Amazon RDS は、ソース DB インスタンスをレプリケーションソースとして設定し始め、ステータスを modifying に設定します。
2. Amazon RDS は、作成先の AWS リージョン で指定されたリードレプリカをセットアップし始め、ステータスを creating に設定します。
3. Amazon RDS は、ソース AWS リージョン でソース DB インスタンスの自動 DB スナップショットを作成します。DB スナップショット名の形式は、`rds:<InstanceID>-<timestamp>` です。ここで、`<InstanceID>` はソースインスタンスの識別子で、`<timestamp>` はコピーの開始日時です。例えば、`rds:mysourceinstance-2013-11-14-09-24` はインスタンス `mysourceinstance` から `2013-11-14-09-24` に作成されたことを表します。自動 DB スナップショットの作成中、ソース DB インスタンスのステータスが modifying のままになり、リードレプリカのステータスが creating のままになります。DB スナップショットのステータスは creating です。コンソールの DB スナップショットページの進行状況列には、DB スナップショット作成の進行状況が報告されます。DB スナップショットが完成すると、DB スナップショットとソース DB インスタンスの両方のステータスが available に設定されます。

4. Amazon RDS は、初期データ転送用のクロスリージョンスナップショットコピーを開始します。スナップショットコピーは、転送先 AWS リージョン の自動スナップショットとして、creating のステータスで一覧表示されます。名前はソース DB スナップショットと同じです。DB スナップショットの進行状況列には、コピーの進行状況が示されます。コピーが完了すると、DB スナップショットコピーのステータスが available に設定されます。
5. 次に、Amazon RDS はリードレプリカで初期データロード用のコピーされた DB スナップショットを使用します。このフェーズの間、リードレプリカは転送先の DB インスタンスの一覧に、creating のステータスで表示されます。ロードが完了すると、リードレプリカのステータスが available に設定され、DB スナップショットコピーが削除されます。
6. リードレプリカが available ステータスに達すると、Amazon RDS は、リードレプリカ作成操作が開始されてからソースインスタンスに加えられた変更のレプリケーションから始めます。このフェーズでは、リードレプリカのレプリケーション遅延時間が 0 より大きくなります。

   レプリケーションのラグタイムについては、「リードレプリケーションのモニタリング」を参照してください。

クロスリージョンレプリケーションに関する考慮事項

AWS リージョン 内でレプリケーションを実行する際の考慮事項はすべて、クロスリージョンレプリケーションに適用されます。AWS リージョン 間のレプリケーションには、他にも次の考慮事項が適用されます。

- ソース DB インスタンスでは、複数の AWS リージョン にクロスリージョンリードレプリカを作成できます。ソース VPC のアクセスコントロールリスト (ACL) のエントリ数に制限があるため、5 つを超えるクロスリージョンリードレプリカの DB インスタンスについては RDS は保証できません。
- アカウントごとに 1 つの送信先 AWS リージョンに進行中のクロスリージョンリードレプリカリクエストを最大 20 件同時に作成できます。同じ送信先 AWS リージョンに追加のクロスリージョンリードレプリカ作成リクエストを送信するのに、これらのリクエストが完了するのを待機する必要があります。
- GovCloud (米国東部) と GovCloud (米国西部) のリージョンの間ではレプリケーションできますが、GovCloud (米国) との間ではレプリケーションできません。
- 以下の RDS エンジンでは、ソース Amazon RDS DB インスタンスが別の Amazon RDS DB インスタンスのリードレプリカでない場合のみ、クロスリージョン Amazon RDS リードレプリカを作成できます。
  - RDS for Db2

  - RDS for SQL Server
  - RDS for Oracle
  - RDS for PostgreSQL バージョン 14.1 未満

  この制限は、RDS for PostgreSQL バージョン 14.1 以降、RDS for MariaDB、RDS for MySQL を実行している DB インスタンスには適用されません。
- リードレプリカがソースインスタンスとは異なる AWS リージョン にある場合は、高いレベルのラグタイムが発生することが予想されます。リージョンのデータセンター間のネットワークチャネルの方が長くなったために、このようなラグタイムが発生します。
- クロスリージョンリードレプリカでは、`--db-subnet-group-name` パラメータを指定する create read replica コマンドのいずれかで、同じ VPC の DB サブネットグループを指定する必要があります。
- 多くの場合、リードレプリカでは、指定された DB エンジンのデフォルトの DB パラメータグループおよび DB オプショングループが使用されます。

  Db2 DB エンジンの場合、レプリカのカスタムパラメータグループを指定する必要があります。BYOL モデルを使用するレプリカの場合、このカスタムパラメータグループには IBM Site ID と IBM Customer ID を含める必要があります。レプリカのカスタムパラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して指定できます。詳細については、「Db2 の Bring-Your-Own-License (BYOL) の IBM ID」を参照してください。

  MySQL エンジンと Oracle DB エンジンの場合、リードレプリカのカスタムパラメータグループは、AWS CLI コマンド create-db-instance-read-replica の `--db-parameter-group-name` オプションで指定できます。AWS マネジメントコンソール を使用している場合、カスタムパラメータグループを指定することはできません。
- リードレプリカは、デフォルトのセキュリティグループを使用します。
- Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle DB の各インスタンスの場合、クロスリージョンリードレプリカのソース DB インスタンスが削除されると、リードレプリカが昇格します。
- PostgreSQL DB インスタンスの場合、クロスリージョンリードレプリカのソース DB インスタンスが削除されると、レプリケーションのステータスは `terminated` に設定されます。リードレプリカは昇格しません。

  リードレプリカを手動で昇格させるか、削除する必要があります。

クロスリージョンリードレプリカのリクエスト

ソースリージョンと通信してクロスリージョンリードレプリカの作成をリクエストするには、リクエスタ (IAM ロールまたは IAM ユーザー) がソース DB インスタンスとソースリージョンへのアクセス権を持っている必要があります。

リクエスタの IAM ポリシーの特定の条件により、リクエストが失敗する可能性があります。次の例では、ソース DB インスタンスが 米国東部 (オハイオ) にあり、リードレプリカが US East (N. Virginia) に作成されていることを前提としています。これらの例は、リクエストが失敗する原因となるリクエスタの IAM ポリシー内の条件を示しています。

- リクエスタのポリシーには、aws:RequestedRegion の条件があります。

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": "us-east-1"
    }
}
```

  ポリシーが出典リージョンへのアクセスを許可していないため、リクエストは失敗します。リクエストを正常に実行するには、ソースリージョンとコピー先リージョンの両方を指定します。

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": [
            "us-east-1",
            "us-east-2"
        ]
    }
}
```

- リクエスタのポリシーでは、ソース DB インスタンスへのアクセスが許可されていません。

```
...
```

```
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "arn:aws:rds:us-east-1:123456789012:db:myreadreplica"
...
```

リクエストを正常に実行するには、ソースインスタンスとレプリカの両方を指定します。

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": [
    "arn:aws:rds:us-east-1:123456789012:db:myreadreplica",
    "arn:aws:rds:us-east-2:123456789012:db:mydbinstance"
]
...
```

- リクエスターのポリシーが aws:ViaAWSService を拒否します。

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
    "Bool": {"aws:ViaAWSService": "false"}
}
```

出典リージョンとの通信は、リクエスタに代わって RDS によって行われます。リクエストを正常に実行するには、AWS のサービスからの呼び出しを拒否しないでください。

- リクエスタのポリシーには、aws:SourceVpc または aws:SourceVpce の条件があります。

RDS がリモートリージョンへの呼び出しを行うとき、指定された VPC または VPC エンドポイントからの呼び出しではないため、これらのリクエストは失敗する可能性があります。

リクエストが失敗する原因となる前述の条件のいずれかを使用する必要がある場合は、ポリシーに aws:CalledVia とともに 2 番目のステートメントを含めて、リクエストを成功させることができます。例えば、次のように aws:CalledVia と aws:SourceVpce を使用できます。

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
```

```
"Resource": "*",
"Condition": {
    "Condition" : {
        "ForAnyValue:StringEquals" : {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
     }
},
{
    "Effect": "Allow",
    "Action": [
        "rds:CreateDBInstanceReadReplica"
    ],
    "Resource": "*",
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": [
                "rds.amazonaws.com"
            ]
        }
    }
}
```

詳細については、IAM ユーザーガイド の 「IAM のポリシーとアクセス許可」を参照してください。

リードレプリカの承認

クロスリージョン DB のリードレプリカ作成リクエストが success を返した後、RDS はバックグラウンドでレプリカの作成を開始します。RDS がソース DB インスタンスにアクセスするための承認が作成されます。この承認は、ソース DB インスタンスをリードレプリカにリンクし、RDS が指定されたリードレプリカにのみコピーできるようにします。

承認は、サービスにリンクされた IAM ロールの rds:CrossRegionCommunication アクセス許可を使用して RDS によって検証されます。レプリカが承認されると、RDS はソースリージョンと通信し、レプリカの作成を完了します。

RDS は、CreateDBInstanceReadReplica リクエストによって以前に承認されていない DB インスタンスにアクセスできません。リードレプリカの作成が完了すると、承認は取り消されます。

RDS は、サービスリンクされたロールを使用して、ソースリージョンでの承認を確認します。レプリケーション作成プロセス中にサービスリンクされたロールを削除すると、作成は失敗します。

詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの使用」を参照してください。

AWS Security Token Service 認証情報の使用

グローバル AWS Security Token Service (AWS STS) エンドポイントからのセッショントークンは、デフォルトで有効になっている AWS リージョン (商用リージョン) でのみ有効です。assumeRole の AWS STS API 操作からの認証情報を使用する場合、出典リージョンがオプトインリージョンである場合は、そのリージョンのエンドポイントを使用します。それ以外の場合、このリクエストは失敗します。これは、認証情報が両方のリージョンで有効である必要があるために発生します。これは、そのリージョンの AWS STS エンドポイントが使用されている場合にのみオプトインリージョンに当てはまります。

グローバルエンドポイントを使用するには、オペレーションで両方のリージョンで有効になっていることを確認します。Valid in all AWS ##### アカウント設定でグローバルエンドポイントを AWS STS に設定します。

署名付き URL パラメータの認証情報にも同じルールが適用されます。

詳細については、IAM ユーザーガイドの 「AWS リージョン での AWS STS の管理」を参照してください。

クロスリージョンレプリケーションのコスト

クロスリージョンレプリケーションから転送されたデータには、Amazon RDS のデータ転送料金が発生します。以下のクロスリージョンレプリケーションアクションでは、ソース AWS リージョンから転送されるデータに対して料金が発生します。

- リードレプリカを作成すると、Amazon RDS によりソースインスタンスのスナップショットが作成され、リードレプリカ AWS リージョン にスナップショットが転送されます。
- ソースデータベースのデータに変更が加えられるたびに、Amazon RDS によりソース AWS リージョン からリードレプリカ AWS リージョン にデータが転送されます。

データ転送料金の詳細については、「Amazon RDS の料金」を参照してください。

MySQL および MariaDB インスタンスの場合、作成するクロスリージョンリードレプリカの数を減らすことで、データ転送コストを削減できます。例えば、1 つの AWS リージョン にソース DB インスタンスがあり、別の AWS リージョン に 3 つのリードレプリカが必要であるとします。この場合、ソース DB インスタンスからリードレプリカを 1 つのみ作成します。他の 2 つのレプリカは、ソース DB インスタンスからではなく、最初のリードレプリカから作成します。

例として、ある AWS リージョン に source-instance-1 がある場合は、次のようにできます。

- ソースとして source-instance-1 を指定して、新しい AWS リージョン に read-replica-1 を作成します。
- read-replica-2 から read-replica-1 を作成します。
- read-replica-3 から read-replica-1 を作成します。

この例では、source-instance-1 から read-replica-1 に転送されるデータ対してのみ課金されます。read-replica-1 から他の 2 つのレプリカに転送されたデータには課金されません。すべて同じ AWS リージョン にあるためです。3 つのレプリカをすべて別の AWS リージョンの source-instance-1 から直接作成した場合、3 つのレプリカすべてへのデータ転送に課金されます。

Amazon RDS リソースのタグ付け

Amazon RDS タグは、DB インスタンスまたは DB スナップショットなどの Amazon RDS リソースを定義し、関連付ける名前と値のペアです。その名前はキーと呼ばれます。オプションで、キーに値を与えることができます。

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、Amazon RDS リソースに対してタグを追加、一覧表示、削除できます。CLI または API を使用するときは、操作する RDS リソースの Amazon リソースネーム (ARN) を指定する必要があります。ARN の作成の詳細については、「Amazon RDS 用 ARN の構築」をご参照ください。

タグを使用して Amazon RDS リソースにメタデータを追加できます。タグを使用して、データベースインスタンス、スナップショット、Aurora クラスターなどに関する独自の表記を追加できます。これにより、Amazon RDS リソースを文書化できます。また、自動メンテナンスの手順でタグを使用することもできます。

特に、これらのタグは IAM ポリシーで使用できます。これらを使用して、Amazon RDS リソースへのアクセスを管理したり、これらのリソースに適用できるアクションを制御したりできます。また、これらのタグを使用して、類似のリソースの費用をグループ化することで、コストを追跡できます。

以下の Amazon RDS リソースにタグを付けることができます。

- DB インスタンス
- DB クラスター
- Aurora グローバルクラスター
- DB クラスターエンドポイント
- リードレプリカ
- DB スナップショット
- DB クラススナップショット
- リザーブド DB インスタンス
- イベントサブスクリプション
- DB オプショングループ
- DB パラメータグループ
- DB クラスターのパラメータグループ
- DB サブネットグループ

- RDS プロキシ
- RDS Proxy エンドポイント
- ブルー/グリーンデプロイ
- ゼロ ETL 統合
- 自動バックアップ
- クラスター自動バックアップ

> **Note**
>
> DB インスタンスにタグを付けると、Amazon RDS はそれらのタグを関連する Performance Insights リソースに自動的に適用します。現在、AWS マネジメントコンソール を使用して、RDS プロキシおよび RDS プロキシエンドポイントにタグ付けすることはできません。

トピック

- Amazon RDS リソースタグを使用する理由
- Amazon RDS リソースタグの仕組み
- Amazon RDS リソースのタグ付けのベストプラクティス
- タグを DB スナップショットにコピーする
- 自動バックアップリソースのタグ付け
- Amazon RDS でのタグの追加と削除
- チュートリアル: タグを使用して、停止する DB インスタンスを指定する

Amazon RDS リソースタグを使用する理由

タグを使用して、以下のことを行うことができます。

- RDS リソースをアプリケーション、プロジェクト、部門、環境などに分類します。例えば、タグキーを使用してカテゴリを定義し、タグ値をそのカテゴリのアイテムにすることができます。タグ `environment=prod` を作成できます。または、`project` のタグキーと `Salix` のタグ値を定義して、Amazon RDS リソースが Salix プロジェクトに割り当てられていることを示すことができます。
- リソース管理タスクを自動化します。例えば、`environment=test` にタグ付けされたインスタンスのウィンドウとは異なる `environment=prod` にタグ付けされたインスタンスのメンテナンス

ウィンドウを作成できます。インスタンスにタグ付けされた environment=prod の自動 DB スナップショットを設定することもできます。

- IAM ポリシー内の RDS リソースへのアクセスを制御します。これを行うには、グローバル条件キー aws:ResourceTag/*tag-key* を使用します。例えば、ポリシーでは、DBAdmin グループ内のユーザーのみを environment=prod でタグ付けされた DB インスタンスの変更を許可できます。IAM ポリシーでタグ付けされたリソースへのアクセスの管理については、「AWS Identity and Access Management ユーザーガイド」の「Amazon RDS での Identity and Access Management」および「AWS リソースへのアクセスの制御」を参照してください。
- タグに基づいてリソースをモニタリングします。例えば、environment=prod でタグ付けされた DB インスタンス用の Amazon CloudWatch ダッシュボードを作成できます。
- 同様にタグ付けされたリソースの費用をグループ化することで、コストを追跡します。例えば、project=Salix で Salix プロジェクトに関連付けられた RDS リソースにタグ付けした場合、コストレポートを生成して、このプロジェクトに経費を割り当てることができます。詳細については、「Amazon RDS でのタグを使用した AWS 請求の仕組み」を参照してください。

Amazon RDS リソースタグの仕組み

AWS はタグに意味論的意味を適用しません。タグは単なる文字列として解釈されます。

トピック

- Amazon RDS のタグセット
- Amazon RDS のタグ構造
- タグ付けの対象となる Amazon RDS リソース
- Amazon RDS でのタグを使用した AWS 請求の仕組み

Amazon RDS のタグセット

すべての Amazon RDS リソースには、タグセットと呼ばれるコンテナがあります。コンテナには、リソースに割り当てられたすべてのタグが含まれます。リソースにはタグセットが 1 つだけあります。

タグセットには 0 ~ 50 個のタグが含まれます。既存のリソースタグと同じキーを持つタグを RDS リソースに追加した場合、既存の値は新しい値によって上書きされます。

Amazon RDS のタグ構造

RDS タグの構造は次のとおりです。

タグキー

キーは、タグの必須の名前です。文字列値は、1 ~ 128 文字の Unicode 文字であり、aws: または rds: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、_、.、:、/、=、+、-、および @ を含めることができます。Java 正規表現は "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$" です。タグキーでは、大文字と小文字が区別されます。したがって、キー project と Project は区別されます。

キーはタグセットに固有です。例えば、project=Trinity や project=Xanadu など、タグセットでキーが同じで値が異なるキーと値のペアは使用できません。

タグ値

値は、タグのオプションの文字列値です。文字列値は、1 ~ 256 文字の Unicode 文字である必要があります。文字列には、一連の Unicode 文字、数字、空白、_、.、:、/、=、+、-、および @ を含めることができます。Java 正規表現は "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$" です。タグ値は大文字と小文字が区別されます。したがって、値 prod と Prod は区別されます。

値はタグセット内で一意である必要はなく、null にできます。例えば、project=Trinity と cost-center=Trinity のタグセット内に 1 つのキーと値のペアを使用できます。

タグ付けの対象となる Amazon RDS リソース

次の Amazon RDS リソースにタグ付けができます。

- DB インスタンス
- DB クラスター
- DB クラスターエンドポイント
- リードレプリカ
- DB スナップショット
- DB クラススナップショット
- リザーブド DB インスタンス
- イベントサブスクリプション

- DB オプショングループ
- DB パラメータグループ
- DB クラスターのパラメータグループ
- DB サブネットグループ
- RDS プロキシ
- RDS Proxy エンドポイント

> **Note**
>
> 現在、AWS マネジメントコンソール を使用して、RDS プロキシおよび RDS プロキシエンドポイントにタグ付けすることはできません。

- ブルー/グリーンデプロイ
- ゼロ ETL 統合 (プレビュー)
- 自動バックアップ
- クラスター自動バックアップ

Amazon RDS でのタグを使用した AWS 請求の仕組み

タグを使用して、自分のコスト構造を反映するように AWS 請求書を整理します。そのためには、サインアップして、タグキー値が含まれた AWS アカウント の請求書を取得する必要があります。次に、結合したリソースのコストを見るには、同じタグキー値のリソースに従って請求書情報を整理します。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。詳細については、AWS Billingユーザーガイド の「コスト配分タグの使用」をご参照ください。

DB スナップショットでのコスト配分タグの仕組み

DB スナップショットにタグを追加することができます。ただし、このグループは請求書に反映されません。コスト配分タグを DB スナップショットに適用するには、以下の条件を満たす必要があります。

- タグは親 DB インスタンスにアタッチする必要があります。
- 親 DB インスタンスは、DB スナップショットと同じ AWS アカウント に存在する必要があります。

- 親 DB インスタンスは、DB スナップショットと同じ AWS リージョン に存在する必要があります。

DB スナップショットは、親 DB インスタンスと同じ リージョンに存在しない場合、孤立したと見なされます。孤立した DB スナップショットは、コスト配分タグをサポートしません。孤立したスナップショットのコストは、タグのない単一の項目に集約されます。クロスアカウント DB スナップショットは、次の条件が満たされても孤立したとは見なされません。

- 親 DB インスタンスと同じリージョンに存在します。
- 親 DB インスタンスはソースアカウントによって所有されます。

> **Note**
>
> 親 DB インスタンスが別の アカウントによって所有されている場合、コスト配分タグはデスティネーションアカウントのクロスアカウントスナップショットに適用されません。

Amazon RDS リソースのタグ付けのベストプラクティス

タグを使用するときには、以下のベストプラクティスに従うことをおすすめします。

- 組織内のすべてのチームが従うタグ使用の規則を文書化します。特に、名前が記述的で一貫性があることを確認してください。例えば、`env:production` を持つリソースにタグ付けするのではなく、形式 `environment:prod` で標準化します。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はタグに格納しないでください。

- タグ付けを自動化して一貫性を確保します。例えば、以下の技法を使用できます。
  - CloudFormation テンプレートにタグを含めます。テンプレートを使用してリソースを作成すると、リソースは自動的にタグ付けされます。
  - AWS Lambda 関数を使用してタグを定義して適用します。
  - RDS リソースにタグを追加する手順を含む SSM ドキュメントを作成します。
- タグは必要な場合にのみ使用します。1 つの RDS リソースに最大 50 個のタグを追加できますが、ベストプラクティスは、不要なタグの拡散や複雑さを回避することです。

- タグの関連性と正確性を定期的に確認します。必要に応じて、古いタグを削除または変更します。
- AWS マネジメントコンソール で AWS タグエディタを使用してタグを作成することを検討してください。タグエディタを使用して、RDS リソースを含む複数のサポートされている AWS リソースにタグを同時に追加できます。詳細については、AWS リソースグループユーザーガイドのタグエディタを参照してください。

タグを DB スナップショットにコピーする

DB インスタンスを作成または復元するとき、DB インスタンスから DB インスタンスのスナップショットにタグがコピーされるように指定できます。タグをコピーすると、DB スナップショットとソース DB インスタンスのメタデータが確実に一致するようになります。また、DB スナップショットとソース DB インスタンスのアクセスポリシーが確実に一致するようになります。

次のアクションでタグが DB スナップショットにコピーされるように指定できます。

- DB インスタンスの作成
- DB インスタンスの復元
- リードレプリカの作成
- DB スナップショットのコピー

前述のアクションでタグをコピーするには、AWS マネジメントコンソールで [スナップショットにタグをコピー] を選択するか、AWS CLI で `--copy-tags-to-snapshot` を指定します。DB スナップショットのコピー中に、同じアカウントで DB スナップショットをコピーする場合にのみ、関連付けられたタグをコピーできます。アカウント間で DB スナップショットをコピーするときにタグをコピーすることはできません。

ほとんどの場合、デフォルトでタグのコピーは行われません。DB スナップショットから DB インスタンスを復元した際に、RDS は新しいタグが指定されているかどうかをチェックします。新しいタグが指定されている場合、そのタグは復元された DB インスタンスに追加されます。新しいタグがなく、DB スナップショットがソース DB インスタンスと同じアカウントにある場合、RDS は、スナップショットの作成時にソース DB インスタンスから復元された DB インスタンスにタグを追加します。DB スナップショットが別の アカウントにコピーされて復元された場合、RDS はソース DB インスタンスからタグを追加しません。

ソース DB インスタンスのタグが復元された DB インスタンスに追加されることを防ぐには、DB インスタンスの復元時に新しいタグを指定することをお勧めします。

> **Note**
>
> 場合によっては、create-db-snapshot AWS CLI コマンドの --tags パラメータに値を含めることができます。または、CreateDBSnapshot API オペレーションに少なくとも 1 つのタグを指定することもできます。このような場合、RDS はソース DB インスタンスから新しい DB スナップショットにタグをコピーしません。この機能は、ソース DB インスタンスの --copy-tags-to-snapshot (CopyTagsToSnapshot) オプションが有効になっている場合でも、適用されます。
> このアプローチを使用すると、DB スナップショットから DB インスタンスのコピーを作成できます。この方法では、新しい DB インスタンスに適用されないタグを追加する必要がなくなります。DB スナップショットは、AWS CLIcreate-db-snapshotコマンド (またはCreateDBSnapshot RDS API オペレーション) を使用して作成します。DB スナップショットを作成した後、このトピックで説明しているように、タグを追加することができます。

自動バックアップリソースのタグ付け

自動バックアップリソースは、バックアップ保持期間値を 0 から 0 以上に設定したときに作成されます。--tag-specifications パラメータを使用して、作成時にインスタンスまたはクラスターの自動バックアップリソースにタグを付けることができます。

タグ仕様パラメータ

--tag-specifications リクエストパラメータをサポートする API (create-db-instance、restore-db-instance-from-db-snapshot、create-db-cluster など) は、作成時に自動バックアップ (リソースタイプ: auto-backup または cluster-auto-backup) にタグを付けることができます。

インスタンスの自動バックアップのタグ付け

自動バックアップが有効になっている DB インスタンスを作成または変更するときは、ResourceType=auto-backup を指定した --tag-specifications を使用します。

クラスター自動バックアップのタグ付け

自動バックアップが有効になっている DB クラスターを作成するときは、ResourceType=cluster-auto-backup を指定した --tag-specifications を使用します。

DB インスタンスの変更中の自動バックアップのタグ付け

バックアップ保持期間を 0 から 0 より大きい値に設定する場合のみ、`--tag-specifications` を使用して自動バックアップにタグを付けることができます。自動バックアップが既に存在する場合は、代わりに AddTagsToResource API を使用します。

> **Note**
>
> - 自動バックアップタグは、ソース DB インスタンスタグ、DB クラスタータグ、または DB スナップショットタグとは独立しています。

Amazon RDS でのタグの追加と削除

以下の操作を行うことができます。

- リソースの作成時にタグを作成します。例えば、AWS CLI コマンド `create-db-instance` を実行するときなどです。
- 既存のリソースにタグを追加するには、コマンド `add-tags-to-resource` を使用します。
- コマンド `list-tags-for-resource` を使用して、特定のリソースに関連付けられたタグを一覧表示します。
- コマンド `add-tags-to-resource` を使用してタグを更新します。
- コマンド `remove-tags-from-resource` を使用して、リソースからタグを削除します。

次の手順は、DB インスタンスに関連するリソースに対して一般的なタグ付け操作を実行する方法を示しています。タグはリソース承認用にキャッシュに格納されます。このため、Amazon RDS リソースにタグを追加または更新すると、変更が利用可能になるまでに数分かかることがあります。

コンソール

Amazon RDS リソースにタグを追加するプロセスはすべてのリソースで同様です。以下の手順では、Amazon RDS DB インスタンスにタグを付加する方法を示します。

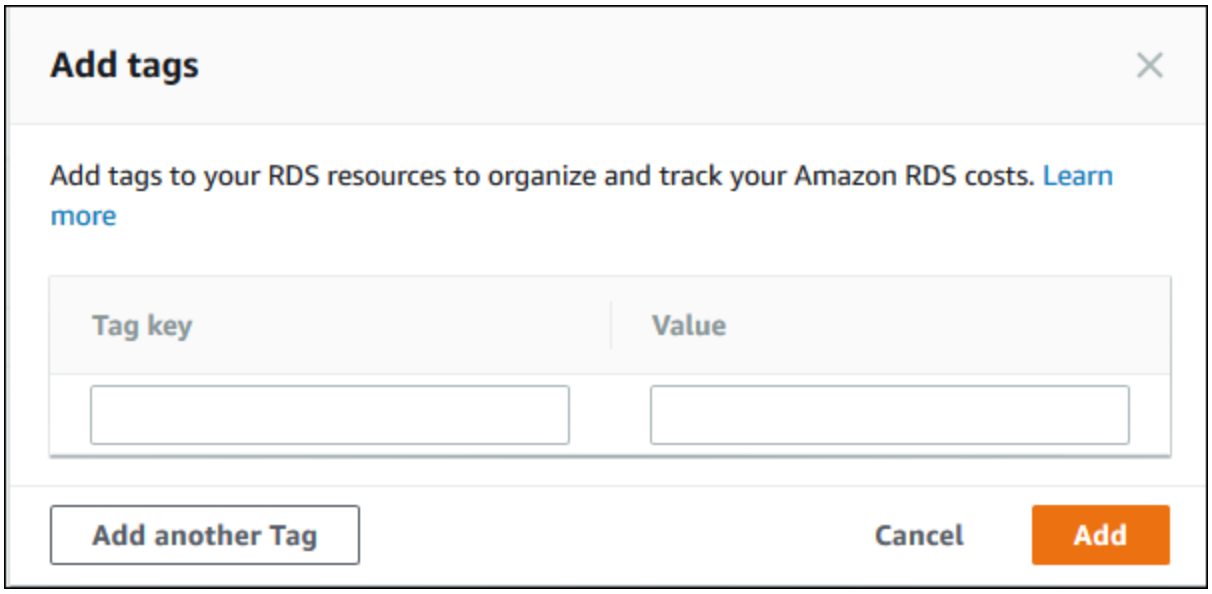
DB インスタンスにタグを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[データベース] を選択します。

> **Note**
>
> [Filter databases (データベースのフィルター)] ペインで DB インスタンスの一覧をフィルターするには、[Filter databases (データベースのフィルター)] のテキスト文字列を入力します。その文字列を含む DB インスタンスのみが表示されます。

3. タグ付けする DB インスタンスの名前を選択して、その詳細を表示します。
4. 詳細セクションで、下にスクロールし、[タグ] を選択します。
5. [Add] を選択します。[タグの追加] ウィンドウが表示されます。



6. [タグキー] と [値] の値を入力します。
7. 別のタグを追加するには、[別のタグを追加] を選択し、[タブキー] と [値] の値を入力します。

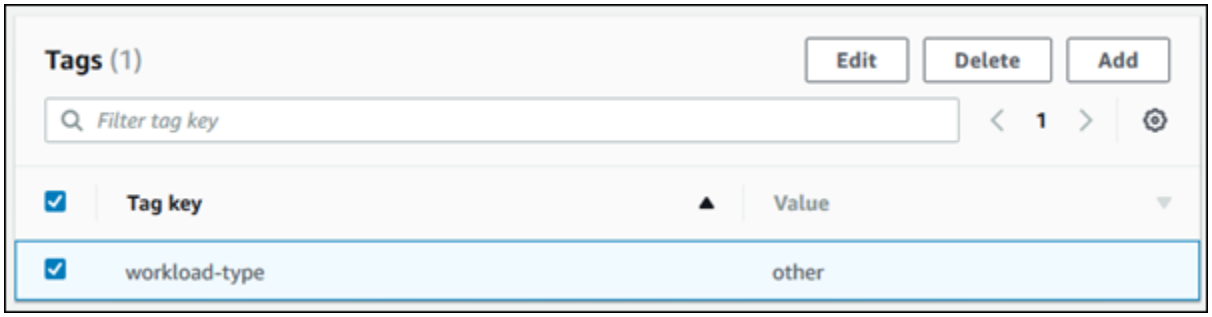
   このステップを必要な回数繰り返します。
8. [Add] を選択します。

DB インスタンスからタグを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

> **Note**
>
> [Filter databases (データベースのフィルター)] ペインで DB インスタンスの一覧をフィルターするには、[Filter databases (データベースのフィルター)] ボックスにテキスト文字列を入力します。その文字列を含む DB インスタンスのみが表示されます。

3. DB インスタンスの名前を選択して、その詳細を表示します。
4. 詳細セクションで、下にスクロールし、[タグ] を選択します。
5. 削除するタグを選択します。



6. [削除] を選択し、[Delete tags] (タグの削除) ウィンドウから [削除] を選択します。

AWS CLI

AWS CLI を使用して DB インスタンスのタグを追加、一覧表示、または削除できます。

- Amazon RDS リソースに 1 つ以上のタグを追加するには、AWS CLI コマンド `add-tags-to-resource` を使用します。
- Amazon RDS リソースのタグを一覧表示するには、AWS CLI コマンド `list-tags-for-resource` を使用します。
- Amazon RDS リソースから 1 つ以上のタグを削除するには、AWS CLI コマンド `remove-tags-from-resource` を使用します。

必要な ARN を作成する方法の詳細については、「Amazon RDS 用 ARN の構築」を参照してください。

RDS API

Amazon RDS API を使用して DB インスタンスのタグを追加、一覧表示、または削除できます。

- Amazon RDS リソースにタグを追加するには、AddTagsToResource オペレーションを使用します。
- Amazon RDS リソースに割り当てられているタグを一覧表示するには、ListTagsForResource を使用します。
- Amazon RDS リソースからタグを削除するには、RemoveTagsFromResource オペレーションを使用します。

必要な ARN を作成する方法の詳細については、「Amazon RDS 用 ARN の構築」を参照してください。

Amazon RDS API を使用して XML を操作する場合、タグでは以下のスキーマを使用します。

```
<Tagging>
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

以下の表に示しているのは、使用可能な XML タグとその特性のリストです。Key と Value の値は、大文字と小文字が区別されます。例えば、project=Trinity と PROJECT=Trinity は別個のタグです。

タグ付け要素	説明
タグセット	タグセットは、Amazon RDS リソースに割り当てられるすべてのタグのコンテナです。リソースごとに割り当て可能なのは 1 つのタグセットのみです。Amazon RDS API によってのみタグセットを操作できます。
Tag	タグはユーザー定義のキーと値のペアです。1 ~ 50 個のタグをタグセットに含めることができます。

タグ付け要素	説明
キー	キーはタグの必須の名前です。制限については、「Amazon RDS のタグ構造」を参照してください。 文字列値は、1 ~ 128 文字の Unicode 文字です。aws: または rds: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」 (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)") のみ使用できます。 キーはタグセットに対して一意である必要があります。例えば、タグセットでキーが同じで値が異なるキーと値のペアは使用できません。例えば、project/Trinity や project/Xanadu は使用できません。
値	値はタグの省略可能な値です。制限については、「Amazon RDS のタグ構造」を参照してください。 文字列値は、1 ~ 256 文字の Unicode 文字です。aws: または rds: をプレフィックスとして使用することはできません。文字列には、一連の Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「+」、「-」 (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)") のみ使用できます。 値はタグセット内で一意である必要はなく、null を指定できます。例えば、project/Trinity と cost-center/Trinity のタグセット内に 1 つのキーと値のペアを使用できます。

チュートリアル: タグを使用して、停止する DB インスタンスを指定する

このチュートリアルでは、開発環境またはテスト環境に複数の DB インスタンスがあることを前提としています。これらの DB インスタンスを数日間保持する必要があります。一部の DB インスタンスは夜間にテストを実行し、その他は夜間に停止し、翌日に再び開始します。

次のチュートリアルでは、夜間に停止するのに適した DB インスタンスにタグを割り当てる方法を示しています。このチュートリアルは、スクリプトがそのタグを持つ DB インスタンスを検出して、タグの付いた DB インスタンスを停止する方法を示しています。この例では、キーと値のペアでの値の部分は重要ではありません。stoppable タグが存在するということは、DB インスタンスにこのユーザー定義プロパティがあることを示します。

次のチュートリアルでは、タグ付け用のコマンドと API が ARN と機能し、RDS は同じ短縮名を持つ可能性のある AWS リージョン、AWS アカウント、およびさまざまなタイプのリソース間でシームレスに動作できます。DB インスタンスで動作する CLI コマンドで、DB インスタンス ID の代わりに ARN を指定できます。

停止する DB インスタンスを指定する

1. 停止可能として指定する DB インスタンスの ARN を決めます。

   次の例では、*dev-test-db-instance* を独自の DB インスタンスの名前に置き換えます。ARN パラメータを使用する後続のコマンドでは、独自の DB インスタンスの ARN を置き換えます。ARN には、お客様自身の AWS アカウント ID と、DB インスタンスが配置されている AWS リージョンの名前が含まれます。

   ```
   $ aws rds describe-db-instances --db-instance-identifier dev-test-db-instance \
     --query "*[].{DBInstance:DBInstanceArn}" --output text
   arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
   ```

2. この DB インスタンスに `stoppable` タグを追加します。

   このタグの名前を選択します。この例では、タグが存在するか存在しないかの属性として扱うため、`Value=` パラメータの `--tags` 部分を省略します。このアプローチにより、すべての関連情報を名前にエンコードする命名規則を考案する必要がなくなります。このような規則では、DB インスタンス名または他のリソースの名前に情報をエンコードすることができます。

   ```
   $ aws rds add-tags-to-resource \
     --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance \
     --tags Key=stoppable
   ```

3. タグが DB インスタンスに存在することを確認します。

   これらのコマンドは、JSON 形式およびタブ区切りのプレーンテキストで DB インスタンスのタグ情報を取得します。

   ```
   $ aws rds list-tags-for-resource \
     --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
   {
       "TagList": [
           {
               "Key": "stoppable",
               "Value": ""
   ```

```
        }
    ]
}
aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance --
output text
TAGLIST stoppable
```

4. stoppable として指定されているすべての DB インスタンスを停止します。

   次の例では、すべての DB インスタンスを一覧表示するテキストファイルを作成します。シェルコマンドはリストをループし、各 DB インスタンスに関連属性がタグ付けされているかどうかを確認し、各 DB インスタンスに対してコマンド aws rds stop-db-instance を実行します。

```
$ aws rds describe-db-instances --query "*[].[DBInstanceArn]" --output text >/tmp/
db_instance_arns.lst
$ for arn in $(cat /tmp/db_instance_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 stoppable)"
  if [[ ! -z "$match" ]]
  then
      echo "DB instance $arn is tagged as stoppable. Stopping it now."
# Note that you need to get the DB instance identifier from the ARN.
      dbid=$(echo $arn | sed -e 's/.*://')
      aws rds stop-db-instance --db-instance-identifier $dbid
  fi
done

DB instance arn:arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance is
 tagged as stoppable. Stopping it now.
{
    "DBInstance": {
        "DBInstanceIdentifier": "dev-test-db-instance",
        "DBInstanceClass": "db.t3.medium",
        ...
```

このようなスクリプトを 1 日の終わりに実行して、必要ではない DB インスタンスが停止されていることを確認できます。cron などのユーティリティを使用してジョブのスケジュールを組み、毎晩そのような確認を実行することもできます。例えば、DB インスタンスが誤って実行されたままに

なった場合に利用できます。ここでは、確認する DB インスタンスのリストを準備するコマンドを微調整できます。

次のコマンドは、DB インスタンスのリストを生成しますが、available 状態の場合にのみ生成します。スクリプトでは、stopped または stopping などのステータス値が異なるため、すでに停止している DB インスタンスを無視できます。

```
$ aws rds describe-db-instances \
  --query '*[].{DBInstanceArn:DBInstanceArn,DBInstanceStatus:DBInstanceStatus}|[?
DBInstanceStatus == `available`]|[].{DBInstanceArn:DBInstanceArn}' \
  --output text
arn:aws:rds:us-east-1:123456789102:db:db-instance-2447
arn:aws:rds:us-east-1:123456789102:db:db-instance-3395
arn:aws:rds:us-east-1:123456789102:db:dev-test-db-instance
arn:aws:rds:us-east-1:123456789102:db:pg2-db-instance
```

> **Tip**
>
> タグを割り当てて、それらのタグを持つ DB インスタンスを検索して、他の方法でコストを削減することができます。例えば、開発とテストに使用される DB インスタンスのシナリオを考えてみましょう。この場合、一日の終わりにいくつかの DB インスタンスを削除するように指定できます。または、使用率が低いと予想される時間帯に DB インスタンスを小さな DB インスタンスクラスに変更するように指定することもできます。

Amazon RDS の Amazon リソースネーム (ARN)

Amazon Web Services で作成されたリソースは、Amazon リソースネーム (ARN) によってそれぞれ一意に識別されます。特定の Amazon RDS オペレーションでは、ARN を指定して、Amazon RDS リソースを一意に識別する必要があります。例えば、RDS DB インスタンスのリードレプリカを作成する場合、ソース DB インスタンスの ARN を指定する必要があります。

ARN のコンストラクトと既存の ARN の取得については、以下のトピックを参照してください。

トピック

- Amazon RDS 用 ARN の構築
- Amazon RDS の既存の ARN の取得

Amazon RDS 用 ARN の構築

Amazon Web Services で作成されたリソースは、Amazon リソースネーム (ARN) によってそれぞれ一意に識別されます。次の構文を使用して Amazon RDS リソースの ARN を構築できます。

```
arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	
米国東部 (バージニア北部)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
		rds-fips.us-east-1.api.aws	HTTPS	
		rds-fips.us-east-1.amazonaws.com	HTTPS	
		rds.us-east-1.api.aws	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
米国西部 (北カリフォルニア)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
		rds.us-west-1.api.aws	HTTPS	
		rds-fips.us-west-1.amazonaws.com	HTTPS	
		rds-fips.us-west-1.api.aws	HTTPS	
米国西部 (オレゴン)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
		rds-fips.us-west-2.amazonaws.com	HTTPS	
		rds.us-west-2.api.aws	HTTPS	
		rds-fips.us-west-2.api.aws	HTTPS	
アフリカ (ケープタウン)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
		rds.af-south-1.api.aws	HTTPS	
アジアパシフィック (香港)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
		rds.ap-east-1.api.aws	HTTPS	
アジアパシフィック (ハイデラバード)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
		rds.ap-south-2.api.aws	HTTPS	
アジアパシフィック (ジャカルタ)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
		rds.ap-southeast-3.api.aws	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (マレーシア)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
アジアパシフィック (メルボルン)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS	
アジアパシフィック (ニュージーランド)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS	
アジアパシフィック (ソウル)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (シドニー)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
アジアパシフィック (台北)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
アジアパシフィック (タイ)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
アジアパシフィック (東京)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
カナダ (中部)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
カナダ西部 (カルガリー)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS
欧州 (フランクフルト)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS
欧州 (アイルランド)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
		rds.eu-west-1.api.aws	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (ロンドン)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS	
ヨーロッパ (ミラノ)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS	
欧州 (パリ)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS	
欧州 (スペイン)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS	
欧州 (ストックホルム)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS	
欧州 (チューリッヒ)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS	
イスラエル (テルアビブ)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS	
メキシコ (中部)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
中東 (バーレーン)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
中東 (アラブ首長国連邦)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS	
南米 (サンパウロ)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS	
AWS GovCloud (米国東部)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS	
AWS GovCloud (米国西部)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS	

次の表に、特定の Amazon RDS リソースの ARN の構築時に使用する形式を示します。

リソースタイプ	ARN 形式
DB インスタンス	arn:aws:rds:*<region>*:*<account>* :db:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
DB クラスター	arn:aws:rds:*<region>*:*<account>* :cluster:*<name>* 例:

リソースタイプ	ARN 形式
	`arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
イベントサブスクリプション	arn:aws:rds:*<region>*:*<account>* :es:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB オプショングループ	arn:aws:rds:*<region>*:*<account>* :og:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :og:my-og`
DB パラメータグループ	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
DB クラスターのパラメータグループ	arn:aws:rds:*<region>*:*<account>* :cluster-pg:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
リザーブド DB インスタンス	arn:aws:rds:*<region>*:*<account>* :ri:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :ri:my-reserved-postgresql`

リソースタイプ	ARN 形式
DB セキュリティグループ	arn:aws:rds:*<region>*:*<account>* :secgrp:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :secgrp:my-public`
Automated DB スナップショット	arn:aws:rds:*<region>*:*<account>* :snapshot:rds:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :snapshot:rds: my-mysql-db-2019-07-22-07-23`
自動 DB クラススナップショット	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:rds:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds: my-aurora-cluster-2019-07-22-16-16`
手動 DB スナップショット	arn:aws:rds:*<region>*:*<account>* :snapshot:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`
手動の DB クラススナップショット	arn:aws:rds:*<region>*:*<account>* :cluster-snapshot:*<name>* 例: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`

リソースタイプ	ARN 形式
DB サブネットグループ	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* 例えば、次のようになります。 `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Amazon RDS の既存の ARN の取得

AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または RDS API を使用して、RDS リソースの ARN を取得できます。

コンソール

AWS マネジメントコンソール から ARN を取得するには、ARN を取得したいリソースに移動し、リソースの詳細を表示します。

例えば、DB インスタンスの詳細の [設定] タブから、DB インスタンスの ARN を取得できます。
DB インスタンス ARN

AWS CLI

特定の RDS リソースの AWS CLI から ARN を取得するには、そのリソースに対して `describe` コマンドを使用します。次の表に、各 AWS CLI コマンド、および ARN を取得するコマンドで使用された ARN のプロパティを示します。

AWS CLI コマンド	ARN プロパティ
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn

AWS CLI コマンド	ARN プロパティ
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	ReservedDBInstanceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-option-groups	OptionGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

例えば、次の AWS CLI コマンドで DB インスタンスの ARN を取得します。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Windows の場合:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

このコマンドの出力は次のようになります。

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

RDS API

特定の RDS リソースの ARN を取得するには、次の RDS API のオペレーションを呼び出し、次に示す ARN のプロパティを使用できます。

RDS API オペレーション	ARN プロパティ
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeOptionGroups	OptionGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Amazon RDS DB インスタンスのストレージを使用する

データを Amazon RDS に保存する方法を指定するには、DB インスタンスを作成または変更する際にストレージタイプを選択し、ストレージサイズを指定します。後で、量を増加するか、あるいは DB インスタンスを編集してストレージタイプを変更できます。ワークロードに使用されるストレージタイプについての詳細は、「Amazon RDS ストレージタイプ」を参照してください。

インスタンスが RDS for Oracle または RDS for SQL Server を実行している場合、各 DB インスタンスに最大 3 つのボリュームを追加できます。ボリュームタイプとして gp3 または io2 を選択できます。これにより、データアクセスパターンに基づいてコストとパフォーマンスを最適化できます。追加ボリュームを使用する DB インスタンスの最大ストレージ容量は 256 TiB です。

トピック

- DB インスタンスのストレージボリュームの詳細の表示
- DB インスタンスストレージの容量を増加する
- 追加ストレージボリュームの削除
- Amazon RDS ストレージの自動スケーリングによる容量の自動管理
- DB インスタンスのストレージファイルシステムのアップグレード
- プロビジョンド IOPS SSD ストレージの設定を変更する
- I/O を大量に消費するストレージの変更
- 汎用 SSD (gp3) ストレージの設定変更
- 専用ログボリューム (DLV) を使用する

DB インスタンスのストレージボリュームの詳細の表示

ストレージボリュームの設定は、AWS マネジメントコンソールまたは AWS CLI から表示できます。これには、プライマリストレージボリュームと DB インスタンスにアタッチされた追加のストレージボリュームの両方に関する詳細が含まれます。

コンソール

コンソールからストレージボリューム設定を表示するには:

1. AWS マネジメントコンソール (https://console.aws.amazon.com/rds/) にサインインします。
2. ナビゲーションペインで、[Databases] (データベース) を選択します。

3. DB インスタンスを選択します。
4. [設定] タブを選択すると、ストレージボリュームの詳細が表示されます。ストレージ情報は、以下のセクションにあります。

   - [プライマリストレージ]
   - [追加のストレージボリューム]

   [追加のストレージボリューム] では、[ストレージボリュームを追加する] を選択して別のボリュームを追加できます。

CLI

AWS CLI からストレージボリューム設定を表示するには、describe-db-instances コマンドを使用します。

```
aws rds describe-db-instances --db-instance-identifier my-database
```

出力で、追加されたボリュームの詳細を表示する AdditionalStorageVolumes 配列を見つけます。

```
    "AdditionalStorageVolumes": [
        {
            "VolumeName": "rdsdbdata2",
            "StorageVolumeStatus": "Not-in-use",
            "AllocatedStorage": 5000,
            "IOPS": 25000,
            "StorageThroughput": 500,
            "StorageType": "gp3"
        }
    ]
```

StorageVolumeStatus フィールドは、ボリュームがデータベースで現在使用されているかどうかを示します。Not-in-use ステータスは、ボリュームがアタッチされているが、データベースエンジンまたは RDS 機能で使用されていないことを意味します。

DB インスタンスストレージの容量を増加する

DB インスタンスのストレージ容量を増やすには、次のいずれかの方法を使用します。

ストレージボリュームをスケールアップする

すべての DB インスタンスには、最大容量が 64 TiB のプライマリストレージボリュームがあります。ストレージボリュームの割り当て領域を少なくとも 10% 増やすことができます。スペースの割り当てを解除することはできません。

最大 3 つのストレージボリューム (io2 または gp3) を追加する

RDS for Oracle または RDS for SQL Server インスタンスの合計ストレージ容量を増やすには、各 DB インスタンスに最大 3 つのボリュームを追加します。追加ボリュームごとに最大 64 TiB のストレージが割り当てられ、インスタンスごとに最大 256 TiB のストレージが割り当てられます。

> **Note**
>
> RDS for Oracle DB インスタンスの場合、最小ストレージサイズが 200 GiB のストレージボリュームを追加できます。

上記の手法には、Amazon RDS マネジメントコンソール、Amazon RDS API、または AWS Command Line Interface (AWS CLI) を使用できます。ストレージの制限に関する詳細は、「Amazon RDS DB インスタンスストレージ」を参照してください。

DB インスタンスストレージのスケールアップ

プライマリボリュームに割り当てられたストレージを増やすことで、既存の DB インスタンスのストレージをスケールアップできます。ストレージ割り当てを増やす場合は、少なくとも 10 パーセント単位で増やす必要があります。10 パーセントに満たない単位で増やそうとすると、エラーになります。ボリュームにストレージを割り当てた後で、ボリューム上のストレージ容量を減らすことはできません。

> **Note**
>
> RDS for SQL Server DB インスタンスの場合、汎用 SSD およびプロビジョンド IOPS SSD ストレージタイプに対してのみストレージをスケーリングできます。

必要に応じて対応できるように DB インスタンスの空きストレージ領域をモニタリングするには、Amazon CloudWatch アラームを作成することをお勧めします。CloudWatch アラームの設定の詳細については、「CloudWatch でのアラームの使用」を参照してください。

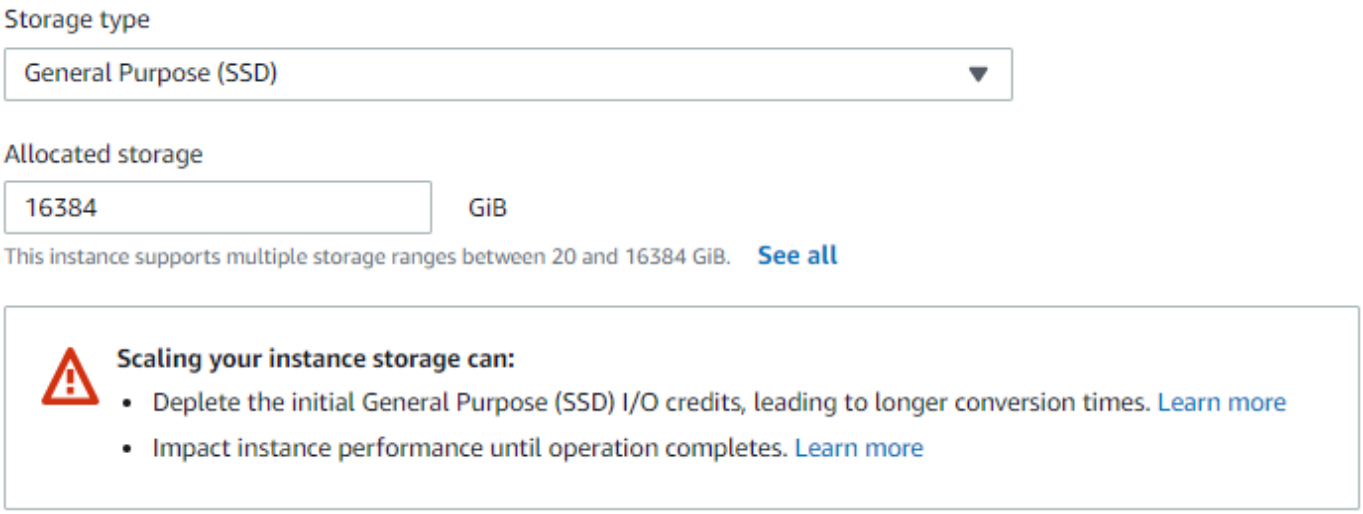
ストレージをスケーリングしても、通常、DB インスタンスの停止やパフォーマンスの低下は発生しません。DB インスタンスのストレージサイズを変更すると、DB インスタンスのステータスは [ストレージの最適化] になります。

ストレージの最適化には数時間かかることがあります。インスタンスでストレージの最適化が完了したら、追加のストレージ変更を行うことができます。任意の 24 時間以内に最大 4 回のストレージ変更を実行できます。ストレージ最適化の進捗状況は、AWS マネジメントコンソール で、または describe-db-instances AWS CLI コマンドを使用して確認できます。最適化中、レスポンスには、影響を受けるボリュームの `StorageOperationStatus="Optimizing"` と `StorageOperationPercentProgress` の値が含まれます。ボリュームごとのフィールドは `AdditionalStorageVolumes` 配列内の各エントリにも表示されるため、最適化中のボリュームを特定できます。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

コンソール

DB インスタンスのストレージを増加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] を選択します。
5. [ストレージ割り当て] に新しい値を入力します。現在値より少なくとも 10% 大きい値である必要があります。



6. [続行] をクリックしてください。

7. DB インスタンスのストレージの変更をすぐに適用するには、[変更のスケジュール] セクションの [すぐに適用] を選択します。

   または、[次に予定されるメンテナンスウィンドウ中に適用します] を選択して、次のメンテナンスウィンドウ中に変更を適用します。

8. 設定が適切な場合は、[Modify DB instance] (DB インスタンスの変更) を選択します。

AWS CLI

DB インスタンスのストレージを増量するには、AWS CLI の `modify-db-instance` コマンドを使用します。以下のパラメータを設定します。

- `--allocated-storage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `--apply-immediately` - すぐにストレージの変更を適用するには、`--apply-immediately` を使用します。

  または `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。変更が適用されると、すぐに停止が発生します。

次の例では、mydbinstance のストレージを 1,000 GiB にスケールアップし、変更をすぐに適用します。このコマンドは、ストレージボリュームを gp3 に移行し、プロビジョンド IOPS を 6000 に設定します。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --allocated-storage 1000 \
    --storage-type gp3 \
    --iops 6000 \
    --apply-immediately
```

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

RDS API

DB インスタンスのストレージを増やすには、Amazon RDS API オペレーション `ModifyDBInstance` を使用します。以下のパラメータを設定します。

- `AllocatedStorage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。

- `ApplyImmediately` - すぐに変更を適用するには、このオプションを `True` に設定します。このオプションを `False` (デフォルト) に設定して、次のメンテナンスウィンドウ中に変更を適用します。変更が適用されると、すぐに停止が発生します。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

ストレージボリュームの追加

RDS for Oracle および RDS for SQL Server DB インスタンスの場合、最大 3 つのストレージボリュームを追加して、インスタンスあたりの合計ストレージ容量を最大 256 TiB まで増やすことができます。追加のストレージボリュームを使用すると、さまざまなストレージタイプ (gp3 と io2) を使用して、データアクセスパターンに基づいてコストとパフォーマンスを最適化できます。

> **Note**
>
> RDS for Oracle DB インスタンスの場合、最小ストレージサイズが 200 GiB のストレージボリュームを追加できます。

AWS マネジメントコンソールまたは AWS CLI を使用して、追加のストレージボリュームを追加、変更、または削除できます。割り当てられたストレージ、IOPS、スループットの設定が異なるボリュームを設定できます。例えば、io2 ボリュームに高性能データを配置し、gp3 ボリュームに履歴データを配置できます。

追加のボリュームでは、次の表に示すボリューム名を使用する必要があります。

RDS for Oracle ボリューム名	RDS for SQL Server ボリューム名
`rdsdbdata2`	`H:`
`rdsdbdata3`	`I:`
`rdsdbdata4`	`J:`

コンソール

DB インスタンスにストレージボリュームを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. [ストレージ] セクションで、[追加のストレージボリュームを追加] を選択します。
6. 追加のストレージボリュームを設定します。
   - ボリューム名 – `rdsdbdata2`、`rdsdbdata3`、または `rdsdbdata4` を選択します。
   - ストレージタイプ – [汎用 SSD (gp3)] または [プロビジョンド IOPS SSD (io2)] を選択します。
   - 割り当てられたストレージ – ストレージサイズを GiB (最小 200 GiB) で入力します。
   - io2 ストレージの場合は、[プロビジョンド IOPS] を設定します。
   - gp3 ストレージの場合は、オプションで [ストレージスループット] を設定します。
7. [続行] をクリックしてください。
8. 設定が適切な場合は、[DB インスタンスを変更] を選択します。

> **Important**
>
> `modify-db-instance` オペレーションを使用してストレージボリュームを追加すると、RDS は `--no-apply-immediately` パラメータに関係なくストレージボリュームをすぐに追加します。リクエストに他の変更がある場合、RDS はスケジュールの変更に基づいて変更を適用します。「スケジュール変更設定の使用」を参照してください。

AWS CLI

DB インスタンスにストレージボリュームを追加するには、AWS CLI コマンド `modify-db-instance` を使用します。追加または変更する追加のストレージボリュームを指定する JSON 配列として `--additional-storage-volumes` を設定します。

次の例では、5000 GiB の gp3 ストレージを持つ rdsdbdata2 という名前のストレージボリュームを mydbinstance に追加します。

ストレージボリュームを追加すると、RDS は --no-apply-immediately パラメータに関係なくすぐに変更を適用します。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --additional-storage-volumes '[
        {
            "VolumeName": "rdsdbdata2",
            "StorageType": "gp3",
            "AllocatedStorage": 5000,
            "StorageThroughput": 725
        }
    ]'
```

RDS API

DB インスタンスにストレージボリュームを追加するには、Amazon RDS API オペレーション ModifyDBInstance を使用します。AdditionalStorageVolumes パラメータを追加のストレージボリューム仕様の配列として設定します。

ModifyDBInstance API オペレーションを使用してストレージボリュームを追加すると、RDS は ApplyImmediately オプションが True か False かに関係なく、ストレージボリュームをすぐに追加します。

追加ストレージボリュームの削除

不要になった追加のストレージボリュームは、RDS for Oracle および RDS for SQL Server DB インスタンスから削除できます。ボリュームを削除する前に、すべてのデータベースファイルをボリュームから移動し、それを参照しているデータベースオブジェクトがないことを確認してください。ボリュームステータスが Not-in-use になっていることを確認します。

> **Important**
>
> プライマリストレージボリュームを削除することはできません。追加のストレージボリュームのみを削除できます。

コンソール

DB インスタンスから追加のストレージボリュームを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. 削除するボリュームを含む DB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. [ストレージ] セクションで、削除する追加のストレージボリュームを見つけます。
6. 削除するボリュームで [ボリュームを削除] を選択します。

> **Note**
>
> ステータスが `Not-in-use` のボリュームのみを削除できます。ボリュームがまだ使用されている場合は、すべてのデータベースファイルをボリューム外に移動します。

7. [続行] をクリックしてください。
8. 設定が適切な場合は、[DB インスタンスを変更] を選択します。

AWS CLI

DB インスタンスから追加のストレージボリュームを削除するには、AWS CLI コマンド `modify-db-instance` を使用します。次のパラメータを設定します。

- `--additional-storage-volumes` – 残りの追加のストレージボリュームを指定する JSON 配列。この配列から削除するボリュームを省略します。

次の例では、残りのボリュームのみを指定して、`rdsdbdata3` という名前の追加ストレージボリュームを `mydbinstance` から削除し、変更をすぐに適用します。

```
aws rds modify-db-instance \
 --db-instance-identifier mydbinstance \
 --additional-storage-volumes '[{ \
  "VolumeName": "rdsdbdata3", \
  "SetForDelete": true
 }]'
```

RDS API

DB インスタンスから追加のストレージボリュームを削除するには、Amazon RDS API オペレーション `ModifyDBInstance` を使用します。以下のパラメータを設定します。

- `AdditionalStorageVolumes` – 保持するボリュームの追加ストレージボリューム仕様の配列。この配列から削除するボリュームを省略します。
- `ApplyImmediately` - すぐに変更を適用するには、このオプションを `True` に設定します。このオプションを `False` (デフォルト) に設定して、次のメンテナンスウィンドウ中に変更を適用します。

Amazon RDS ストレージの自動スケーリングによる容量の自動管理

ワークロードが予測不能な場合は、Amazon RDS DB インスタンスのストレージの自動スケーリングを有効にすることができます。そのためには、Amazon RDS コンソール、Amazon RDS API、または AWS CLI を使用できます。

> **Note**
>
> ストレージの自動スケーリングは、追加のストレージボリュームではサポートされていません。

例えば、ユーザーの導入が急速に拡大している新しいモバイルゲームアプリケーションでこの機能を使用できます。この場合、急激に増加するワークロードは、利用可能なデータベースストレージを超過する可能性があります。データベースストレージを手動でスケールアップしなくても済むように、Amazon RDS ストレージの自動スケーリングを使用できます。

ストレージの自動スケーリングが有効になっている場合、Amazon RDS でデータベースでの空き領域が不足していることが検出されると、自動的にストレージがスケールアップされます。次の要因が当てはまる場合、Amazon RDS によって自動スケーリングが有効な DB インスタンスのストレージ変更が開始されます。

- 使用可能な空き領域は、割り当てられたストレージの 10% 未満です。
- 低ストレージ状態は 5 分以上続きます。
- 前回のストレージ変更に対するインスタンスでのストレージ最適化が完了しており、過去 24 時間に発生したストレージの変更は 4 回未満です。

追加のストレージは、次のうちいずれか大きい方の増分です。

- 10 GiB
- 現在割り当てられているストレージの 10%
- 過去 1 時間の `FreeStorageSpace` メトリクスから、今後 7 時間以内にストレージが現在割り当てられているストレージサイズを超えると予測されます。メトリクスの詳細については、Amazon CloudWatch によるモニタリングを参照してください。

ストレージの最大しきい値は、DB インスタンスの自動スケーリングに設定する制限です。以下の制約があります。

- 最大ストレージしきい値は、現在割り当てられているストレージより少なくとも 10% 多く設定する必要があります。ストレージサイズが最大ストレージしきい値に近づいているというイベントの通知を受け取らないように、少なくとも 26% 以上に設定することをお勧めします。

  例えば、1000 GiB の割り当て済みストレージを持つ DB インスタンスがある場合、最大ストレージしきい値を少なくとも 1100 GiB に設定します。そうしないと、*`engine_name`* の最大ストレージサイズが無効ですようなエラーが発生します。ただし、イベントの通知を回避するために、最大ストレージしきい値を 1260 GiB 以上に設定することを推奨します。
- プロビジョンド IOPS (io1 または io2 Block Express) ストレージを使用する DB インスタンスの場合、IOPS と最大ストレージしきい値の比率 (GiB 単位) が特定の範囲内である必要があります。詳細については、「プロビジョンド IOPS SSD ストレージ」を参照してください。
- 自動スケーリング対応のインスタンスの場合、ストレージの最大しきい値を、データベースエンジンおよび DB インスタンスクラスに割り当てられた最大ストレージより大きい値に設定することはできません。

  例えば、db.m5.xlarge にある SQL Server Standard Edition には、20 GiB (最小) のインスタンスに対するデフォルトのストレージ割り当てと 16,384 GiB の最大ストレージ割り当てがあります。自動スケーリングに対するデフォルトの最大ストレージしきい値は 1,000 GiB です。このデフォルトを使用する場合、インスタンスは 1,000 GiB を超えて自動スケーリングしません。これは、インスタンスの最大ストレージ割り当てが 16,384 GiB であっても、該当します。

> **Note**
>
> 使用パターンと顧客のニーズに基づいて、最大ストレージしきい値を慎重に選択することをお勧めします。使用パターンに異常があり、Auto Scaling で予測しきい値が非常に高くなっ

た場合、最大ストレージしきい値により、ストレージが予期しない高い値にスケーリングされることを防止できます。DB インスタンスが自動スケーリングされた後、割り当てられたストレージを減らすことはできません。

トピック

- ストレージの自動スケーリングの制限
- 新しい DB インスタンスのストレージの自動スケーリングを有効化する
- DB インスタンスのストレージの自動スケーリング設定を変更する
- DB インスタンスを手動でスケールダウンまたはスケールインする
- DB インスタンスのストレージの自動スケーリングをオフにする

ストレージの自動スケーリングの制限

ストレージの自動スケーリングには、次の制限が適用されます。

- ストレージの増分により最大ストレージしきい値を超える場合、自動スケーリングは実行されません。
- オートスケーリングの場合、RDS は後続のオートスケーリングオペレーションのストレージサイズを予測します。後続のオペレーションが最大ストレージしきい値を超えると予測される場合、RDS では最大ストレージしきい値にオートスケーリングします。
- 自動スケーリングでは、大量のデータロードに伴うストレージ不足状態を完全に防ぐことはできません。これは、インスタンスでストレージの最適化が完了するまで追加のストレージ変更を行うことができず、任意の 24 時間以内に最大 4 回の変更に制限されるためです。

  大量のデータロードを実行し、Auto Scaling で十分な領域が得られない場合、データベースは数時間ストレージ不足状態になることがあります。これはデータベースに悪影響を与える場合があります。
- Amazon RDS が自動スケーリングオペレーションを開始するのと同時にストレージのスケーリングオペレーションを開始した場合は、ストレージの変更が優先されます。自動スケーリングオペレーションはキャンセルされます。
- 自動スケーリングでは、割り当てられたストレージを減らすことはできません。ストレージを割り当てた後に DB インスタンスのストレージ容量を減らすことはできません。
- 自動スケーリングは磁気ストレージでは使用できません。

- 自動スケーリングは、順序付け可能なストレージが 6 TiB 未満の旧世代のインスタンスクラス (db.m3.large、db.m3.xlarge、db.m3.2xlarge) では使用できません。
- 自動スケーリング操作は、AWS CloudTrail でログに記録されません。CloudTrail の詳細については、「AWS CloudTrail での Amazon RDS API コールのモニタリング」を参照してください。
- 自動スケーリングは、追加のストレージボリュームではサポートされていません。ストレージボリュームの追加方法については、「ストレージボリュームの追加」を参照してください。

自動スケーリングは Amazon RDS DB インスタンスのストレージを動的に増やすのに役立ちますが、それでも DB インスタンスの初期ストレージを通常のワークロードに適したサイズに設定する必要があります。

新しい DB インスタンスのストレージの自動スケーリングを有効化する

Amazon RDS DB インスタンスを作成する場合は、ストレージの自動スケーリングを有効にするかどうかを選択することができます。また、Amazon RDS が DB インスタンスに対して割り当てることができるストレージの上限を設定することもできます。

ストレージの自動スケーリングを有効にしても、データベースを再起動する必要はなく、ダウンタイムも発生しません。この機能は、データベースの操作を中断することなく、すぐに有効になります。

> **Note**
>
> ストレージの自動スケーリングが有効になっている Amazon RDS DB インスタンスのクローンを作成する場合、その設定は、クローンを作成したインスタンスに自動的に継承されません。新しい DB インスタンスには、元のインスタンスと同じ量のストレージが割り当てられます。クローンを作成したインスタンスのストレージ要件を再度増やしている場合は、新しいインスタンスに対してストレージの自動スケーリングを再度有効にすることができます。

コンソール

新しい DB インスタンスのストレージの自動スケーリングを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。

4. [データベースの作成] を選択します。[エンジンの選択] ページで、データベースエンジンを選択し、Amazon RDS のスタート方法 に示されている DB インスタンスの情報を指定します。
5. [Storage autoscaling (ストレージの自動スケーリング)] セクションで、DB インスタンスの [Maximum storage threshold (ストレージの最大しきい値)] を設定します。
6. Amazon RDS のスタート方法 に示されているように、DB インスタンス情報の残りを指定します。

AWS CLI

新しい DB インスタンスのストレージの自動スケーリングを有効にするには、AWS CLI コマンド `create-db-instance` を使用します。次のパラメータを設定します。

- `--max-allocated-storage` – ストレージの自動スケーリングをオンにして、ストレージサイズの上限 (ギビバイト単位) を設定します。

DB インスタンスに対して Amazon RDS ストレージの自動スケーリングが使用可能であることを確認するには、AWS CLI `describe-valid-db-instance-modifications` コマンドを使用します。インスタンスを作成する前に、インスタンスクラスに基づいて確認するには、`describe-orderable-db-instance-options` コマンドを使用します。戻り値の以下のフィールドを確認します。

- `SupportsStorageAutoscaling` – DB インスタンスまたはインスタンスクラスでストレージの自動スケーリングがサポートされているかどうかを示します。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

RDS API

新しい DB インスタンスのストレージの自動スケーリングを有効にするには、Amazon RDS API オペレーション `CreateDBInstance` を使用します。次のパラメータを設定します。

- `MaxAllocatedStorage` – Amazon RDS のストレージの自動スケーリングをオンにして、ストレージサイズの上限 (ギビバイト単位) を設定します。

Amazon RDS ストレージの自動スケーリングが DB インスタンスに対して使用可能なことを確認するには、Amazon RDS API `DescribeValidDbInstanceModifications` オペレーションを既存インスタンスに対して使用するか、インスタンスの作成前に

DescribeOrderableDBInstanceOptions オペレーションを使用します。戻り値の以下のフィールドを確認します。

- SupportsStorageAutoscaling – DB インスタンスでストレージの自動スケーリングがサポートされているかどうかを示します。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

DB インスタンスのストレージの自動スケーリング設定を変更する

既存の Amazon RDS DB インスタンスのストレージの自動スケーリングをオンにすることができます。また、Amazon RDS が DB インスタンスに対して割り当てることができるストレージの上限を変更することもできます。

ストレージの自動スケーリング設定を変更しても、データベースを再起動する必要はなく、ダウンタイムも発生しません。変更は、データベースの操作を中断することなく、すぐに有効になります。

コンソール

DB インスタンスのストレージ自動スケーリング設定を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択してから、[変更] を選択します。[Modify DB instance] ページが表示されます。
4. [自動スケーリング] セクションのストレージ制限を変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
5. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
6. 確認ページで、変更内容を確認します。正しい場合は、[DB インスタンスを変更] を選択して変更を保存します。正しくない場合は、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更を取り消します。

   ストレージの自動スケーリングの制限の変更はただちに実行されます。この設定は、[Apply immediately] 設定を無視します。

AWS CLI

DB インスタンスのストレージ自動スケーリング設定を変更するには、AWS CLI コマンド modify-db-instance を使用します。次のパラメータを設定します。

- --max-allocated-storage – ストレージサイズの上限 (ギビバイト単位) を設定します。値が --allocated-storage パラメータを超える場合、ストレージの自動スケーリングはオンになります。値が --allocated-storage パラメータと同一の場合、ストレージの自動スケーリングはオフになります。

DB インスタンスに対して Amazon RDS ストレージの自動スケーリングが使用可能であることを確認するには、AWS CLI describe-valid-db-instance-modifications コマンドを使用します。インスタンスを作成する前に、インスタンスクラスに基づいて確認するには、describe-orderable-db-instance-options コマンドを使用します。戻り値の以下のフィールドを確認します。

- SupportsStorageAutoscaling – DB インスタンスでストレージの自動スケーリングがサポートされているかどうかを示します。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

RDS API

DB インスタンスのストレージ自動スケーリング設定を変更するには、Amazon RDS API オペレーション ModifyDBInstance を使用します。次のパラメータを設定します。

- MaxAllocatedStorage – ストレージサイズの上限 (ギビバイト単位) を設定します。

Amazon RDS ストレージの自動スケーリングが DB インスタンスに対して使用可能なことを確認するには、Amazon RDS API DescribeValidDbInstanceModifications オペレーションを既存インスタンスに対して使用するか、インスタンスの作成前に DescribeOrderableDBInstanceOptions オペレーションを使用します。戻り値の以下のフィールドを確認します。

- SupportsStorageAutoscaling – DB インスタンスでストレージの自動スケーリングがサポートされているかどうかを示します。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

DB インスタンスを手動でスケールダウンまたはスケールインする

Amazon RDS は、需要の増加に合わせたストレージの自動スケーリングを提供しています。ただし、スケールダウンとスケールインには制限があります。

- RDS ストレージ – RDS は需要の増加に応じてストレージの自動スケールアップをサポートしていますが、自動的にスケールダウンすることはありません。
- リードレプリカ – RDS は、リードレプリカの自動スケールアウト (追加) またはスケールイン (削除) をサポートしていません。ロード要件に従って、リードレプリカを手動で追加または削除する必要があります。

RDS リソースをスケールダウンするには、次の手動アクションを実行します。

- ストレージの場合、`modify-db-instance` コマンドを使用して DB インスタンスの割り当てられたストレージを手動で減らすことはできません。代わりに、次のいずれかの方法を選択します。
  - DB エンジンがブルー/グリーンデプロイをサポートしている場合は、ブルー/グリーンデプロイを使用します。小さいストレージサイズでグリーンデータベースを作成し、グリーンデータベースをブルーデータベースに昇格させます。詳細については、「ストレージとパフォーマンスの設定を変更する」を参照してください。
  - 小さく割り当てられたストレージで新しい DB インスタンスを作成し、現在のデータベースから新しく作成されたデータベースインスタンスにデータを手動で移行して、データベースエンドポイントを切り替えます。
- リードレプリカの場合は、RDS コンソールまたは AWS CLI を使用して未使用のレプリカを手動で削除します。

DB インスタンスのストレージの自動スケーリングをオフにする

Amazon RDS で Amazon RDS DB インスタンスのストレージを自動的に増やす必要がなくなった場合は、ストレージの自動スケーリングをオフにできます。オフにしても、DB インスタンスのストレージ領域は手動で増やすことができます。

コンソール

DB インスタンスのストレージの自動スケーリングをオフにするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択してから、[変更] を選択します。[Modify DB instance] ページが表示されます。
4. [Storage autoscaling (ストレージの自動スケーリング)] セクションの [Enable storage autoscaling (ストレージの自動スケーリングを有効にする)] チェックボックスをオフにします。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
5. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
6. 確認ページで、変更内容を確認します。正しい場合は、[DB インスタンスを変更] を選択して変更を保存します。正しくない場合は、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更を取り消します。

ストレージの自動スケーリングの制限の変更はただちに実行されます。この設定は、[Apply immediately] 設定を無視します。

AWS CLI

DB インスタンスのストレージの自動スケーリングをオフにするには、AWS CLI コマンド `modify-db-instance` および次のパラメータを使用します。

- `--max-allocated-storage` – `--allocated-storage` 設定と同等の値を指定して、指定された DB インスタンスでそれ以上 Amazon RDS ストレージの自動スケーリングが行われるのを防ぐことができます。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

RDS API

DB インスタンスのストレージの自動スケーリングをオフにするには、Amazon RDS API オペレーション `ModifyDBInstance` を使用します。次のパラメータを設定します。

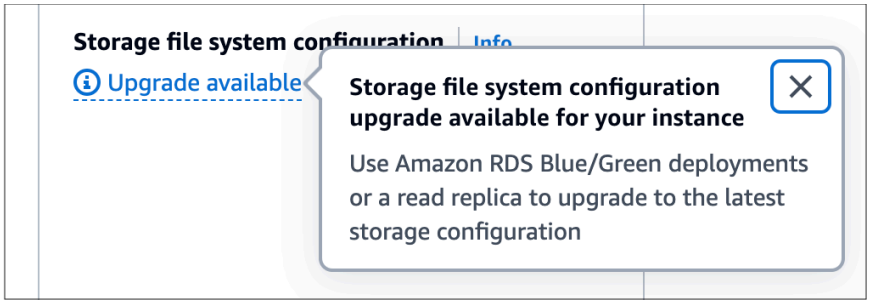
- `MaxAllocatedStorage` – `AllocatedStorage` 設定と同等の値を指定して、指定された DB インスタンスでそれ以上 Amazon RDS ストレージの自動スケーリングが行われるのを防ぐことができます。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

DB インスタンスのストレージファイルシステムのアップグレード

ほとんどの RDS DB インスタンスでは、RDS for MariaDB、MySQL、および PostgreSQL データベースの最大ストレージサイズは 64 TiB です。ただし、一部の古い 32 ビットファイルシステムでは、ストレージ容量が少なくなっています。DB インスタンスのストレージ容量を確認するには、describe-valid-db-instance-modifications AWS CLI コマンドを使用します。

RDS は、ストレージシステムに 16 TiB のストレージサイズ、2 TiB のファイルサイズ制限、または最適化されていない書き込みがあるかどうかをチェックします。DB インスタンスがこれらの条件を満たしている場合、RDS はファイルシステム設定がアップグレードの対象であることを警告します。DB インスタンスのアップグレード資格は、DB インスタンスの詳細ページの [ストレージ] パネルで確認できます。



DB インスタンスがファイルシステムアップグレードの対象である場合は、次のいずれかの方法を使用します。

- Blue/Green デプロイを作成し、[ストレージファイルシステム設定のアップグレード] を指定します。

  このオプションは、グリーン環境のファイルシステムを適切な設定にアップグレードします。その後、ブルー/グリーンデプロイを切り替えて、グリーン環境を新しい本番環境に昇格できます。詳細な手順については、「the section called “ブルー/グリーンデプロイの作成”」を参照してください。
- DB インスタンスのリードレプリカを作成し、[ストレージファイルシステム設定のアップグレード] を指定します。

  このオプションは、リードレプリカのファイルシステムを適切な設定にアップグレードします。次に、リードレプリカをスタンドアロンインスタンスに昇格させることができます。詳細な手順については、「the section called “リードレプリカの作成”」を参照してください。

ストレージのアップグレード中は、データベースエンジンは使用できません。ストレージ設定のアップグレードは I/O の負荷が高い操作であり、リードレプリカとブルー/グリーンデプロイでは作成時間が長くなります。次の条件の両方が満たされると、ストレージのアップグレードプロセスが高速になります。

- ソース DB インスタンスは、プロビジョンド IOPS SSD (io1 または io2 Block Express) ストレージを使用します。
- インスタンスサイズが 4xlarge 以上のグリーン環境またはリードレプリカをプロビジョニングしました。

汎用 SSD (gp2) ストレージを含むストレージをアップグレードすると、I/O クレジットバランスを使い切る可能性があり、よってよりアップグレード時間が長くなります。詳細については、「the section called “DB インスタンスストレージ”」を参照してください。

ストレージのアップグレード中に、次の条件の両方が満たされた場合、RDS はグリーンインスタンスまたはリードレプリカに割り当てられたストレージサイズを 10% 増やします。

- ソース DB インスタンスのストレージ消費量は、割り当てられたストレージサイズの 90% 以上です。
- ストレージの自動スケーリングが有効になっています。

新しいストレージサイズがインスタンスに設定された最大割り当てストレージ以上の場合、RDS は自動スケーリングをオフにします。ストレージのアップグレードの開始前にストレージのオートスケーリングが無効になっている場合、アップグレード中にストレージサイズは増加しません。

プロビジョンド IOPS SSD ストレージの設定を変更する

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、プロビジョンド IOPS SSD ストレージを使用する DB インスタンスの設定を変更できます。必要なストレージタイプ、割り当て済みストレージ、プロビジョンド IOPS の量を指定します。この範囲は、データベースエンジンとインスタンスタイプによって異なります。

インスタンスにプロビジョニングされた IOPS の量を減少することはできますが、ストレージサイズを減少させることはできません。

ほとんどの場合、ストレージのスケーリングには停止が必要ではなく、サーバーのパフォーマンスを低下させません。DB インスタンスのストレージ IOPS を変更すると、DB インスタンスのステータスは storage-optimization になります。

> **Note**
>
> ストレージの最適化には数時間かかることがあります。インスタンスでストレージの最適化が完了したら、追加のストレージ変更を行うことができます。任意の 24 時間以内に最大 4 回のストレージ変更を実行できます。

各データベースエンジンで使用できる割り当て済みストレージとプロビジョンド IOPS の範囲については、「プロビジョンド IOPS SSD ストレージ」を参照してください。

コンソール

DB インスタンスのプロビジョンド IOPS の設定を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

   DB インスタンスの一覧をフィルターするには、[Filter databases (データベースのフィルター)] に、結果をフィルターするために使用する Amazon RDS のテキスト文字列を入力します。その文字列を名前に含む DB インスタンスのみが表示されます。
3. 変更するプロビジョンド IOPS を使う DB インスタンスを選択します。
4. [Modify] を選択します。
5. [DB インスタンスを変更] ページの [ストレージタイプ] で [プロビジョンド IOPS SSD (io1)[プロビジョンド or [プロビジョンド IOPS SSD (io2)[プロビジョンド を選択します。
6. プロビジョンド IOPS に値を入力します。

   ストレージの割り当て または プロビジョンド IOPS で指定した値が、他のパラメータでサポートされている制限を超えている場合、警告メッセージが表示されます。このメッセージには他のパラメータで必要な値の範囲が表示されます。
7. [Continue] を選択します。
8. DB インスタンスの変更をすぐに適用するには、[Scheduling of modifications] (変更のスケジュール) セクションの [Apply immediately] (すぐに適用) を選択します。または、[次に予定されるメンテナンスウィンドウ中に適用します] を選択して、次のメンテナンスウィンドウ中に変更を適用します。
9. 変更するパラメータを確認し、[Modify DB instance] (DB インスタンスの変更) を選択して変更を完了します。

ストレージの割り当てまたはプロビジョンド IOPS の新しい値が ステータス 列に表示されます。

AWS CLI

DB インスタンスのプロビジョンド IOPS の設定を変更するには、AWS CLI の `modify-db-instance` コマンドを使用します。以下のパラメータを設定します:

- `--storage-type` – プロビジョンド IOPS には `io1` または `io2` を指定します。
- `--allocated-storage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `--iops` – DB インスタンスのプロビジョンド IOPS の新しい値。1 秒あたりの I/O オペレーション数で表されます。
- `--apply-immediately` – `--apply-immediately` を使用して、すぐに変更を適用します。`--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

RDS API

DB インスタンスのプロビジョンド IOPS の設定を変更するには、Amazon RDS API オペレーションの `ModifyDBInstance` を使用します。以下のパラメータを設定します:

- `StorageType` – プロビジョンド IOPS には `io1` または `io2` を指定します。
- `AllocatedStorage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `Iops` – DB インスタンスの新しい IOPS レート。1 秒あたりの I/O オペレーション数で表されます。
- `ApplyImmediately` – このオプションを `True` に設定して、すぐに変更を適用します。または `False` (デフォルト) を設定して、次のメンテナンスウィンドウ中にストレージの変更を適用します。

I/O を大量に消費するストレージの変更

Amazon RDS DB インスタンスは、データベースおよびログストレージに Amazon Elastic Block Store (EBS) ボリュームを使用します。要求されるストレージの量によって、RDS (RDS for SQL Server を除く) は自動的に複数の Amazon EBS ボリューム全体をストライプして、パフォーマンスを向上させます。SSD ストレージタイプの RDS DB インスタンスは、RAID 0 構成の 1 つまたは 4

つのストライプ化された Amazon EBS ボリュームによってバックアップされます。設計上、RDS DB インスタンスのストレージ変更操作が進行中のデータベース操作に与える影響は最小限に抑えられます。

ほとんどの場合、ストレージのスケーリング変更は Amazon EBS レイヤーに完全にオフロードされ、データベースに対して透過的に行われます。通常、このプロセスは数分で完了します。ただし、古い RDS ストレージボリュームの中には、サイズ、プロビジョンド IOPS、またはストレージタイプを変更するために別のプロセスが必要なものがあります。これには、I/O を大量に消費する可能性のある操作を使用してデータのフルコピーを作成することが含まれます。

ストレージの変更では、次のいずれかの要因が当てはまる場合、I/O を大量に消費する操作を使用します。

- ソースストレージタイプはマグネティックです。マグネティックストレージは伸縮自在のボリューム変更をサポートしていません。
- RDS DB インスタンスは、1 ボリュームまたは 4 ボリュームの Amazon EBS レイアウト上にはありません。拡張モニタリングメトリクスを使用すると、RDS DB インスタンスで使用されている Amazon EBS ボリュームの数を確認できます。詳しくは、「RDS コンソールでの OS メトリクスの表示」を参照してください。
- 変更リクエストのターゲットサイズにより、RDS for MariaDB、MySQL、および PostgreSQL インスタンスでは 400 GiB 以上、RDS for Oracle では 200 GiB 以上の割り当てストレージが増加します。ストレージの自動スケーリング操作は、DB インスタンスに割り当てられたストレージサイズがこれらのしきい値を超えて増加した場合にも同じ効果があります。

ストレージの変更に I/O を大量に消費する操作が含まれる場合、I/O リソースが消費され、DB インスタンスの負荷が増加します。汎用 SSD (gp2) ストレージを含む I/O を大量に消費する操作でストレージを変更すると、I/O クレジット残高を使い切る可能性があり、変換時間が長くなることがあります。

このようなストレージの変更リクエストは、ベストプラクティスとして、ストレージの変更操作の完了に必要な時間を短縮するために、ピーク時間帯以外にスケジュールすることをお勧めします。DB インスタンスのリードレプリカを作成し、リードレプリカのストレージを変更を実行することもできます。次に、リードレプリカがプライマリ DB インスタンスに昇格されます。詳しくは、「DB インスタンスのリードレプリカの操作」を参照してください。

詳細については、次を参照してください。「割り当てられたストレージを増やそうとすると、Amazon RDS DB インスタンスが変更中の状態で止まるのはなぜですか?」

汎用 SSD (gp3) ストレージの設定変更

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、汎用 SSD (gp3) ストレージを使用する DB インスタンスの設定を変更できます。必要なストレージタイプ、割り当て済みストレージ、プロビジョンド IOPS の量、ストレージのスループットを指定します。

DB インスタンスのプロビジョンド IOPS の量とストレージスループットを減らすことはできますが、ストレージサイズを減らすことはできません。

ほとんどの場合、ストレージをスケーリングしても停止する必要はありません。DB インスタンスのストレージ IOPS を変更すると、DB インスタンスのステータスは storage-optimization になります。ストレージの最適化中は、レイテンシーが高くなることが予想されますが、それでも数ミリ秒の範囲内です。ストレージ変更後、DB インスタンスは完全に動作します。

ボリュームごとの最適化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。最適化中、`StorageOperationStatus` は `Optimizing` に設定されます。

> **Note**
>
> インスタンスでストレージの最適化が完了したら、追加のストレージ変更を行うことができます。任意の 24 時間以内に最大 4 回のストレージ変更を実行できます。

各データベースエンジンで使用できる割り当て済みストレージ、プロビジョンド IOPS、ストレージのスループットの範囲については、「gp3 ストレージ (推奨)」を参照してください。

コンソール

DB インスタンスのストレージのパフォーマンス設定を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

   DB インスタンスの一覧をフィルターするには、[Filter databases (データベースのフィルター)] に、結果をフィルターするために使用する Amazon RDS のテキスト文字列を入力します。その文字列を名前に含む DB インスタンスのみが表示されます。
3. 変更する gp3 ストレージを使用する DB インスタンスを選択します。

4. [Modify] (変更) を選択します。
5. [DB インスタンスを変更] ページで、[ストレージタイプ] に [汎用 SSD (gp3)] を選択し、次の操作を行います。

   a. プロビジョンド IOPS で、値を選択します。

      ストレージの割り当て または プロビジョンド IOPS で指定した値が、他のパラメータでサポートされている制限を超えている場合、警告メッセージが表示されます。このメッセージには他のパラメータで必要な値の範囲が表示されます。

   b. ストレージスループット で値を選択します。

      プロビジョンド IOPS またはストレージのスループットで指定した値が、他のパラメータでサポートされている制限を超えている場合、警告メッセージが表示されます。このメッセージには他のパラメータで必要な値の範囲が表示されます。

6. [Continue] を選択します。
7. DB インスタンスの変更をすぐに適用するには、[Scheduling of modifications] (変更のスケジュール) セクションの [Apply immediately] (すぐに適用) を選択します。または、[次に予定されるメンテナンスウィンドウ中に適用します] を選択して、次のメンテナンスウィンドウ中に変更を適用します。
8. 変更するパラメータを確認し、[Modify DB instance] (DB インスタンスの変更) を選択して変更を完了します。

   プロビジョンド IOPS の新しい値が ステータス 列に表示されます。

AWS CLI

DB インスタンスのストレージのパフォーマンス設定を変更するには、AWS CLI コマンド [modify-db-instance](#) を使用します。以下のパラメータを設定します:

- `--storage-type` – 汎用 SSD (gp3) の場合は `gp3` に設定します。
- `--allocated-storage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `--iops` – DB インスタンスのプロビジョンド IOPS の新しい値。1 秒あたりの I/O オペレーション数で表されます。
- `--storage-throughput` – DB インスタンスの新しいストレージスループットで、MIBP で表します。

- `--apply-immediately` – `--apply-immediately` を使用して、すぐに変更を適用します。`--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

RDS API

DB インスタンスのストレージのパフォーマンス設定を変更するには、Amazon RDS API オペレーション `ModifyDBInstance` を使用します。以下のパラメータを設定します:

- `StorageType` – 汎用 SSD (gp3) の場合は `gp3` に設定します。
- `AllocatedStorage` – DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `Iops` – DB インスタンスの新しい IOPS レート。1 秒あたりの I/O オペレーション数で表されます。
- `StorageThroughput` – DB インスタンスの新しいストレージスループットで、MIBP で表します。
- `ApplyImmediately` – このオプションを `True` に設定して、すぐに変更を適用します。または `False` (デフォルト) を設定して、次のメンテナンスウィンドウ中にストレージの変更を適用します。

専用ログボリューム (DLV) を使用する

プロビジョンド IOPS (PIOPS) ストレージを使用する DB インスタンスには、専用のログボリューム (DLV) を使用できます。DLV は、PostgreSQL データベーストランザクションログと MySQL または MariaDB の REDO ログとバイナリログを、データベーステーブルが配置されたボリュームとは別のストレージボリュームに移動します。DLV を使用すると、トランザクション書き込みロギングの効率と一貫性が向上します。DLV は、割り当てられたストレージの容量が大きいデータベース、1 秒あたりの I/O (IOPS) 要件が高い、または遅延の影響を受けやすいワークロードがあるデータベースに最適です。

DLV は PIOPS ストレージ (io1 および io2 Block Express) でサポートされており、1,024 GiB の固定サイズと 3,000 のプロビジョンド IOPS で作成されます。

Amazon RDS は、次のバージョンについて、すべての AWS リージョン の DLV でサポートします。

- MariaDB 10.6.7 以上の 10 バージョン
- MySQL 8.0.28 以上の 8.0 バージョン、MySQL 8.4.3 以上の 8.4 バージョン
- PostgreSQL 13.10 以上の 13 バージョン、14.7 以上の 14 バージョン、15.2 以上の 15 バージョン

RDS は、マルチ AZ 配置で DLV をサポートします。マルチ AZ インスタンスを変更または作成すると、プライマリとセカンダリの両方に DLV が作成されます。

RDS はリードレプリカによる DLV をサポートします。プライマリ DB インスタンスで DLV が有効になっている場合、DLV を有効にした後に作成されたすべてのリードレプリカにも DLV が設定されます。DLV への切り替え前に作成されたリードレプリカは、明示的に変更されない限り、有効になりません。DLV を有効にする前にプライマリインスタンスにアタッチされていたすべてのリードレプリカも、DLV を使用するように手動で変更することが推奨されます。

> **Note**
>
> 5 TiB 以上のデータベース設定には DLV をお勧めします。

DLV の利点について詳しくは、以下のブログ投稿を参照してください。

- Amazon RDS 専用ログボリュームによるデータベースパフォーマンスの向上
- Amazon RDS for PostgreSQL を専用ログボリュームでベンチマークする
- Percona ドキュメントの「Maximizing performance of AWS RDS for MySQL with dedicated log volumes」

各データベースエンジンで使用できる割り当て済みストレージ、プロビジョンド IOPS、ストレージのスループットの範囲については、「プロビジョンド IOPS SSD ストレージ」を参照してください。

トピック

- DLV を有効または無効にする際の考慮事項
- DB インスタンス作成時に DLV を有効にする
- 既存の DB インスタンスで DLV を有効にする
- DLV ストレージのモニタリング

DLV を有効または無効にする際の考慮事項

DLV の有効化と無効化には時間がかかり、ダウンタイムが発生する可能性があります。このプロセスでは、有効化時にすべてのトランザクションログ、または REDO ログとバイナリログ (データベースエンジンによって異なります) を新しいボリュームにコピーするか、無効化時に元のストレージに戻します。このオペレーションの期間は、いくつかの要因の影響を受けます。

- トランザクションログの数:
  - トランザクションが多いデータベースが大きいほどログが生成され、コピーにかかる時間が長くなります。
  - レプリケーションスロットが非アクティブの場合、またはレプリケーションが遅延している場合、トランザクションログがプライマリ DB インスタンスに蓄積され、コピーに必要な時間が長くなります。レプリケーションが最新であることを確認し、不要なスロットをすべて削除します。
- ストレージ設定:
  - DB インスタンスの EBS 帯域幅 – 帯域幅が大きいほど、データ転送が速くなります。
  - プロビジョンド IOPS の数 – 1 秒あたりの入出力オペレーション (IOPS) が増えると、コピープロセスが高速化されます。
- データベースアクティビティ – 設定中のデータベースアクティビティのレベルが高いと、プロセスが遅くなる可能性があります。

ダウンタイムを最小限に抑えるために、アクティビティが少ない期間やメンテナンス期間中に計画とスケジュールを設定することをお勧めします。

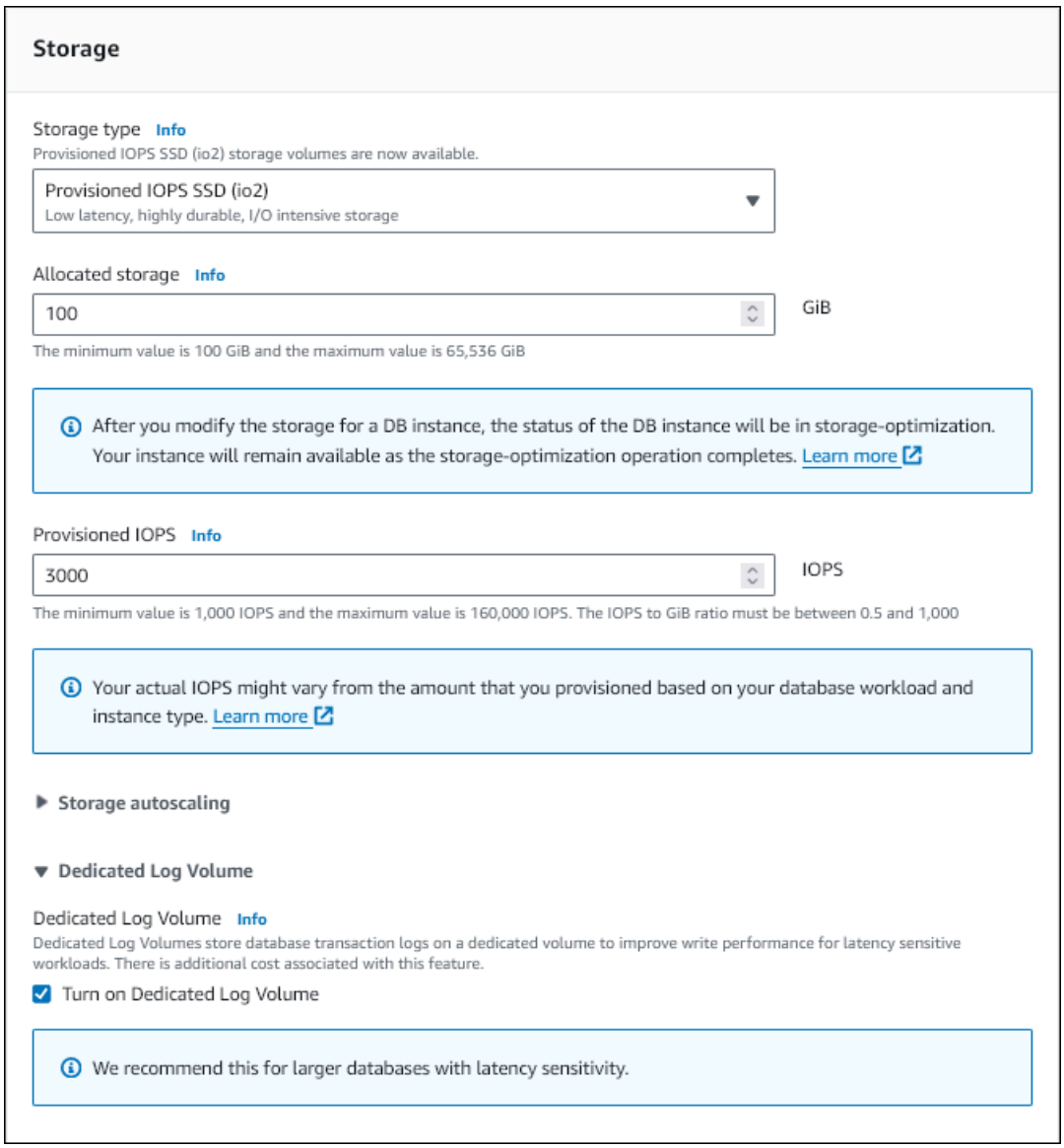
DB インスタンス作成時に DLV を有効にする

DLV を有効にした DB インスタンスを作成するには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用できます。

コンソール

新しい DB インスタンスで DLV を有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベースの作成] を選択します。
3. [DB インスタンスの作成] ページで、DLV をサポートする DB エンジンを選択します。
4. ストレージ:
   a. [プロビジョンド IOPS SSD (io1)] または [プロビジョンド IOPS SSD (io2)] を選択します。
   b. 使用する [割り当て済みストレージ] と [プロビジョンド IOPS] を入力します。
   c. [専用ログボリューム] を展開して、[専用ログボリュームを有効にする] を選択します。



5. 必要に応じて、その他の設定を選択します。
6. [データベースの作成] を選択します。

データベースが作成されると、専用ログボリュームの値がデータベースの詳細ページの [設定] タブに表示されます。

CLI

プロビジョンド IOPS ストレージを使用して DB インスタンスを作成する際に DLV を有効にするには、AWS CLI コマンド create-db-instance を使用します。以下のパラメータを設定します:

- `--dedicated-log-volume` – 専用ログボリュームを有効にします。
- `--storage-type` – プロビジョンド IOPS には `io1` または `io2` を指定します。
- `--allocated-storage` - DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `--iops` – DB インスタンスのプロビジョンド IOPS の量 (1 秒あたりの I/O オペレーション数単位)。

RDS API

プロビジョンド IOPS ストレージを使用して DB インスタンスを作成する際に DLV を有効または無効にするには、Amazon RDS API オペレーション CreateDBInstance を使用します。以下のパラメータを設定します:

- `DedicatedLogVolume` — `true` に設定して、専用のログボリュームを有効にします。
- `StorageType` – プロビジョンド IOPS には `io1` または `io2` を指定します。
- `AllocatedStorage` - DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。
- `Iops` – この DB インスタンスの新しい IOPS レート (1 秒あたりの I/O オペレーション数単位)。

既存の DB インスタンスで DLV を有効にする

DB インスタンスを更新して DLV を有効にするには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用できます。

DB インスタンスの DLV 設定を変更したら、DB インスタンスを再起動する必要があります。

コンソール

既存の DB インスタンスで DLV を有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

   DB インスタンスの一覧をフィルターするには、[Filter databases (データベースのフィルター)] に、結果をフィルターするために使用する Amazon RDS のテキスト文字列を入力します。その文字列を名前に含む DB インスタンスのみが表示されます。
3. 変更するプロビジョンド IOPS ストレージを使用する DB インスタンスを選択します。

4. [Modify] (変更) を選択します。
5. [DB インスタンスを変更] ページで、以下を実行します。
   - [ストレージ] では、[専用ログボリューム] を展開して、[専用ログボリュームを有効にする] を選択します。
6. [続行] をクリックしてください。
7. DB インスタンスへの変更を直ちに適用するには、[すぐに適用] をクリックします。または、[次に予定されるメンテナンスウィンドウ中に適用します] を選択して、次のメンテナンスウィンドウ中に変更を適用します。
8. 変更するパラメータを確認し、[Modify DB instance] (DB インスタンスの変更) を選択して変更を完了します。

専用ログボリュームの新しい値がデータベースの詳細ページの [設定] タブに表示されます。

CLI

プロビジョンド IOPS ストレージを使用して既存の DB インスタンスで DLV を有効または無効にするには、AWS CLI コマンド `modify-db-instance` を使用します。以下のパラメータを設定します:

- `--dedicated-log-volume` – 専用ログボリュームを有効にします。

  `--no-dedicated-log-volume` (デフォルト) を使用して、専用ログボリュームを無効にします。
- `--apply-immediately` – `--apply-immediately` を使用して、すぐに変更を適用します。

  `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

RDS API

プロビジョンド IOPS ストレージを使用して既存の DB インスタンスで DLV を有効または無効にするには、Amazon RDS API オペレーション `ModifyDBInstance` を使用します。以下のパラメータを設定します:

- `DedicatedLogVolume` - このオプションを `true` に設定して、専用ログボリュームを有効にします。

このオプションを false に設定して、専用ログボリュームを無効にします。これは、デフォルト値です。

- ApplyImmediately – このオプションを True に設定して、すぐに変更を適用します。

  または False (デフォルト) を設定して、次のメンテナンスウィンドウ中にストレージの変更を適用します。

DLV ストレージのモニタリング

CloudWatch の FreeStorageSpaceLogVolume メトリクスを使用して、DLV ストレージの使用状況をモニタリングできます。

RDS for PostgreSQL の次のクエリを使用して、トランザクションログが占めるサイズを確認することができます。

```
SELECT pg_size_pretty(COALESCE(sum(size), 0)) AS total_wal_generated_size
FROM pg_catalog.pg_ls_waldir();
```

DLV のストレージが不足すると、DB インスタンスは storage-full 状態になり、ダウンタイムが発生します。

DB インスタンスを削除する

> **Warning**
>
> データベースインスタンスを削除すると、そのすべてのデータが削除され、関連するリソースに影響する可能性があります。先に進む前に、重要なデータをバックアップしていることを確認してください。

AWS マネジメントコンソール、AWS CLI、RDS API を使用して DB インスタンスを削除できます。Aurora DB クラスター内の DB インスタンスを削除する場合、「Aurora DB クラスターと DB インスタンスを削除する」を参照してください。

トピック

- DB インスタンスの削除の前提条件
- DB インスタンスを削除する場合の考慮事項
- DB インスタンスを削除する

DB インスタンスの削除の前提条件

DB インスタンスの削除を試みる前に、削除保護がオフになっていることを確認してください。デフォルトでは、コンソールで作成された DB インスタンスの削除保護がオンになっています。

DB インスタンスで削除保護が有効になっている場合は、インスタンス設定を変更することで削除保護を無効にできます。データベースの詳細ページで [変更] を選択するか、modify-db-instance コマンドを呼び出します。このオペレーションでは停止は発生しません。詳細については、「DB インスタンスの設定」を参照してください。

DB インスタンスを削除する場合の考慮事項

DB インスタンスを削除すると、インスタンスの復旧可能性、バックアップの可用性、リードレプリカのステータスに影響します。以下の問題について考えます。

- 最終 DB スナップショットを作成するかどうかを選択できます。次のオプションがあります。
  - 最終スナップショットを作成する場合、そのスナップショットを使用して、削除した DB インスタンスを復元できます。RDS は、最終スナップショットと以

前に作成した手動スナップショットの両方を保持します。DB インスタンスが creating、deleting、failed、incompatible-create、incompatible-network、incompatible-restore、insufficient-capacity、restore-error、または upgrade-failed の状態の場合、最終的な DB スナップショットを作成することはできません。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

- 最終スナップショットを作成しない場合、後で復元できる最終スナップショットは存在しません。削除した DB インスタンスを復元する場合は、自動バックアップを保持するか、以前の手動スナップショットを使用して DB インスタンスを以前のスナップショットの時点に復元します。

- 自動バックアップを保持するかどうかを選択できます。次のオプションがあります。
  - 自動バックアップを保持する場合、自動バックアップは、RDS により DB インスタンスの削除時に有効な保持期間だけ保持されます。自動バックアップを使用して、DB インスタンスを保持期間中に復元できますが、保持期間終了後は復元できません。保持期間は、最終 DB スナップショットを作成するかどうかに関係なく有効です。保持されている自動バックアップを削除するには、「保持している自動バックアップの削除」を参照してください。
  - 自動バックアップと手動スナップショットを保持すると、削除されるまで請求が発生します。詳細については、「保持コスト」を参照してください。
  - 自動バックアップを保持しない場合、DB インスタンスと同じ AWS リージョン リージョンにある自動バックアップは、RDS により削除されます。これらのバックアップを復元することはできません。自動バックアップが別の AWS リージョン に複製されている場合に自動バックアップを保持しない場合、RDS が保持します。詳細については、「別の AWS リージョン への自動バックアップのレプリケーション」を参照してください。
- DB インスタンスを削除しても、RDS は手動 DB スナップショットを削除しません。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
- すべての RDS リソースを削除する場合、次のリソースには請求料金が発生することに注意してください。
  - DB インスタンス
  - DB スナップショット
  - DB クラスター

  リザーブドインスタンスを購入した場合、インスタンスの購入時に合意した契約に従って請求されます。詳細については、「Amazon RDS 向けリザーブド DB インスタンス」を参照してください。AWS Cost Explorer を使用して、すべての AWS リソースの請求情報を取得できます。詳細については、「AWS Cost Explorer によるコストの分析」を参照してください。

- 同じ AWS リージョン にリードレプリカがある DB インスタンスを削除すると、各リードレプリカはスタンドアロン DB インスタンスに自動的に昇格されます。詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。異なる AWS リージョン にリードレプリカがある DB インスタンスの場合、クロスリージョンリードレプリカのソース DB インスタンスの削除について、「クロスリージョンレプリケーションに関する考慮事項」を参照してください。
- DB インスタンスのステータスが `deleting` の場合、その CA 認定の値は、RDS コンソールにも、AWS CLI コマンドまたは RDS API オペレーションの出力にも表示されません。CA 認定の詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
- DB インスタンスの削除に必要な時間は、バックアップ保持期間、削除するデータの量、最終スナップショットを作成するか、最終スナップショットが増分か完全かによって異なります。最終的なスナップショットを作成しなくても、サービスはデータの整合性を確保するために内部オペレーションを実行する必要があります。これらのオペレーションにより、削除時間が長くなる場合があります。スキップすることはできません。DB インスタンスのステータスは、すべてのオペレーションが完了するまで削除中状態のままです。DB インスタンスの削除に必要な時間を短縮するために、DB インスタンスを削除する前に手動スナップショットを作成することをお勧めします。
- 意図せずに削除されたインスタンスを復元する必要がある場合は、すぐに AWS サポートチームにお問い合わせください。削除リクエスト後、最大 6 日間は復元が可能な場合があります。

DB インスタンスを削除する

AWS マネジメントコンソール、AWS CLI、RDS API を使用して DB インスタンスを削除できます。以下を実行する必要があります。

- DB インスタンスの名前を入力します。
- インスタンスの最終 DB スナップショットを取得するオプションを有効/無効にする
- 自動バックアップを保持するオプションを有効/無効にする

> **Note**
>
> 削除保護が有効になっている場合、DB インスタンスを削除することはできません。詳細については、「DB インスタンスの削除の前提条件」を参照してください。

コンソール

DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、削除する DB インスタンスを選択します。
3. [アクション] で、[削除] を選択します。
4. DB インスタンスの最終 DB スナップショットを作成するには、[最終スナップショットを作成しますか?] を選択します。
5. 最終スナップショットの作成を選択した場合は、[Final snapshot name (最終スナップショット名)] を入力します。
6. 自動バックアップを保持するには、[Retain automated backups (自動バックアップの保持)] を選択します。
7. ボックスに「**delete me**」と入力します。
8. [削除] を選択します。

AWS CLI

アカウント内の DB インスタンスのインスタンス ID を確認するには、describe-db-instances コマンドを呼び出します。

```
aws rds describe-db-instances --query 'DBInstances[*].[DBInstanceIdentifier]' --output
 text
```

AWS CLI を使用して DB インスタンスを削除するには、以下のオプションを使用して delete-db-instance コマンドを呼び出します。

- `--db-instance-identifier`
- `--final-db-snapshot-identifier`、または `--skip-final-snapshot`

Example最終スナップショットを作成し、自動バックアップを保持しない場合

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
```

```
    --final-db-snapshot-identifier mydbinstancefinalsnapshot \
    --delete-automated-backups
```

Windows の場合:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --final-db-snapshot-identifier mydbinstancefinalsnapshot ^
    --delete-automated-backups
```

Example自動バックアップを保持し、最終スナップショットを作成しない場合

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
    --skip-final-snapshot \
    --no-delete-automated-backups
```

Windows の場合:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --skip-final-snapshot ^
    --no-delete-automated-backups
```

RDS API

Amazon RDS API を使用して DB インスタンスを削除するには、以下のパラメータを指定して DeleteDBInstance オペレーションを呼び出します。

- DBInstanceIdentifier
- FinalDBSnapshotIdentifier、または SkipFinalSnapshot

チュートリアル: 開発から本番環境までの MySQL DB インスタンス環境の管理

序章

Amazon RDS DB インスタンスを管理する一般的なシナリオでは、初期開発から本番デプロイまでのライフサイクルを監視します。このチュートリアルでは、データベースのパフォーマンスを最適に保ち、変化する運用ニーズに合わせて適応できるように、主要なタスクを処理するためのガイダンスを提供します。さらに、開発環境と本番環境の間で行われた変更を同期して一貫性と信頼性を確保するオプションについても概説しています。

これらの手順を実行することで、次のことが理解できます。

- Amazon RDS タグの追加と更新、ストレージの拡張、リードレプリカの作成、リソースの削除など、MySQL DB インスタンスで特定のタスクを実行する方法。
- 包括的なテストと検証のために、本番環境から開発環境に更新を同期する方法。

このチュートリアルを完了するには、次のタスクを実行します。

1. MySQL DB インスタンスを作成します。
2. Amazon RDS タグを追加して、DB インスタンスを開発環境として分類します。
3. DB インスタンスのストレージ容量を増やして、ワークロードの増加に対応します。
4. リードレプリカを作成して、DB インスタンスの耐障害性と可用性を強化します。
5. Amazon RDS タグを更新して、DB インスタンスを本番環境として分類します。
6. 追加コストが発生しないように、不要になった DB インスタンスを削除します。
7. 次のステップ: 開発インスタンスを本番環境と同期して、環境間で一貫性を保つ

前提条件

開始する前に、以下のセクションのステップを完了してください。

- AWS アカウント へのサインアップ

タグを追加して DB インスタンスを開発環境として分類する

DB インスタンスを開発環境として分類するには、作成したインスタンスに Amazon RDS タグを追加します。Amazon RDS タグは、定義して RDS インスタンスに関連付けるキーと値のペアです。AWS リソースにタグを付けることで、開発リソースと本番環境用 AWS リソースを区別しやすくなります。Amazon RDS タグの詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

1. Amazon RDS コンソールで、[データベース] を選択します。
2. タグを付ける DB インスタンスを選択します。
3. [詳細] セクションで、[タグ] セクションまでスクロールします。
4. [タグを管理] を選択し、[新しいタグを追加] を選択します。
5. [タグキー] と [値] の値を入力します。例えば、タグキーとして environment を使用し、値を dev に設定することで、そのデータベースインスタンスが開発環境の一部であることを指定できます。
6. [新しいタグを追加]、[変更を保存] の順に選択します。

これで、DB インスタンスが開発環境としてタグ付けされました。これにより、DB インスタンスの識別と、このリソースに関連するコストの管理が容易になります。

DB インスタンスのストレージ容量を増やして、データニーズの増加に対応する

次に、追加のデータに対応するために、MySQL DB インスタンスのストレージ容量を変更します。初期状態では、DB インスタンスのストレージ容量は、アプリケーションの即時のニーズを満たすように設定されています。ただし、データボリュームの増加に応じて、データベースのパフォーマンスと安定性が維持されるようにストレージ設定を調整する必要がある場合があります。このプロセスでは、DB インスタンスの割り当て済みストレージを増やします。DB インスタンスのストレージ容量の変更に関する詳細については、「Amazon RDS DB インスタンスのストレージを使用する」を参照してください。

1. Amazon RDS コンソールで、[データベース] を選択します。
2. 変更する DB インスタンスを選択します。
3. [Modify] (変更) を選択します。
4. ストレージで、ストレージ割り当てを増やします。変更後のストレージの値は、現在の値よりも大きくなければなりません。

5. [続行] をクリックしてください。
6. [変更のスケジューリング] では、[すぐに適用] を選択して DB インスタンスにストレージの変更をすぐに適用するか、[次回のメンテナンス時間帯に適用] を選択して、次のメンテナンス時間中に変更を適用できます。
7. 設定が適切な場合は、[DB インスタンスを変更] を選択します。

これで DB インスタンスのストレージ容量が増加しました。これにより、より大きなデータボリュームを効果的に処理でき、アプリケーションのデータニーズの増加に応じてパフォーマンスと安定性が維持されます。

リードレプリカを作成して DB インスタンスの耐障害性と可用性を強化する

MySQL DB インスタンスのリードレプリカを作成します。リードレプリカは、DB インスタンスの耐障害性と可用性を強化します。プライマリ DB インスタンスの読み込みトラフィックを減らすには、DB インスタンスのリードレプリカを作成します。これにより、クエリがリードレプリカにルーティングされ、負荷の分散とデータベース全体のパフォーマンスの向上に役立ちます。DB インスタンスのリードレプリカの詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

MySQL DB インスタンスがレプリケーションのソースとして機能するには、ソース DB インスタンスで自動バックアップを有効にする必要があります。これを行うには、バックアップ保持期間の値を 0 以外の値に設定します。MySQL リードレプリカの詳細については、「MySQL リードレプリカの使用」を参照してください。

1. Amazon RDS コンソールで、[データベース] を選択します。
2. リードレプリカのソースとして使用する DB インスタンスを選択します。
3. [アクション] で [リードレプリカを作成] を選択します。
4. [DB インスタンス識別子] に、リードレプリカの名前をすべて小文字で入力します。
5. インスタンス設定を選択します。リードレプリカでもソース DB インスタンスと同等以上の DB インスタンスクラスとストレージタイプを使用することをお勧めします。
6. AWS リージョン では、リードレプリカの作成先のリージョンを指定します。
7. デフォルト設定のままにするか、必要に応じて変更します。
8. [Create read replica] を選択します。

リードレプリカは、RDS コンソールの [データベース] ページのソース DB インスタンスの下に表示されます。ロール列にレプリカが表示されます。

タグを更新して DB インスタンスを本番環境として分類する

DB インスタンスを開発段階から本番環境に移行する準備ができたら、移行を反映してタグを更新することが重要です。DB インスタンスを運用戦略とモニタリング戦略に合わせるため、DB インスタンスが本番環境の一部になったことを示すように初期タグを更新します。これにより、データベースの可視性と管理が向上します。

1. Amazon RDS コンソールで、[データベース] を選択します。
2. 更新する DB インスタンスを選択します。
3. [詳細] セクションで、[タグ] セクションまでスクロールします。
4. [タグを管理] を選択します。
5. 開発環境を示している初期タグを削除します。
6. [新しいタグを追加] を選択します。
7. [タグキー] と [値] に新しい値を入力します。例えば、タグキーとして environment を使用し、値を prod に設定することで、その DB インスタンスが本番環境の一部であることを指定できます。
8. [新しいタグを追加]、[変更を保存] の順に選択します。

DB インスタンスのタグが更新され、データベースが本番環境に移行されたことを示します。

追加コストが発生しないように、不要になった DB インスタンスを削除する

このチュートリアルを終了する前に、リソースの管理を行うことが重要です。不要になったリソースがある場合は、追加コストが発生しないように削除し、クラウド環境を最適化する必要があります。

1. Amazon RDS コンソールで、[データベース] を選択します。
2. 削除する DB インスタンスを選択します。
3. [アクション] で、[削除] を選択します。DB インスタンスを削除すると、そのすべてのコンテンツと関連リソースを含むインスタンスが完全に削除されます。
4. [削除] を選択して、DB インスタンスの削除を確定します。

または、DB インスタンスを将来使用するために維持することを選択した場合でも、本番環境の一部として引き続き管理できます。これには、包括的なテストと検証を容易にするために、同期された開

発環境を維持することが含まれます。詳細については、「次のステップ: 開発インスタンスを本番環境と同期して、環境間で一貫性を保つ」を参照してください。

次のステップ: 開発インスタンスを本番環境と同期して、環境間で一貫性を保つ

開発環境を作成する

本番環境を管理するには、包括的なテストと検証のために、同期された開発環境を維持することが重要です。新しい開発環境を作成するには、まず現在の本番環境 DB インスタンスの DB スナップショットを作成します。DB スナップショットは、ストレージボリュームスナップショットを作成して DB インスタンス全体をキャプチャします。Amazon RDS コンソールで DB スナップショットを作成する方法については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

本番環境の DB スナップショットを作成したら、DB スナップショットを復元して、開発環境用の新しい DB インスタンスを作成します。復元された DB インスタンスは、デフォルトの DB パラメータグループとオプショングループに自動的に関連付けられます。ただし、カスタムパラメータグループとオプショングループは、復元中に指定することで適用できます。DB スナップショットの復元手順については、「チュートリアル: DB スナップショットからの Amazon RDS DB インスタンスの復元」を参照してください。

最後に、Amazon RDS タグを更新して、新しい DB インスタンスを新しい開発環境として指定します。この変更を反映するために Amazon RDS タグを更新する方法については、前のセクション「タグを更新して DB インスタンスを本番環境として分類する」を参照してください。

これで、本番環境のデータベース設定をミラーリングする新しい開発環境ができました。

開発環境と本番環境を同期する

新しい開発環境が確立されたら、本番環境で発生した変更と同期を保つ必要があります。これにより、開発環境に本番環境の現在の状態が正確に反映されます。これは、効果的なテスト、検証、トラブルシューティングに不可欠です。Amazon RDS には、開発環境を本番環境の最新の状態に保つさまざまな方法が用意されています。これらのオプションの詳細については、「Orchestrating database refreshes for Amazon RDS and Amazon Aurora」を参照してください。

開発環境と本番環境を同期する主な方法の 1 つは、DB スナップショットを作成して復元することです。DB スナップショットを使用すると、スナップショットの作成時の本番環境のデータベース設定を反映する開発環境を作成できます。DB スナップショットの詳細については、「手動バックアップの管理」を参照してください。DB インスタンスの復元の詳細については、「DB インスタンスへの復元」を参照してください。

DB スナップショットは、以下のユースケースで特に有用です。

- 開発環境の初期設定: DB スナップショットは、スナップショット作成時の本番環境の正確な状態を反映する一貫したベースラインを提供するため、テスト用の初期開発環境を作成する際に役立ちます。
- 高トラフィックアプリケーション: 継続的な運用が重要な本番環境では、スナップショットにマルチ AZ 配置を使用すると、プライマリデータベースの I/O 停止を回避し、中断のないパフォーマンスと高可用性を確保できます。
- 異なる RDS アカウント間でのデータ共有: DB スナップショットを異なる AWS アカウント 間で共有できるため、アカウントまたはリージョン間でのデータ転送が容易になります。これは、さまざまな目的でデータ共有が必要となる共同プロジェクトやシナリオで役立ちます。詳細については、「Amazon RDS の DB スナップショットを共有する」を参照してください。

このチュートリアルでは、DB インスタンスのライフサイクル全体にわたる管理に不可欠なタスクについて学びました。DB インスタンスの作成、Amazon RDS タグの追加と更新、ストレージの拡張、リードレプリカの作成方法を学習しました。また、これらの基本的な操作を基にして、本番環境を効果的に管理する方法についても学びました。これには、テスト用の開発環境を確立し、一貫性を保つために本番環境と同期することが含まれます。これらのタスクを行うことで、データベースインフラストラクチャの耐障害性とスケーラビリティを維持し、Amazon RDS 環境を効率的に運用できます。

Amazon RDS でのマルチ AZ 配置の設定と管理

マルチ AZ デプロイには、1 つのスタンバイ DB インスタンスまたは 2 つのスタンバイ DB インスタンスを持つことができます デプロイにスタンバイ DB インスタンスが 1 つある場合は、マルチ AZ DB インスタンスのデプロイと呼ばれます。マルチ AZ DB インスタンスのデプロイには、フェイルオーバーサポートを提供するスタンバイ DB インスタンスが 1 つありますが、読み取りトラフィックは処理されません。デプロイに 2 つのスタンバイ DB インスタンスが含まれている場合は、マルチ AZ DB クラスターデプロイ。マルチ AZ DB クラスターデプロイには、フェイルオーバーサポートを提供し、読み取りトラフィックを処理できるスタンバイ DB インスタンスがあります。

AWS マネジメントコンソール を使用して、マルチ AZ 配置がマルチ AZ DB インスタンス配置であるか、マルチ AZ DB クラスター配置であるかを特定できます。ナビゲーションペインで [Databases] (データベース) を選択し、DB 識別子を選択します。

- マルチ AZ DB インスタンス配置には、次の特性があります。
  - DB インスタンスの行は 1 行のみ。
  - [Role] (ロール) の値は [Instance] (インスタンス) または [Primary] (プライマリ)。
  - [Multi-AZ] (マルチ AZ) の値は [Yes] (はい)。
- マルチ AZ DB クラスター配置には、次の特性があります。
  - クラスターレベルの行があり、その下に 3 つの DB インスタンスの行がある。
  - クラスターレベルの行の [Role] (ロール) の値は [Multi-AZ DB cluster] (マルチ AZ DB クラスター)。
  - 各インスタンスレベルの行の [Role] (ロール) の値は [Writer instance] (ライターインスタンス) または [Reader instance] (リーダーインスタンス)。
  - 各インスタンスレベルの行の [Multi-AZ] (マルチ AZ) の値は [3 Zones] (3 ゾーン)。

トピック

- Amazon RDS のマルチ AZ DB インスタンスデプロイ
- Amazon RDS のマルチ AZ DB クラスターデプロイ

さらに、以下のトピックは DB インスタンスとマルチ AZ DB クラスターの両方に適用されます。

- the section called “RDS リソースのタグ付け”
- the section called “Amazon RDS の ARN”

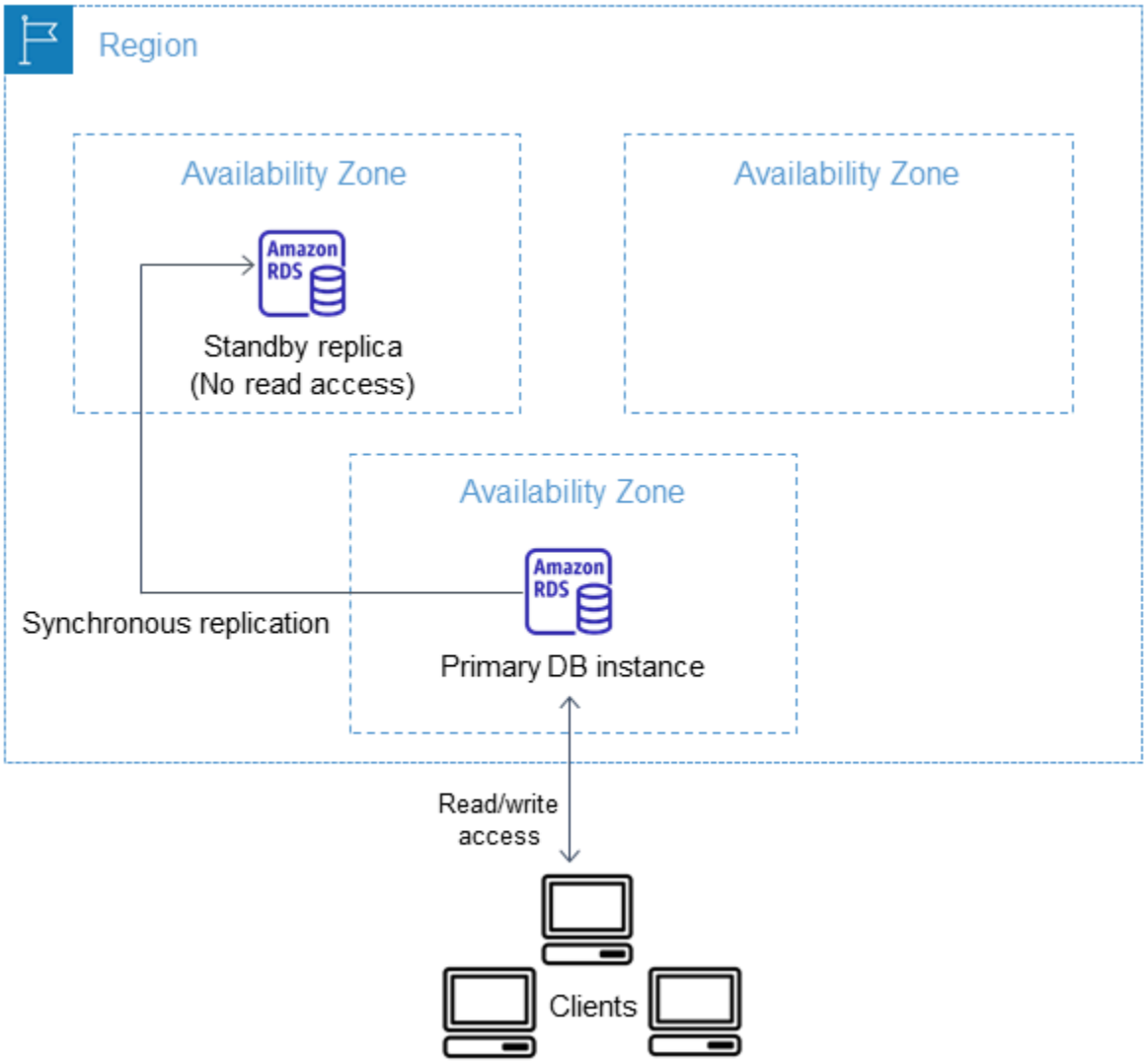
- the section called “ストレージの使用”
- the section called “DB インスタンスのメンテナンス”
- the section called “エンジンバージョンのアップグレード”

Amazon RDS のマルチ AZ DB インスタンスデプロイ

Amazon RDS は、単一のスタンバイ DB インスタンスでマルチ AZ デプロイを使用して DB インスタンスの高可用性およびフェイルオーバーサポートを提供します。このタイプのデプロイは、マルチ AZ DB インスタンスのデプロイと呼ばれます。Amazon RDS は複数の異なるテクノロジーを使用してフェイルオーバーサポートを提供します。MariaDB、MySQL、Oracle、PostgreSQL、および RDS Custom for SQL Server DB インスタンスのマルチ AZ デプロイでは、Amazon のフェイルオーバーテクノロジーが使用されます。Microsoft SQL Server DB インスタンスでは、SQL Server データベースのミラーリング (DBM) または Always On 可用性グループ (AGs) が使用されます。マルチ AZ の SQL Server バージョンのサポートについては、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。マルチ AZ の RDS Custom for SQL Server の使用については、「RDS Custom for SQL Server のマルチ AZ 配置の管理」を参照してください。

マルチ AZ DB インスタンスのデプロイでは、Amazon RDS は、異なるアベイラビリティーゾーンで同期スタンバイレプリカを自動的にプロビジョンおよび維持します。プライマリ DB インスタンスは、アベイラビリティーゾーン間でスタンバイレプリカに同期的に複製され、データの冗長性を提供し、システムバックアップ中の遅延スパイクを最小限に抑えます。高可用性を備えた DB インスタンスを実行すると、計画されたシステムメンテナンス中の可用性が向上する可能性があります。また、DB インスタンスの障害とアベイラビリティーゾーンの中断からデータベースを保護することを助けることもできます。アベイラビリティーゾーンの詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

> **Note**
>
> 高可用性のオプションは、読み取り専用シナリオ向けのスケーリングソリューションではありません。スタンバイレプリカを使用してリードトラフィックを処理することはできません。読み取り専用トラフィックを処理するには、代わりにマルチ AZ DB クラスターまたはリードレプリカを使用します。マルチ AZ DB クラスターの詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。リードレプリカの詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。



RDS コンソールを使用すると、DB インスタンスを作成する際にマルチ AZ を指定するだけでマルチ AZ DB デプロイを作成できます。コンソールを使用し、DB インスタンスを変更し、マルチ AZ オプションを指定して、マルチ AZ DB インスタンスを変更することで、既存の DB インスタンスをマルチ AZ DBインスタンスのデプロイに変換できます。AWS CLIまたは Amazon RDS API を使用して、マルチ AZ DB インスタンスのデプロイを指定することもできます。create-db-instance または modify-db-instance CLI コマンドを使用するか、CreateDBInstance または ModifyDBInstance API オペレーションを使用します。

RDS コンソールには、スタンバイレプリカ (セカンダリ AZ と呼ばれます) のアベイラビリティーゾーンが表示されます。また、describe-db-instances CLI コマンドまたは DescribeDBInstances API オペレーションを使用してセカンダリ AZ を見つけることもできます。

マルチAZ DB デプロイを使用する DBインスタンスは、シングル AZ のデプロイと比較して書き込みとコミットの待ち時間が長くなる可能性があります。これは、同期データレプリケーションが発生し

ているために起こる可能性があります。AWS はアベイラビリティーゾーン間でのネットワーク接続レイテンシーが低くなるように設計されていますが、配置がスタンバイレプリカにフェイルオーバーした場合はレイテンシーに変化が見られる可能性があります。本番ワークロードの場合、高速で一貫したパフォーマンスを実現するために、プロビジョンドIOPS (1 秒あたりの入出力操作) を使用することをお勧めします。DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

Amazon RDS の DB インスタンスをマルチ AZ 配置に変換する

DB インスタンスをマルチ AZ 配置に変更すると、別のアベイラビリティーゾーンにスタンバイインスタンスを追加することで、可用性が向上します。このプロセスではダウンタイムを最小限に抑え、ストレージとパフォーマンスへの影響について慎重に計画する必要があります。この変更により耐障害性が向上し、障害発生時の復旧時間が短縮されるため、高可用性環境に最適です。

シングル AZ デプロイの DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更すると、Amazon RDS で以下のアクションが実行されます。

1. プライマリ DB インスタンスの Amazon Elastic Block Store (EBS) ボリュームのスナップショットを作成します。
2. スナップショットからスタンバイレプリカ用の新しいボリュームを作成します。これらのボリュームはバックグラウンドで初期化され、データが完全に初期化された後に最大のボリュームパフォーマンスが得られます。
3. プライマリレプリカとスタンバイレプリカのボリューム間の同期ブロックレベルレプリケーションを有効にします。

変換中、DB インスタンスのステータスは [利用可能] から [変更中] に移行し、新しいスタンバイのボリュームがバックグラウンドでハイドレートし続ける間は、[利用可能] に戻ります。新しいスタンバイでボリュームの初期化をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。初期化中、`StorageOperationStatus` は `Initializing` に設定され、初期化が完了すると両方のフィールドは存在しなくなります。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

> **Important**
>
> シングル AZ からマルチ AZ への変換中にスナップショットからスタンバイ DB インスタンスを作成すると、ダウンタイムを回避できますが、特に書き込みに敏感なワークロードで

はパフォーマンスに影響する可能性があります。同期レプリケーションは I/O レイテンシーを増加し、データベースのパフォーマンスに影響を与える可能性があります。ベストプラクティスとして、本番 DB インスタンスをマルチ AZ DB インスタンスに変換することは避けてください。
代わりに、リードレプリカを作成し、バックアップを有効にしてマルチ AZ に変換し、そのボリュームにデータをロードしてから、プライマリ DB インスタンスに昇格します。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更するには 2 つの方法があります。

トピック

- RDS コンソールを使用して、マルチ AZ DB インスタンスのデプロイに変換する
- DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更する

RDS コンソールを使用して、マルチ AZ DB インスタンスのデプロイに変換する

RDS コンソールを使用して、DB インスタンスをマルチ AZ DB インスタンスのデプロイに変換できます。

コンソールは、変換を完了するためにのみ使用できます。AWS CLI または RDS API を使用するには、DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更する の手順に従います。

RDS コンソールを使用して、マルチ AZ DB インスタンスのデプロイに変換するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Actions] (アクション) から、[Convert to Multi-AZ deployment] (マルチ AZ 配置に変換) を選択します。
4. 変更をすぐに適用するには、確認ページで [Apply Immediately] (すぐに適用) を選択します。このオプションを選択してもダウンタイムは発生しませんが、パフォーマンスに影響する可能性があります。または、次のメンテナンスウィンドウの間に更新を適用することもできます。詳細については、「スケジュール変更設定の使用」を参照してください。
5. [Convert to Multi-AZ] (マルチ AZ に変換) を選択します。

DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更する

次の方法で、DB インスタンスをマルチ AZ DB インスタンスのデプロイに変更できます。

- RDS コンソールを使用して DB インスタンスを変更し、[Multi-AZ deployment] (マルチ AZ 配置) を [Yes] (はい) に設定します。
- AWS CLI を使用して modify-db-instance コマンドを呼び出し、`--multi-az` オプションを設定します。
- RDS API を使用して、ModifyDBInstance オペレーションを呼び出して、`MultiAZ` パラメータを `true` に設定します。

DB インスタンスの変更については、「Amazon RDS DB インスタンスを変更する」を参照してください。変更が完了した後、Amazon RDS は、プロセスが完了したことを示すイベント (RDS-EVENT-0025) をトリガーします。Amazon RDS イベントをモニタリングできます。 イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。

Amazon RDS 用のマルチ AZ DB インスタンスのフェイルオーバー

マルチ AZ DB インスタンスの計画的または計画外の停止がインフラストラクチャの欠陥に起因する場合、Amazon RDS は別のアベイラビリティーゾーンのスタンバイレプリカに自動的に切り替わります。

フェイルオーバーが完了するまでにかかる時間は、プライマリ DB インスタンスが使用できなくなったときのデータベースアクティビティおよびその他の条件によって異なります。フェイルオーバー時間は通常 60 ~ 120 秒です。ただし、大規模なトランザクションや長期にわたる復旧プロセスによって、フェイルオーバー時間が増加する場合があります。フェイルオーバーが完了してから、新しいアベイラビリティーゾーンが RDS コンソールに反映されるまでさらに時間がかかる場合があります。

> **Note**
>
> マルチ AZ DB インスタンスを再起動するときに手動でフェイルオーバーを強制的に実行することができます。詳細については、「 DB インスタンスの再起動」を参照してください。

Amazon RDS はフェイルオーバーを自動的に処理するため、管理者が操作しなくても可能な限りすみやかにデータベース操作を再開できます。次の表に記載されている条件のいずれかが発生した場合、プライマリ DB インスタンスがスタンバイレプリカに自動的に切り替わります。これらのフェイルオーバーの理由は、イベントログで確認できます。

フェイルオーバーした理由	説明
RDS データベースインスタンスの基盤となるオペレーティングシステムには、オフライン操作でパッチが適用されています。	OS パッチまたはセキュリティ更新プログラムのメンテナンス期間中に、フェイルオーバーがトリガーされました。 詳細については、「DB インスタンスのメンテナンス」を参照してください。
RDS マルチ AZ インスタンスのプライマリホストが異常です。	マルチ AZ DB インスタンスのデプロイは、障害のあるプライマリ DB インスタンスを検出し、フェイルオーバーしました。
RDS マルチ AZ インスタンスのプライマリホストで、ネットワーク接続が切断されたため、到達できません。	RDS モニタリングは、プライマリ DB インスタンスへのネットワーク到達可能性による障害を検出し、フェイルオーバーをトリガーしました。
RDS インスタンスはお客様によって変更されました。	RDS DB インスタンスの変更により、フェイルオーバーがトリガーされました。 詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
RDS マルチ AZ プライマリインスタンスはビジーで応答しません。	プライマリ DB インスタンスが応答しません。次のことを行うことをお勧めします。 • イベントログと CloudWatch Logs を調べて、CPU、メモリ、またはスワップ領域の使用量が多すぎるかどうかを調べます。詳細については、「Amazon RDS イベント通知の操作」および「Amazon RDS イベントでトリガーするルールの作成」を参照してください。 • ワークロードを評価して、適切な DB インスタンスクラスを使用しているかどうかを判断

フェイルオーバーした理由	説明
	します。詳細については、「DB インスタンスクラス」を参照してください。 • オペレーティングシステムのリアルタイムメトリクスには、拡張モニタリングを使用します。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。 • Performance Insights を使用して、DB インスタンスのパフォーマンスに影響を与えるすべての問題を分析することができます。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。 これらの推奨事項の詳細については、Amazon RDS のモニタリングツール および Amazon RDS のベストプラクティス を参照してください。
RDS マルチ AZ インスタンスのプライマリホストの基盤となるストレージボリュームでエラーが発生しました。	マルチ AZ DB インスタンスのデプロイは、プライマリ DB インスタンスでストレージの問題を検出し、フェイルオーバーしました。
ユーザーが DB インスタンスのフェイルオーバーをリクエストしました。	DB インスタンスを再起動し、[Reboot with failover (フェイルオーバーで再起動)] を選択しました。 (詳しくは、「DB インスタンスの再起動」を参照してください。)

マルチ AZ DB インスタンスがフェイルオーバーされたかどうかを判断するには、次を実行します。

- フェイルオーバーがスタートされたことを電子メールまたはSMSで通知するようにDBイベントサブスクリプションを設定します。 イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。
- RDS コンソールまたは API オペレーションを使用して DB イベントを表示します。
- RDS コンソールまたは API オペレーションを使用して、マルチ AZ DB インスタンスのデプロイの現在の状態を表示します。

フェイルオーバーへの応答、復旧時間の短縮、Amazon RDS のその他のベストプラクティスについては、「Amazon RDS のベストプラクティス」を参照してください。

DNS 名参照用の JVM TTL の設定

フェイルオーバーメカニズムでは、スタンバイ DB インスタンスをポイントするように DB インスタンスのドメインネームシステム (DNS) レコードが自動的に変更されます。したがって、DB インスタンスへの既存の接続の再確立が必要になります。Java 仮想マシン (JVM) 環境では、Java DNS キャッシュ機構がどのように機能するかによって、JVM 設定の再構成が必要になる場合があります。

JVM は DNS 名参照をキャッシュします。JVM がホスト名を IP アドレスに変換するとき、time-to-live (TTL) と呼ばれる指定期間 IP アドレスをキャッシュします。

AWS リソースは、ときどき変更される DNS 名を使用するため、60 秒を超えない TTL 値で JVM を設定することをお勧めします。こうすることにより、リソースの IP アドレスが変更されたときに、アプリケーションは DNS に対して再度クエリを実行することで、リソースの新しい IP アドレスを取得して使用できます。

一部の Java 設定では JVM のデフォルトの TTL が設定されるため、JVM が再起動されるまで、DNS エントリが更新されることはありません。したがって、アプリケーションがまだ実行中に AWS リソースの IP アドレスが変更された場合、JVM を手動で再起動し、キャッシュされた IP 情報が更新されるまで、そのリソースを使用することはできません。この場合、キャッシュされた IP 情報が定期的に更新されるように JVM の TTL を設定することがきわめて重要です。

JVM のデフォルト TTL は、`networkaddress.cache.ttl` プロパティ値を取得することで取得できます。

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> デフォルト TTL は、JVM のバージョンと、セキュリティマネージャーがインストールされているかどうかに応じて変わります。多くの JVM はデフォルト TTL を 60 秒以下にしています。このような JVM を使用しており、セキュリティマネージャーを使用していない場合、このトピックの残り部分は無視してかまいません。Oracle のセキュリティマネージャーの詳細については、Oracle ドキュメントの 「The Security Manager」を参照してください。

JVM の TTL を変更するには、`networkaddress.cache.ttl` プロパティ値を設定します。ニーズに応じて、次の方法のいずれかを使用します。

- JVM を使用するすべてのアプリケーションのプロパティ値をグローバルに設定するには、`networkaddress.cache.ttl` ファイルで `$JAVA_HOME/jre/lib/security/java.security` を設定します。

```
networkaddress.cache.ttl=60
```

- アプリケーションに対してのみプロパティをローカルに設定するには、ネットワーク接続を確立する前に、アプリケーションの初期化コードで `networkaddress.cache.ttl` を設定します。

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

追加のストレージボリュームによるマルチ AZ フェイルオーバー

マルチ AZ 配置は、追加のストレージボリュームを持つ DB インスタンスをサポートします。フェイルオーバー中、RDS は DB インスタンスにアタッチされた追加のストレージボリュームを使用して、スタンバイインスタンスに自動的にフェイルオーバーします。このプロセスにより、データ整合性と可用性が確保されます。

追加のストレージボリュームを持つ DB インスタンスのマルチ AZ 配置を設定すると、Amazon RDS はすべてのボリュームを別のアベイラビリティーゾーンのスタンバイインスタンスに自動的にレプリケートします。レプリケートされたストレージには以下が含まれます。

- プライマリストレージボリューム
- DB インスタンスにアタッチされたすべての追加のストレージボリューム

フェイルオーバー中、Amazon RDS はスタンバイインスタンスを昇格させ、すべてのストレージボリュームが利用可能で一貫性があることを確認します。フェイルオーバーは、ボリューム名、ストレージタイプ、パフォーマンス特性など、同じストレージ設定を維持します。

フェイルオーバーが成功したら、ストレージ設定の詳細を表示して、すべてのストレージボリュームが正しくアタッチされ、アクセス可能であることを確認します。詳細については、「DB インスタンスのストレージボリュームの詳細の表示」を参照してください。

追加のストレージボリュームを持つ DB インスタンスのフェイルオーバー時間は、プライマリストレージのみを持つ DB インスタンスとほぼ同じです。

Amazon RDS のマルチ AZ DB クラスターデプロイ

あるマルチ AZ DB クラスターデプロイは、2 つの読み取り可能なリードレプリカ DB インスタンスを持つ Amazon RDS の準同期の高可用性のデプロイモードです。マルチ AZ DB クラスターには、同じ AWS リージョン に 3 つの別々のアベイラビリティーゾーンに 1 つのライター DB インスタンスと 2 つのリーダー DB インスタンスがあります。マルチ AZ DB クラスターは、マルチ AZ DB インスタンスの配置と比較して、高可用性、読み取りワークロードの容量の増加、および書き込みレイテンシーの低減を提供します。

ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする の説明に従って、オンプレミスデータベースからマルチ AZ DB クラスターにデータをインポートできます。

マルチ AZ DB クラスターのリザーブド DB インスタンスを購入できます。詳細については、「マルチ AZ DB クラスターのリザーブド DB インスタンス」を参照してください。

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。マルチ AZ DB クラスターを使用した Amazon RDS のバージョンとリージョンの可用性の詳細ついては、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。

トピック

- マルチ AZ DB クラスターで利用できるインスタンスクラス
- マルチ AZ DB クラスターアーキテクチャ
- マルチ AZ DB クラスターのパラメータグループ
- マルチ AZ DB クラスターを備えた RDS Proxy
- レプリカの遅延とマルチ AZ DB クラスター
- マルチ AZ DB クラスターのスナップショット
- Amazon RDS 用のマルチ AZ DB クラスターの作成
- Amazon RDS のマルチ AZ DB クラスターへの接続
- AWS コンピューティングリソースと Amazon RDS のマルチ AZ DB クラスターを自動的に接続する
- Amazon RDS のマルチ AZ DB クラスターの変更
- Amazon RDS のマルチ AZ DB クラスターのエンジンバージョンのアップグレード
- Amazon RDS のマルチ AZ DB クラスターの名前の変更
- Amazon RDS のマルチ AZ DB クラスターとリーダー DB インスタンスの再起動

- Amazon RDS 用のマルチ AZ DB クラスターのフェイルオーバー
- Amazon RDS のマルチ AZ DB クラスターを使用した PostgreSQL 論理レプリケーションの設定
- Amazon RDS のマルチ AZ DB クラスターのリードレプリカの操作
- Amazon RDS のマルチ AZ DB クラスターからの外部レプリケーションの設定
- Amazon RDS 用のマルチ AZ DB クラスターの削除
- Amazon RDS のマルチ AZ DB クラスターの制限事項

> **Important**
>
> マルチ AZ DB クラスターは Aurora DB クラスターと同じではありません。Aurora DB クラスターの情報については、Amazon Aurora ユーザーガイドを参照してください。

マルチ AZ DB クラスターで利用できるインスタンスクラス

マルチ AZ DB クラスター配置は、次の DB インスタンスクラスでサポートされています。`db.c6gd`、`db.m5d`、`db.m6gd`、`db.m6id`、`db.m6idn`、`db.m8gd`、`db.r5d`、`db.r6gd`、`db.r6id`、

> **Note**
>
> `medium` インスタンスサイズをサポートするインスタンスクラスは、c6gd インスタンスクラスのみです。

DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

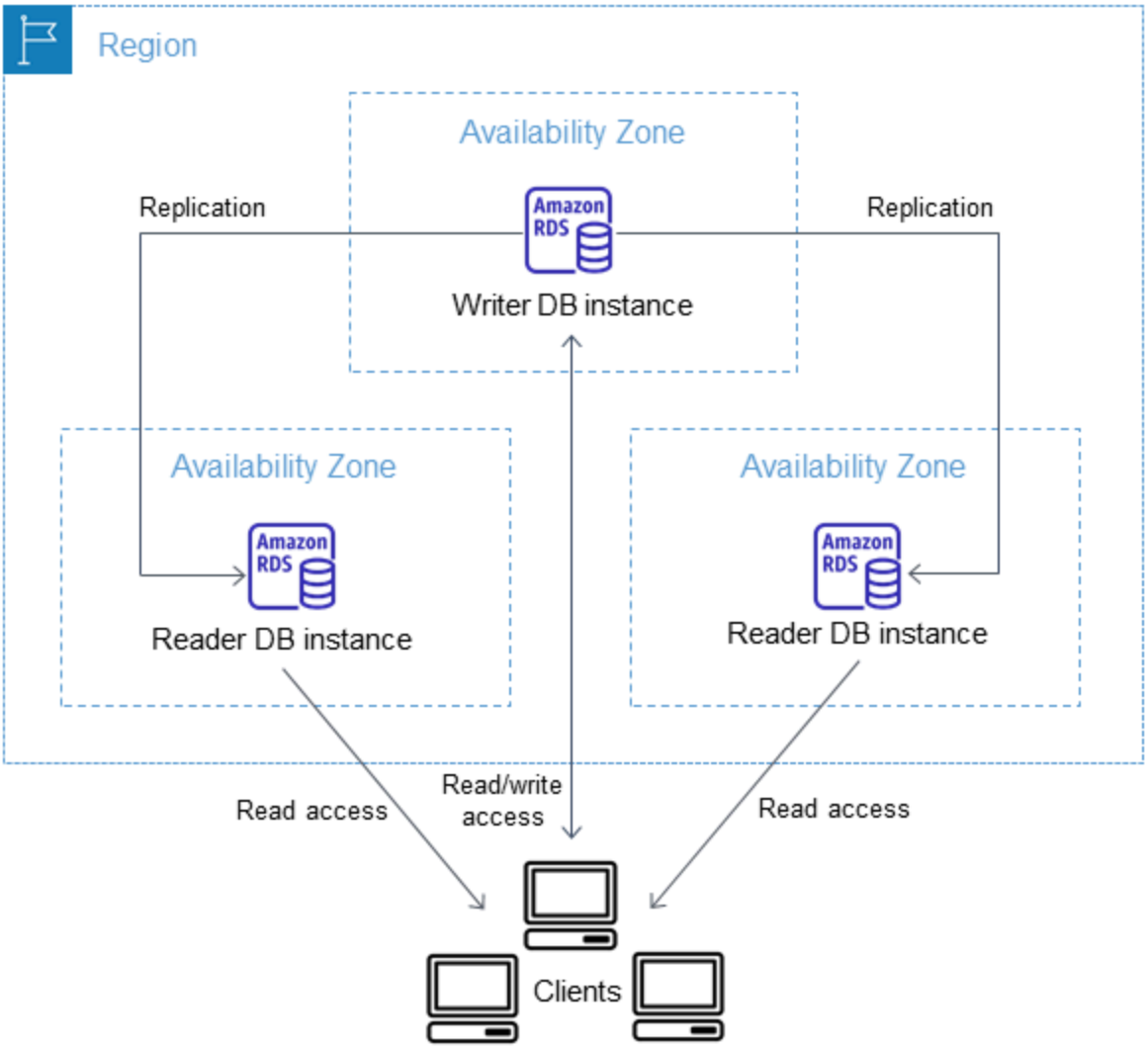
マルチ AZ DB クラスターアーキテクチャ

マルチ AZ DB クラスターでは、Amazon RDS は DB エンジンのネイティブレプリケーション機能を使用して、ライター DB インスタンスから両方のリーダー DB インスタンスにデータを複製します。ライター DB インスタンスに変更が加えられると、各リーダー DB インスタンスに送信されます。

マルチ AZ DB クラスター配置では、準同期レプリケーションを使用します。変更をコミットするには、少なくとも 1 つのリーダー DB インスタンスからの承認が必要です。イベントが完全に実行され、すべてのレプリカでコミットされたことを確認する必要はありません。

リーダー DB インスタンスは自動フェイルオーバーターゲットとして機能し、読み取りトラフィックを処理してアプリケーションの読み取りスループットを向上させます。 ライター DB インスタンスで機能停止が発生した場合、RDS はリーダー DB インスタンスのうち 1 つへのフェイルオーバーを管理します。RDS は、最新の変更記録があるリーダー DB インスタンスを基にこれを実行します。

次の図は、マルチ AZ DB クラスターを示しています。



マルチ AZ DB クラスターは、マルチ AZ DB インスタンスの配置と比較して、通常、書き込みレイテンシーが少なくなります。また、リーダー DB インスタンスで読み取り専用ワークロードを実行することもできます。RDS コンソールには、ライター DB インスタンスのアベイラビリティーゾーンとリーダー DB インスタンスのアベイラビリティーゾーンが表示されます。また、 describe-db-clustersCLI コマンドまたはこの情報を見つけるためのDescribeDBClustersAPI オペレーションを使用することもできます。

> **Important**
>
> RDS for MySQL マルチ AZ DB クラスターのレプリケーションエラーを防ぐため、すべてのテーブルにプライマリキーを設定することを強くお勧めします。

マルチ AZ DB クラスターのパラメータグループ

マルチ AZ DBクラスターでは、DB クラスターパラメータグループは、マルチ AZ DB クラスター内のすべての DB インスタンスに適用されるエンジン構成値のコンテナーとして機能します。

マルチ AZ DB クラスターでは、DB パラメータグループは、DB エンジンおよび DB エンジンバージョンのデフォルトの DB パラメータグループに設定されています。DB クラスターパラメータグループの設定は、クラスター内のすべての DB インスタンスに使用されます。

パラメータグループの詳細については、「the section called “DB クラスターのパラメータグループ”」を参照してください。

マルチ AZ DB クラスターを備えた RDS Proxy

Amazon RDS Proxy を使用すると、マルチ AZ DB クラスターに対してプロキシを作成できます。RDS プロキシを使用すると、アプリケーションではデータベース接続をプールおよび共有し、そのスケール能力を向上させることができます。各プロキシは、接続の多重化も実行します。これは接続の再利用とも呼ばれます。多重化により、RDS Proxy は 1 つの基となるデータベース接続を使用して 1 つのトランザクションのすべてのオペレーションを実行します。RDS Proxy を使用すると、マルチ AZ DB クラスターのマイナーバージョンアップグレードのダウンタイムを 1 秒以下に短縮することもできます。RDS Prox のメリットの詳細については、「Amazon RDS Proxy 」を参照してください。

マルチ AZ DB クラスターのプロキシを設定するには、クラスター作成時に [RDS Proxy の作成] を選択します。RDS Proxy エンドポイントの作成と管理の手順については、「the section called “RDS Proxy エンドポイントの操作”」を参照してください。

レプリカの遅延とマルチ AZ DB クラスター

レプリカの遅延とは、ライター DB インスタンスの最新のトランザクションと、リーダー DB インスタンスで最後に適用されたトランザクションとの時間の差です。Amazon CloudWatch メトリクス `ReplicaLag` は、この時間の差を表します。CloudWatch のメトリクスの詳細については、

「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

マルチ AZ DB クラスターでは高い書き込みパフォーマンスが得られますが、エンジンベースのレプリケーションの性質上、レプリカの遅延が発生する可能性があります。フェイルオーバーでは、新しいライター DB インスタンスに昇格する前にまずレプリカの遅延を解決する必要があるため、レプリカの遅延のモニタリングおよび管理は考慮事項です。

RDS for MySQL マルチ AZ DB クラスターの場合、フェイルオーバーの時間は残りの両方のリーダー DB インスタンスのレプリカの遅延によって異なります。どちらのリーダー DB インスタンスについても、いずれかが新しいライター DB インスタンスに昇格する前に、未適用のトランザクションを適用する必要があります。

RDS for PostgreSQL マルチ AZ DB クラスターの場合、フェイルオーバーの時間は残りの 2 つのリーダー DB インスタンスの最も低いレプリカの遅延によって異なります。レプリカの遅延が最も低いリーダー DB インスタンスについては、新しいライター DB インスタンスに昇格する前に、未適用のトランザクションを適用する必要があります。

レプリカの遅延が設定された時間を超過した際に CloudWatch アラームを作成する方法のチュートリアルについては、「チュートリアル: Amazon RDS 向けマルチ AZ DB クラスターレプリカラグ用の Amazon CloudWatch アラームの作成」を参照してください。

レプリカの遅延の一般的な原因

一般に、レプリカの遅延は書き込みワークロードが高すぎてリーダー DB インスタンスでトランザクションを効率的に適用できない場合に発生します。異なるワークロードでは、一時的に、または継続的にレプリカの遅延が発生する可能性があります。一般的な原因の例をいくつか次に示します。

- 書き込みの同時実行性が高いか、ライター DB インスタンスで大量のバッチ更新が行われるため、リーダー DB インスタンスの適用プロセスが遅れている。
- 1 つ以上のリーダー DB インスタンスでリソースを使用しており、読み取りワークロード負荷が高い。低速または大規模なクエリを実行すると、適用プロセスに影響し、レプリカの遅延が発生する可能性があります。
- 大量のデータまたは DDL ステートメントを変更するトランザクション。データベースのコミットの順序を保持する必要があるため、レプリカの遅延が一時的に増加することがあります。

レプリカの遅延の軽減

RDS for MySQL および RDS for PostgreSQL のマルチ AZ DB クラスターでは、ライター DB インスタンスの負荷を軽減することで、レプリカの遅延を軽減できます。また、フロー制御を使用してレプリカの遅延を軽減できます。フロー制御は、ライター DB インスタンスに対する書き込みをスロットリングすることで機能します。これにより、レプリカの遅延が無制限に増加し続けることを防ぎます。書き込みスロットリングは、トランザクションの最後に遅延を追加することで実現されます。これにより、ライター DB インスタンスの書き込みスループットが低下します。フロー制御は遅延をなくすことを保証するものではありませんが、多くのワークロードで全体的な遅延を軽減するのに役立ちます。次のセクションでは、RDS for MySQL および RDS for PostgreSQL でのフロー制御の使用方法について説明します。

RDS for MySQL でのフロー制御によるレプリカの遅延の軽減

RDS for MySQL マルチ AZ DB クラスターを使用している場合、動的パラメータ `rpl_semi_sync_master_target_apply_lag` を使用することでデフォルトでフロー制御が有効になります。このパラメータは、レプリカの遅延における上限を指定します。レプリカの遅延が設定された上限に近づくと、フロー制御は指定された値より小さいレプリカの遅延を含むようにライター DB インスタンス上の書き込みトランザクションをスロットリングします。場合によっては、レプリカの遅延が指定された上限を超えることがあります。デフォルトでは、パラメータは 120 秒に設定されています。フロー制御を無効にするには、このパラメータを最大値の 86,400 秒 (1 日) に設定します。

フロー制御によって挿入される現在の遅延を表示するには、次のクエリを実行してパラメータ `Rpl_semi_sync_master_flow_control_current_delay` を表示します。

```
SHOW GLOBAL STATUS like '%flow_control%';
```

出力は以下のようになります。

```
+-----------------------------------------------+-------+
| Variable_name                                 | Value |
+-----------------------------------------------+-------+
| Rpl_semi_sync_master_flow_control_current_delay | 2010  |
+-----------------------------------------------+-------+
1 row in set (0.00 sec)
```

> **Note**
>
> 遅延はマイクロ秒単位で示されます。

RDS for MySQL マルチ AZ DB クラスターの Performance Insights を有効にすると、クエリがフロー制御によって遅延したことを示す SQL ステートメントに対応する待機イベントをモニタリングできます。フロー制御により遅延が発生すると、Performance Insights ダッシュボードで SQL ステートメントに対応する待機イベント `/wait/synch/cond/semisync/semi_sync_flow_control_delay_cond` を表示できます。これらのメトリクスを表示するには、Performance Schema が有効になっていることを確認してください。Performance Insights の詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。

RDS for PostgreSQL でのフロー制御によるレプリカの遅延の軽減

RDS for PostgreSQL のマルチ AZ DB クラスターを使用している場合、フロー制御は拡張機能としてデプロイされています。これにより、DB クラスターのすべての DB インスタンスでバックグラウンドワーカーが開始します。デフォルトでは、リーダー DB インスタンスのバックグラウンドワーカーは、現在のレプリカの遅延をライター DB インスタンスのバックグラウンドワーカーに伝えます。いずれかのリーダー DB インスタンスで遅延が 2 分を超える場合、ライター DB インスタンスのバックグラウンドワーカーは、トランザクションの最後に遅延を追加します。遅延のしきい値を制御するには、パラメータ `flow_control.target_standby_apply_lag` を使用します。

フロー制御が PostgreSQL プロセスをスロットリングすると、`pg_stat_activity` および Performance Insights の `Extension` 待機イベントに表示されます。現在追加されている遅延の量に関する詳細が関数 `get_flow_control_stats` に表示されます。

フロー制御は、短いが負荷の高い同時トランザクションを持つほとんどのオンライントランザクション処理 (OLTP) のワークロードにメリットがあります。バッチ操作などの長時間実行されるトランザクションによって遅延が発生した場合、それほど大きなメリットはありません。

フロー制御を無効にするには、`shared_preload_libraries` から拡張機能を削除するか、DB インスタンスを再起動します。

マルチ AZ DB クラスターのスナップショット

Amazon RDS は、設定されたバックアップ期間中に、マルチ AZ DB クラスターの自動バックアップを作成して保存します。RDS は DB クラスターのストレージボリュームのスナップショットを作成し、個々のインスタンスだけではなく、クラスター全体をバックアップします。

マルチ AZ DB クラスターの手動バックアップを取ることもできます。非常に長期間のバックアップの場合、スナップショットデータを Amazon S3 にエクスポートすることを検討してください。詳細については、「the section called “マルチ AZ DB クラスターのスナップショットの作成”」を参照してください。

マルチ AZ DB クラスターを指定の時点に復元し、新しいマルチ AZ DB クラスターを作成できます。手順については、「the section called “マルチ AZ DB クラスターを指定の時点の状態に復元する”」を参照してください。

または、マルチ AZ DB クラスターのスナップショットをシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイに復元することができます。手順については、「the section called “マルチ AZ DB クラスターのスナップショットから DB インスタンスへの復元”」を参照してください。

Amazon RDS 用のマルチ AZ DB クラスターの作成

マルチ AZ DB クラスターには、3 つの別々のアベイラビリティーゾーンに 1 つのライター DB インスタンスと 2 つのリーダー DB インスタンスがあります。マルチ AZ DB クラスターは、マルチ AZ 配置と比較して、高可用性、読み取りワークロードの容量の増加、および低レイテンシーを提供します。マルチ AZ DB の配置の詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。

> **Note**
>
> マルチ AZ DB クラスターは MySQL および PostgreSQL DB エンジンでのみサポートされます。

DB クラスターの前提条件

> **Important**
>
> マルチ AZ DB クラスターを作成する前に、Amazon RDS 環境のセットアップ のタスクを完了する必要があります。

マルチ AZ DB クラスターを作成する前に完了しておく必要がある前提条件を次に示します。

トピック

- DB クラスターのネットワークを設定する
- 追加の前提条件

DB クラスターのネットワークを設定する

マルチ AZ DB クラスターは、Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) でのみ作成できます。少なくとも 3 つのアベイラビリティーゾーンを持つ AWS リージョン である必要があります。DB クラスターで選択する DB サブネットグループは、少なくとも 3 つのアベイラビリティーゾーンを対象とする必要があります。この設定により、DB クラスターの各 DB インスタンスが別のアベイラビリティーゾーンに配置されます。

新しい DB クラスターと同じ VPC 内の Amazon EC2 インスタンス間の接続を設定するには、DB クラスターの作成時に設定します。同じ VPC 内の EC2 インスタンス以外のリソースから DB クラスターに接続するには、ネットワーク接続を手動で設定します。

トピック

- EC2 インスタンスとの自動ネットワーク接続を設定する
- ネットワークを手動で設定する

EC2 インスタンスとの自動ネットワーク接続を設定する

マルチ AZ DB クラスターを作成する場合は、AWS マネジメントコンソール を使用して EC2 インスタンスと新しい DB クラスター間の接続をセットアップできます。これを行うと、RDS では VPC とネットワークの設定を自動で行います。EC2 インスタンスが DB クラスターにアクセスできるように、EC2 インスタンスと同じ VPC 内に DB クラスターを作成します。

EC2 インスタンスと DB クラスターを接続するための要件は次のとおりです。

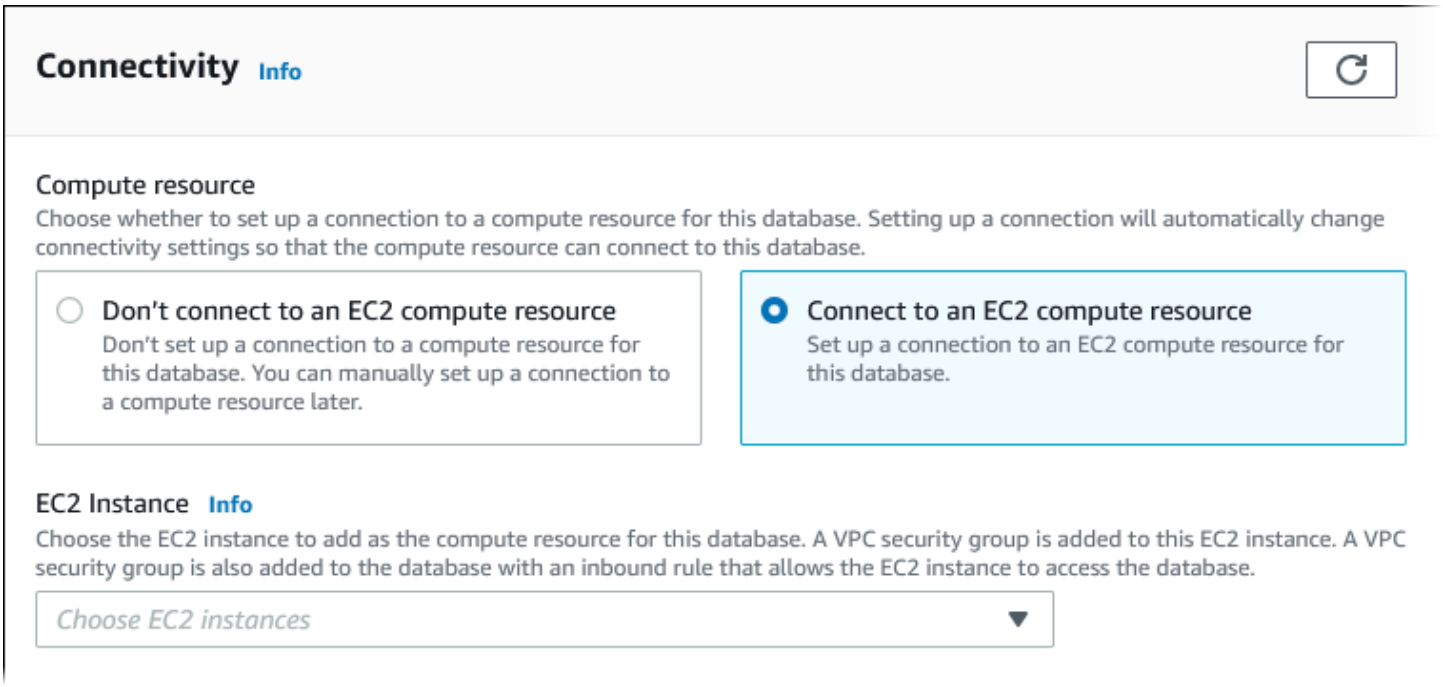
- DB クラスターを作成する前に、AWS リージョン に EC2 インスタンスが存在する必要があります。

  AWS リージョン に EC2 インスタンスが存在しない場合、コンソールには EC2 インスタンス作成用のリンクが表示されます。
- DB クラスターを作成するユーザーには、次の操作を実行する権限が必要です。
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`

- `ec2:RevokeSecurityGroupEgress`

このオプションを使用すると、プライベート DB クラスターが作成されます。DB クラスターでは、プライベートサブネットのみを持つ DB サブネットグループを使用して、VPC 内のリソースへのアクセスを制限します。

EC2 インスタンスを DB クラスターに接続するには、[Create database] (データベースの作成) ページの [Connectivity] (接続) セクションで、[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択します。



[Connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続する) を選択すると、RDS では次のオプションを自動的に設定します。[Don't connect to an EC2 compute resource] (EC2 コンピューティングリソースに接続しない) を選択して EC2 インスタンスとの接続をセットアップしない限り、これらの設定は変更できません。

コンソールオプション	自動ログ記録
仮想プライベートクラウド (VPC)。	RDS は EC2 インスタンスに関連付けられている VPC に設定します。
DB サブネットグループ	RDS では、EC2 インスタンスと同じアベイラビリティーゾーンにプライベートサブネットを持つ DB サブネットグループが必

コンソールオプション	自動ログ記録
	要です。この要件を満たす DB サブネットグループが存在する場合、RDS は既存の DB サブネットグループを使用します。デフォルトでは、このオプションは [Automatic setup] (自動セットアップ) に設定されています。 [Automatic setup] (自動セットアップ) を選択したとき、この要件を満たす DB サブネットグループがない場合、次のアクションが実行されます。RDS は 3 つのアベイラビリティーゾーンで 3 つの使用可能なプライベートサブネットを使用します。アベイラビリティーゾーンのうちの 1 つは EC2 インスタンスと同じです。プライベートサブネットがアベイラビリティーゾーンで使用できない場合、RDS はアベイラビリティーゾーンにプライベートサブネットを作成します。次に、RDS は DB サブネットグループを作成します。 プライベートサブネットが使用可能な場合、RDS はサブネットに関連付けられているルートテーブルを使用して、作成したサブネットをこのルートテーブルに追加します。プライベートサブネットが使用できない場合、RDS はインターネットゲートウェイにアクセスできないルートテーブルを作成し、作成したサブネットをルートテーブルに追加します。 RDS では、既存の DB サブネットグループを使用することもできます。既存の DB サブネットグループを使用する場合は、[Choose existing] (既存を選択) を選択します。
パブリックアクセス	RDS では [No] (いいえ) を選択して、DB クラスターがパブリックアクセス可能にならないようにします。 セキュリティ上の理由から、データベースは非公開として、インターネットからアクセスできないようにするのがベストプラクティスです。

コンソールオプション	自動ログ記録
VPC セキュリティグループ (ファイアウォール)	RDS では DB クラスターに関連付けられている新しいセキュリティグループを作成します。セキュリティグループの名前は `rds-ec2-`*n* で、*n* は数字です。このセキュリティグループには、EC2 VPC セキュリティグループ (ファイアウォール) をソースとするインバウンドルールが含まれています。DB クラスターに関連付けられているこのセキュリティグループにより、EC2 インスタンスが DB クラスターにアクセスできます。 また、RDS では EC2 インスタンス関連付けられている新しいセキュリティグループを作成します。セキュリティグループの名前は `ec2-rds-`*n* で、*n* は数字です。このセキュリティグループには、DB クラスターの VPC セキュリティグループをソースとするアウトバウンドルールが含まれています。このセキュリティグループにより、EC2 インスタンスは DB クラスターにトラフィックを送信できます。 [Create new] (新規作成) を選択して、新しいセキュリティグループの名前を入力すると、別のセキュリティグループを新規に追加できます。 既存のセキュリティグループを追加するには、[Choose existing] (既存を選択) を選択し、追加するセキュリティグループを選択します。
アベイラビリティーゾーン	RDS では、マルチ AZ DB クラスター配置内の 1 つの DB インスタンスの EC2 インスタンスのアベイラビリティーゾーンを選択します。RDS は他の両方の DB インスタンスに対して異なるアベイラビリティーゾーンをランダムに選択します。書き込み DB インスタンスは、EC2 インスタンスと同じアベイラビリティーゾーンに作成されます。フェイルオーバーと書き込み DB インスタンスが異なるアベイラビリティーゾーンにある場合、アベイラビリティーゾーン間のコストが発生する可能性があります。

これらの設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」をご参照ください。

DB クラスターの作成後にこれらの設定を変更すると、EC2 インスタンスと DB クラスター間の接続に影響する可能性があります。

ネットワークを手動で設定する

同じ VPC 内の EC2 インスタンス以外のリソースから DB クラスターに接続するには、ネットワーク接続を手動で設定します。AWS マネジメントコンソール を使用して マルチ AZ DB クラスターを作成する場合は、お客様に代わって Amazon RDS に VPC を自動的に作成させることができます。または、既存の VPC を使うか、 マルチ AZ DB クラスター用に新しい VPC を作成することができます。マルチ AZ DB クラスターで使用するには、VPC の少なくとも 3 つのアベイラビリティーゾーンのそれぞれに少なくとも 1 つのサブネットが必要です。VPC の詳細については、「Amazon VPC と Amazon RDS」を参照してください。

デフォルト VPC を持っていない、または VPC を作成しておらず、コンソールを使用する予定がない場合は、次の操作を実行します。

- DB クラスターをデプロイする AWSリージョンで、少なくとも 3 つのアベイラビリティーゾーンのそれぞれに 1 つ以上のサブネットを持つ VPC を作成します。(詳しくは、「VPC 内の DB インスタンスの使用」を参照してください。)
- DB クラスターへの接続を許可する VPC セキュリティグループを指定します。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」および「セキュリティグループによるアクセス制御」を参照してください。
- マルチAZ DB クラスターが使用できる VPC 内の最低 3つのサブネットを定義する RDS DBサブネットグループを指定します。(詳しくは、「DB サブネットグループの使用」を参照してください。)

マルチ AZ DB クラスターに適用される制限事項については、「Amazon RDS のマルチ AZ DB クラスターの制限事項」を参照してください。

マルチ AZ DB クラスターと同じ VPC にないリソースに接続する場合は、VPC の DB インスタンスにアクセスするシナリオ の該当するシナリオを参照してください。

追加の前提条件

マルチ AZ DB クラスターを作成する前に、以下に示す追加の前提条件を検討してください。

- DB クラスターの設定パラメータを調整するには、必要なパラメータ設定を持つ DB クラスターパラメータグループを指定します。DB クラスターのパラメータグループの作成または変更については、「マルチ AZ DB クラスターのパラメータグループ 」を参照してください。
- DB クラスター用に指定する TCP/IP ポート番号を確認します。一部の会社のファイアウォールによっては、デフォルトポートへの接続がブロックされます。会社のファイアウォールがデフォルトのポートをブロックする場合は、お客様の DB クラスター用に別のポートを選択します。DB クラスターのすべての DB インスタンスは同じポートを使用します。
- データベースのメジャーエンジンバージョンが RDS 標準サポート終了日に達した場合は、延長サポート CLI オプションまたは RDS API パラメータを使用する必要があります。詳細については、「マルチ AZ DB クラスターを作成するための設定」の「RDS 延長サポート」を参照してください。

DB クラスターの作成

マルチAZ DB クラスターは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して作成できます。

コンソール

マルチAZ DB クラスターを作成するには、[Availability and durability] (可用性と耐久性) セクションで[Multi-AZ DB cluster] (マルチAZ DB クラスター) を選択します。

コンソールを使用して マルチAZ DB クラスターを作成します

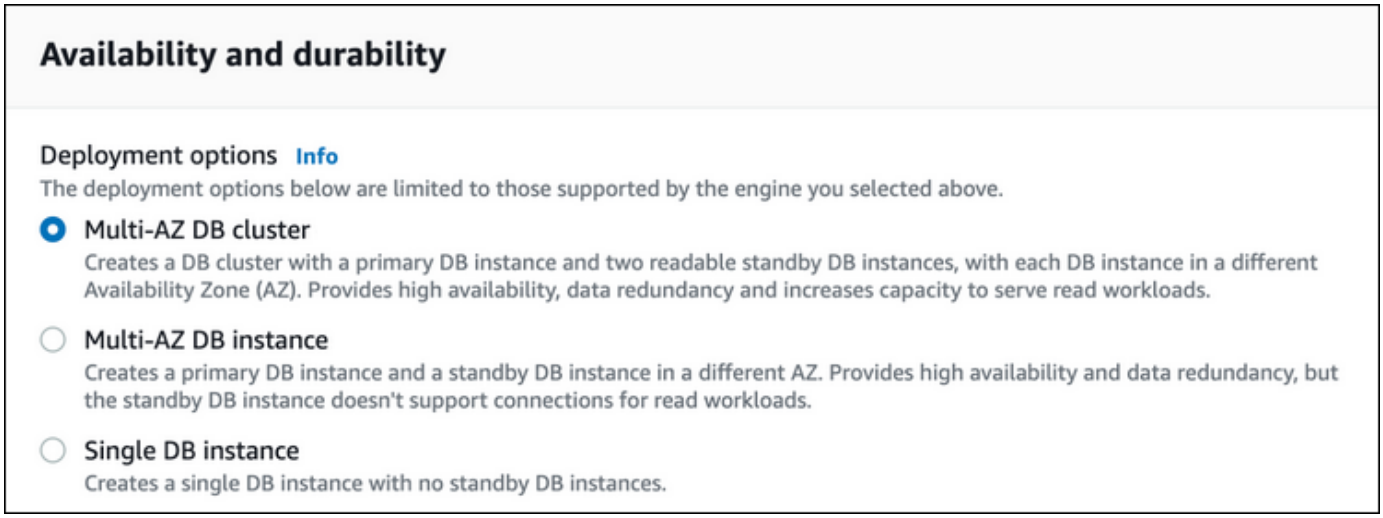
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. AWS マネジメントコンソールで、DB クラスターを作成する AWS リージョンを選択します。

   マルチ AZ DB クラスターをサポートする AWS リージョン の詳細ついては、「Amazon RDS のマルチ AZ DB クラスターの制限事項」を参照してください。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。

   マルチ AZ DB クラスターを作成するには、スタンダード作成が選択され、イージークリエイトではないことを確認します。
5. エンジンのタイプで、MySQLまたはPostgreSQLを選択します。
6. 「バージョン」 で、DB のエンジンバージョンを選択します。

マルチ AZ DB クラスターをサポートする DB エンジンバージョンの詳細ついては、「Amazon RDS のマルチ AZ DB クラスターの制限事項」を参照してください。

7. テンプレートで、デプロイに適したテンプレートを選択します。
8. [Availability and durability] (可用性と耐久性) で、[Multi-AZ DB cluster] (マルチ AZ DB クラスター)-選択します。



9. [DB cluster identifier] (DB クラスター識別子)で、DB クラスターの識別子を入力します。
10. マスターユーザーネームで、マスターユーザーネームを入力するか、デフォルトの設定のままにします。
11. マスターパスワードを入力します。
    a. [設定] セクションで、[認証情報の設定] を開きます。
    b. パスワードを指定する場合は、「パスワードの自動生成」ボックスが選択されている場合はオフにします。
    c. (オプション) マスターユーザーネームの値を変更します。
    d. マスターパスワードと確認パスワードに同じパスワードを入力します。
12. [DB インスタンスクラス] で、DB インスタンスクラスを選択します。サポートされているインスタンスクラスのリストについては、「the section called “マルチ AZ DB クラスターで利用できるインスタンスクラス”」を参照してください。
13. (オプション) この DB クラスターのコンピューティングリソースへの接続を設定します。

    DB クラスターの作成時に、Amazon EC2 インスタンスと新しい DB クラスター間の接続を設定できます。詳細については、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。

14. [VPC セキュリティグループ (ファイアウォール)] の [接続] セクションで [新規作成] を選択した場合、ローカルコンピュータの IP アドレスにデータベースへのアクセスを許可するインバウンドルールを使用して VPC セキュリティグループが作成されます。
15. 残りのセクションで、DB クラスター設定を指定します。各設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。
16. [データベースの作成] を選択します。

    自動生成されたパスワードを使用することを選択した場合は、[データベース] ページに [認証情報の詳細の表示] ボタンが表示されます。

    DB クラスターのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

    マスターユーザーとして DB クラスターに接続するには、表示されているユーザー名およびパスワードを使用します。

    > **Important**
    >
    > マスターユーザーのパスワードを再度表示することはできません。

17. 「データベース」 で、新しい DB クラスターの名前を選択します。

RDS コンソールに、新しい DB クラスターの詳細が表示されます。DB クラスターが作成され、使用できるようになるまで、DB クラスターのステータスは 「作成中」 になります。ステータスが [Available] (使用可能) に変わると、DB クラスターに接続できます。DB クラスタークラスと割り当てられたストレージによっては、新しい DB クラスターを使用できるようになるまで数分かかる可能性があります。

AWS CLI

「AWS CLI」を使用してマルチ AZ DB クラスターを作成する前に、必要な前提条件を満たすことを確認してください。これには、VPC と RDS DB サブネットグループの作成が含まれます。(詳しくは、「DB クラスターの前提条件」を参照してください。)

「AWS CLI」を使用してマルチ AZ DB クラスターを作成するには、create-db-clusterコマンドを呼び出します。「`--db-cluster-identifier`」を指定します。向けの「`--engine`」オプションで、`mysql`または`postgres`のいずれかを指定します。

各オプションの詳細については、「」を参照してください。マルチ AZ DB クラスターを作成するための設定

マルチ AZ DB クラスターをサポートする AWS リージョン、DB エンジン、および DB エンジンバージョンの詳細ついては、「Amazon RDS のマルチ AZ DB クラスターの制限事項」を参照してください。

`create-db-cluster`コマンドは、DB クラスターのライター DB インスタンスと 2 つのリーダー DB インスタンスを作成します。各 DB インスタンスが異なるアベイラビリティーゾーンにあります。

例えば、次のコマンドは `mysql-multi-az-db-cluster`という名前のMySQL 8.0 マルチ AZ DB クラスターを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster \
   --db-cluster-identifier mysql-multi-az-db-cluster \
   --engine mysql \
   --engine-version 8.0.32  \
   --master-username admin \
   --manage-master-user-password  \
   --port 3306 \
   --backup-retention-period 1  \
   --db-subnet-group-name default \
   --allocated-storage 4000 \
   --storage-type io1 \
   --iops 10000 \
   --db-cluster-instance-class db.m5d.xlarge
```

Windows の場合:

```
aws rds create-db-cluster ^
   --db-cluster-identifier mysql-multi-az-db-cluster ^
   --engine mysql ^
   --engine-version 8.0.32 ^
   --manage-master-user-password ^
   --master-username admin ^
   --port 3306 ^
   --backup-retention-period 1 ^
```

```
    --db-subnet-group-name default ^
    --allocated-storage 4000 ^
    --storage-type io1 ^
    --iops 10000 ^
    --db-cluster-instance-class db.m5d.xlarge
```

次のコマンドはpostgresql-multi-az-db-clusterという名前の PostgreSQL 13.4 マルチ AZ DB クラスターを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster \
    --db-cluster-identifier postgresql-multi-az-db-cluster \
    --engine postgres \
    --engine-version 13.4 \
    --manage-master-user-password \
    --master-username postgres \
    --port 5432 \
    --backup-retention-period 1  \
    --db-subnet-group-name default \
    --allocated-storage 4000 \
    --storage-type io1 \
    --iops 10000 \
    --db-cluster-instance-class db.m5d.xlarge
```

Windows の場合:

```
aws rds create-db-cluster ^
    --db-cluster-identifier postgresql-multi-az-db-cluster ^
    --engine postgres ^
    --engine-version 13.4 ^
    --manage-master-user-password ^
    --master-username postgres ^
    --port 5432 ^
    --backup-retention-period 1 ^
    --db-subnet-group-name default ^
    --allocated-storage 4000 ^
    --storage-type io1 ^
    --iops 10000 ^
    --db-cluster-instance-class db.m5d.xlarge
```

RDS API

RDS API を使用してマルチ AZ DB クラスターを作成する前に、VPC や RDS DB サブネットグループの作成など、必要な前提条件を満たすことを確認してください。(詳しくは、「DB クラスターの前提条件」を参照してください。)

RDS API を使用してマルチマスタークラスターを作成するには、CreateDBCluster オペレーションを実行します。「`DBClusterIdentifier`」 を指定します。`Engine`パラメータは、`mysql`または`postgresql`のいずれかを指定します。

各オプションの詳細については、「」を参照してください。マルチ AZ DB クラスターを作成するための設定

`CreateDBCluster`オペレーションは、DB クラスターのライター DB インスタンスと2つのリーダー DB インスタンスを作成します。各 DB インスタンスが異なるアベイラビリティーゾーンにあります。

マルチ AZ DB クラスターを作成するための設定

マルチ AZ DB クラスターを作成するときに選択する設定の詳細については、次の表を参照してください。AWS CLIのオプションの詳細については、create-db-clusterを参照してください。RDS API パラメータの詳細については、「CreateDBCluster」を参照してください。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

ストレージ割り当て	DB クラスター内の各 DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。(詳しくは、「Amazon RDS DB インスタンスストレージ」を参照してください。)	CLI オプション:
		
		`--allocated-storage`
		
		API パラメータ:
		
		`AllocatedStorage`

マイナーバージョン自動アップグレード	自動マイナーバージョンアップグレードを有効にして、DB クラスターが優先マイナー DB エンジンバージョンアップグレードを利用可能になったときに自動的に受信するようにします。Amazon RDS では、メンテナンスウィンドウでマイ	CLI オプション:
		
		`--auto-minor-version-upgrade`
		
		`--no-auto-minor-version-upgrade`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ
	ナーバージョンの自動アップグレードが実行されます。	API パラメータ: `AutoMinorVersionUpgrade`

バックアップの保存期間	DB クラスターの自動バックアップを保持する日数。マルチ AZ DB クラスターの場合、この値は**1**以上に設定する必要があります。
	
	(詳しくは、「バックアップの概要」を参照してください。)
		CLI オプション:
		
		`--backup-retention-period`
		
		API パラメータ:
		
		`BackupRetentionPeriod`

バックアップ期間:	Amazon RDS が DB クラスターのバックアップを自動的に作成する期間。データベースのバックアップを保持する期間を指定しない場合は、デフォルトの「指定なし」 を使用します。
	
	詳細については、「バックアップの概要」を参照してください。
		CLI オプション:
		
		`--preferred-backup-window`
		
		API パラメータ:
		
		`PreferredBackupWindow`

認証局	DB クラスターによって使用されるサーバー証明書の認定機関 (CA)。
	
	詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
		CLI オプション:
		
		`--ca-certificate-identifier`
		
		RDS API パラメータ:
		
		`CACertificateIdentifier`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

Copy tags to snapshots	このオプションは、DB スナップショットの作成時に、DB クラスターの任意のタグを、そのスナップショットにコピーします。
	
	(詳しくは、「Amazon RDS リソースのタグ付け」を参照してください。)
		CLI オプション:
		
		`-copy-tags-to-snapshot`
		
		`-no-copy-tags-to-snapshot`
		
		RDS API パラメータ:
		
		`CopyTagsToSnapshot`

データベース認証	使用するデータベース認証オプション。
	
	データベースパスワードのみを使用してデータベースのユーザーを認証するには、[パスワード認証] を選択します。
	
	ユーザーとロールでデータベースパスワードとユーザー認証情報を使用してデータベースユーザーを認証するには、[Password and IAM DB authentication] (パスワードと IAM DB 認証) を選択します。詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。
		CLI オプション:
		
		`--enable-iam-database-authentication`
		
		`--no-enable-iam-database-authentication`
		
		RDS API パラメータ:
		
		`EnableIAMDatabaseAuthentication`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

データベースポート	DB クラスターへのアクセスに使用するポート。デフォルトのポートが示されています。
	
	DB クラスターの作成後にポートを変更することはできません。
	
	一部の会社のファイアウォールは、これらのデフォルトポートへの接続をブロックします。会社のファイアウォールがデフォルトのポートをブロックする場合は、お客様の DB クラスター用に別のポートを選択します。
		CLI オプション:
		
		`--port`
		
		RDS API パラメータ:
		
		`Port`

DB クラスター識別子	DB クラスターの名前。オンプレミスのサーバーに名前を付けるのと同様に、DB クラスターに名前を付けます。DB クラスター識別子は、英数字 63 文字まで含めることができ、選択した AWS リージョン内で自分のアカウントに対して一意であることが必要です。	CLI オプション:
		
		`--db-cluster-identifier`
		
		RDS API パラメータ:
		
		`DBClusterIdentifier`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

DB インスタンスクラス:	マルチ AZ DB クラスター内の各 DB インスタンス (db.m5d.xlarge など) の計算容量とメモリ容量。
	
	可能であれば、一般的なクエリの作業セットをメモリに保持できる十分な大きさの DB インスタンスクラスを選択します。作業セットがメモリに保持されていると、システムによるディスクへの書き込みが回避され、これによりパフォーマンスが向上します。
	
	サポートされているインスタンスクラスのリストについては、「the section called “マルチ AZ DB クラスターで利用できるインスタンスクラス”」を参照してください。
		CLI オプション:
		
		`--db-cluster-instance-class`
		
		RDS API パラメータ:
		
		`DBClusterInstanceClass`

DB クラスターのパラメータグループ	DB クラスターに関連付ける DB クラスターのパラメータグループ。
	
	(詳しくは、「マルチ AZ DB クラスターのパラメータグループ」を参照してください。)
		CLI オプション:
		
		`--db-cluster-parameter-group-name`
		
		RDS API パラメータ:
		
		`DBClusterParameterGroupName`

DB エンジンバージョン	使用するデータベースエンジンのバージョン。	CLI オプション:
		
		`--engine-version`
		
		RDS API パラメータ:
		
		`EngineVersion`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ
DB クラスターのパラメータグループ	DB クラスターに関連付けられている DB インスタンスパラメータグループ。 詳細については、「マルチ AZ DB クラスターのパラメータグループ」を参照してください。	CLI オプション: `--db-cluster-parameter-group-name` RDS API パラメータ: **`DBClusterParameterGroupName`**

DB サブネットグループ:	DB クラスターで使用する DB サブネットグループ。
	既存の DB サブネットグループを使用するには、[Choose existing] (既存を選択) を選択します。次に、[Existing DB subnet groups] (既存の DB サブネットグループ) ドロップダウンリストから必要なサブネットグループを選択します。
	
	RDS が互換性のある DB サブネットグループを選択できるようにするには、[Automatic setup] (自動セットアップ) を選択します。存在しない場合、RDS はクラスターの新しいサブネットグループを作成します。
	
	詳細については、「DB サブネットグループの使用」を参照してください。
		CLI オプション:
		
		`--db-subnet-group-name`
		
		RDS API パラメータ:
		
		`DBSubnetGroupName`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

削除保護	削除保護を有効にして、DB クラスターが削除されないようにします。コンソールを使用して本番 DB クラスターを作成する場合、削除保護はデフォルトでオンになっています。
	
	(詳しくは、「DB インスタンスを削除する」を参照してください。)
		CLI オプション:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API パラメータ:
		
		`DeletionProtection`

暗号化:	この DB クラスターを保管時に暗号化するには、「Enable Encryption」 を選択します。
	
	マルチ AZ DB クラスターでは、暗号化がデフォルトでオンになっています。
	
	(詳しくは、「Amazon RDS リソースの暗号化」を参照してください。)
		CLI オプション:
		
		`--kms-key-id`
		
		`--storage-encrypted`
		
		`--no-storage-encrypted`
		
		RDS API パラメータ:
		
		`KmsKeyId`
		
		`StorageEncrypted`

拡張モニタリング	拡張モニタリングを有効にして、DB クラスターが実行されているオペレーティングシステムのメトリック収集をリアルタイムでオンにします。
	
	(詳しくは、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。)
		CLI オプション:
		
		`--monitoring-interval`
		
		`--monitoring-role-arn`
		
		RDS API パラメータ:
		
		`MonitoringInterval`
		
		`MonitoringRoleArn`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

初期データベース名:	DB クラスター上のデータベースの名前。名前を指定しないと、Amazon RDS は MySQL の DB クラスターにデータベースを作成しません。ただし、PostgreSQL 用の DB クラスターにはデータベースが作成されます。名前にはデータベースエンジンの予約語を使用できません。DB エンジンによっては、他の制約があります。
	
	MySQL:
	
	• 1～64 個の英数字を使用する必要があります。
	
	PostgreSQL:
	
	• 1～63 個の英数字を使用する必要があります。
	• 先頭は英字またはアンダースコアにする必要があります。後続の文字には、英字、アンダースコア、または数字 (0 ～ 9) を含めることができます。
	• 初期のデータベース名は `postgres` です。
		CLI オプション:
		
		`--database-name`
		
		RDS API パラメータ:
		
		`DatabaseName`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

ログのエクスポート	Amazon CloudWatch Logs に発行するデータベースログファイルのタイプ。
	
	詳細については、「Amazon CloudWatch Logs へのデータベースログの発行」を参照してください。
		CLI オプション:
		
		`-enable-cloudwatch-logs-exports`
		
		RDS API パラメータ:
		
		`EnableCloudwatchLogsExports`

メンテナンスウィンドウ	DB クラスターへの変更保留が適用される 30 分単位のウィンドウ。期間が重要ではない場合は、「No Preference」 を選択します。
	
	詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。
		CLI オプション:
		
		`--preferred-maintenance-window`
		
		RDS API パラメータ:
		
		`PreferredMaintenanceWindow`

でマスター認証情報を管理するAWS Secrets Manager	[Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択して、Secrets Manager でユーザーのパスワードをシークレットに管理します。
	
	オプションで、シークレットを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
	
	詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。
		CLI オプション:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		RDS API パラメータ:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

マスターパスワード:	マスターユーザーアカウントのパスワード。	CLI オプション:
		
		`--master-user-password`
		
		RDS API パラメータ:
		
		`MasterUserPassword`

マスターユーザーネーム	すべてのデータベース権限で DB クラスターにログオンするためのマスターユーザー名として使用する名前。
	
	• 1 ~ 16 文字の英数字とアンダースコアを使用できます。
	• 1 字目は文字である必要があります。
	• データベースエンジンの予約語は使用できません。
	
	マルチ AZ DB クラスターの作成後にマスターユーザー名を変更することはできません。
	
	マスターユーザーに付与される権限の詳細については、マスターユーザーアカウント権限を参照してください。
		CLI オプション:
		
		`--master-username`
		
		RDS API パラメータ:
		
		`MasterUsername`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

Performance Insights	Performance Insights を有効にして DB クラスターのロードをモニタリングし、データベースのパフォーマンスを分析およびトラブルシューティングできるようにします。
	
	保持期間を選択して、保持するパフォーマンスインサイトデータ履歴の量を決定します。保持期間の設定は、デフォルト (7 日間) です。パフォーマンスデータをさらに長期間保持するには、1～24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。
	
	このデータベースボリュームの暗号化に使用されるキーを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
	
	詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。
		CLI オプション:
		
		`--enable-performance-insights`
		
		`--no-enable-performance-insights`
		
		`--performance-insights-retention-period`
		
		`--performance-insights-kms-key-id`
		
		RDS API パラメータ:
		
		`EnablePerformanceInsights`
		
		`PerformanceInsightsRetentionPeriod`
		
		`PerformanceInsightsKMSKeyId`

プロビジョンド IOPS	初めに DB クラスターに割り当てられるプロビジョンド IOPS (1 秒あたりの入力/出力操作数) の合計です。	CLI オプション:
		
		`--iops`
		
		RDS API パラメータ:
		
		`Iops`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

パブリックアクセス	パブリック IP アドレスを DB クラスターに付与する場合は 「パブリックアクセス可能」。これにより、VPC 外からアクセス可能になります。パブリックにアクセス可能となるよう、DB クラスターは、VPC のパブリックサブネット内にある必要があります。
	
	VPC 内からのみ DB クラスターにアクセス可能にするには、「パブリックアクセス不可」 です。
	
	(詳しくは、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。)
	
	VPC の外部から DB クラスターに接続するには、DB クラスターがパブリックにアクセスできる必要があります。また、DB クラスターのセキュリティグループのインバウンドルールを使用してアクセスを許可し、その他の要件を満たしている必要があります。(詳しくは、「Amazon RDS DB インスタンスに接続できない」を参照してください。)
	
	DB クラスターがパブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続またはDirect Connect 接続を使用してプライベートネットワークからアクセスすることもできます。(詳しくは、「ネットワーク
		CLI オプション:
		
		`--publicly-accessible`
		
		`--no-publicly-accessible`
		
		RDS API パラメータ:
		
		`PubliclyAccessible`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ
	間のトラフィックのプライバシー」を参照してください。)	

RDS 延長サポート	RDS 標準サポート終了日を過ぎてもサポートされているメジャーエンジンバージョンを 引き続き実行するには、[RDS 延長サポートを有効にする] を選択します。
	
	DB クラスターを作成する場合、Amazon RDS はデフォルトで RDS 延長サポートを選択します。RDS の標準サポート終了日後に新しい DB クラスターが作成されて RDS 延長サポートの料金が発生するのを避けるには、この設定を無効にします。既存の DB クラスターについて、RDS 延長サポートの課金開始日以前に料金が発生することはありません。
	
	詳細については、「Amazon RDS の Amazon RDS 延長サポート」を参照してください。
		CLI オプション:
		
		`--engine-lifecycle-support`
		
		RDS API パラメータ:
		
		`EngineLifecycleSupport`

ストレージスループット	DB クラスターのストレージスループット値。この設定は、ストレージタイプに汎用 SSD (gp3) を選択した場合にのみ表示されます。 この設定はユーザーが指定した IOPS をもとに自動的に設定され、変更することはできません。 詳細については、「gp3 ストレージ (推奨)」を参照してください。	この値は自動的に計算されます。CLI オプションはありません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ
RDS Proxy	[Create an RDS Proxy] (RDS Proxy の作成) を選択して、DB クラスターにプロキシを作成します。Amazon RDS は、プロキシの IAM ロールと Secrets Manager シークレットを自動的に作成します。	DB クラスターの作成時には使用できません。

ストレージタイプ	DB クラスターのストレージタイプ。
	
	汎用 SSD (gp3)、プロビジョンド IOPS (io1)、およびプロビジョンド IOPS SSD (io2) ストレージのみがサポートされています。
	
	詳細については、「Amazon RDS ストレージタイプ」を参照してください。
		CLI オプション:
		
		`--storage-type`
		
		RDS API パラメータ:
		
		`StorageType`

仮想プライベートクラウド (VPC)。	この DB クラスターと関連付ける Amazon VPC サービスに基づく VPC。 (詳しくは、「Amazon VPC と Amazon RDS」を参照してください。)	CLI と API の場合は、VPC セキュリティグループ ID を指定します。

VPC セキュリティグループ (ファイアウォール)	DB クラスターに関連付けるセキュリティグループ。
	
	(詳しくは、「VPC セキュリティグループの概要」を参照してください。)
		CLI オプション:
		
		`--vpc-security-group-ids`
		
		RDS API パラメータ:
		
		`VpcSecurityGroupIds`

マルチ AZ DB クラスターの作成時に適用されない設定

AWS CLI コマンドの `create-db-cluster`、および RDS API の `CreateDBCluster` オペレーションでの以下の設定は、マルチ AZ DB クラスターには適用されません。

コンソールでマルチ AZ DB クラスターにこれらの設定を指定することもできません。

AWS CLI の設定	RDS API の設定
`--availability-zones`	`AvailabilityZones`
`--backtrack-window`	`BacktrackWindow`
`--character-set-name`	`CharacterSetName`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-global-write-forwarding | --no-enable-global-write-forwarding`	`EnableGlobalWriteForwarding`
`--enable-http-endpoint | --no-enable-http-endpoint`	`EnableHttpEndpoint`
`--global-cluster-identifier`	`GlobalClusterIdentifier`
`--option-group-name`	`OptionGroupName`
`--pre-signed-url`	`PreSignedUrl`
`--replication-source-identifier`	`ReplicationSourceIdentifier`
`--scaling-configuration`	`ScalingConfiguration`

Amazon RDS のマルチ AZ DB クラスターへの接続

マルチ AZ DB クラスターには、単一の DB インスタンスではなく、3 つの DB インスタンスがあります。各接続は特定の DB インスタンスで処理されます。マルチ AZ DB クラスターに接続するとき、指定するホスト名とポートは、エンドポイントと呼ばれる完全修飾ドメイン名を指します。マルチ AZ DB クラスターは、エンドポイントメカニズムを使用してこれらの接続を抽象化するため、DB クラスター内のどの DB インスタンスに接続するかを正確に指定する必要はありません。そのため、すべてのホスト名をハードコード化したり、一部の DB インスタンスが使用できないときに接続を再ルーティングするための独自のロジックを書いたりする必要はありません。

ライターエンドポイントは、読み取りと書き込みの両方の操作をサポートする DB クラスターのライター DB インスタンスに接続します。リーダーエンドポイントは、読み取り操作のみをサポートする 2 つのリーダー DB インスタンスのいずれかに接続します。

エンドポイントを使用すると、ユースケースに基づいて各接続を対応する DB インスタンスまたは DB インスタンスグループにマッピングできます。例えば、DDL および DML ステートメントを実行するために、ライター DB インスタンスであるどちらの DB インスタンスにも接続できます。クエリを実行するには、リーダーエンドポイントに接続して、マルチ AZ DB クラスターがリーダー DB インスタンス間の接続を自動的に管理するようにします。診断またはチューニングの場合は、特定の DB インスタンスエンドポイントに接続して、特定の DB インスタンスに関する詳細を調査できます。

DB インスタンスへの接続方法については、「Amazon RDS DB インスタンスへの接続」を参照ください。

マルチ AZ DB クラスターへの接続の詳細については、以下のトピックを参照してください。

トピック

- クラスターエンドポイント
- リーダーエンドポイント
- インスタンスエンドポイント
- 高可用性接続
- Amazon RDS の AWS ドライバーを使用したマルチ AZ DB クラスターへの接続

マルチ AZ DB クラスターエンドポイントの種類

エンドポイントは、ホストアドレスを含む一意の ID によって表されます。マルチ AZ DB クラスターでは、以下のタイプのエンドポイントを使用できます。

クラスターエンドポイント

マルチ AZ DB クラスターのクラスターエンドポイント (またはライターエンドポイント) は、その DB クラスターの現在のライター DB インスタンスに接続します。このエンドポイントは、DDL や DML ステートメントなどの書き込みオペレーションを実行できる唯一のエンドポイントです。このエンドポイントは、読み取り操作を実行することもできます。

マルチ AZ DB クラスターごとに 1 つのクラスターエンドポイントと 1 つのライター DB インスタンスがあります。

クラスターエンドポイントは、DB クラスターに対するすべての書き込みオペレーション (挿入、更新、削除、DDL の変更など) で使用します。クラスターエンドポイントは、クエリなどの読み取りオペレーションでも使用できます。

DB クラスターの現在のライター DB インスタンスが失敗した場合、マルチ AZ DB クラスターは新しいライター DB インスタンスに自動的にフェイルオーバーします。フェイルオーバー中、DB クラスターは、新しいライター DB インスタンスからクラスターエンドポイントへの接続リクエストに継続して対応し、サービスの中断は最小限に抑えられます。

次の例では、マルチ AZ DB クラスターのクラスターエンドポイントを示します。

```
mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com
```

クラスターエンドポイントへの接続に関する詳細は、「クラスターエンドポイント」を参照してください。

リーダーエンドポイント

マルチAZ DB クラスターのリーダーエンドポイントは、DB クラスターへの読み取り専用接続をサポートしています。リーダーエンドポイントは、SELECT クエリなどの読み取りオペレーションで使用します。このエンドポイントは、リーダー DB インスタンスでこれらのステートメントを処理することにより、ライター DB インスタンスのオーバーヘッドを削減します。また、クラスターが同時SELECTクエリを処理する能力を拡張するのにも役立ちます。マルチAZ DBクラスターごとに 1 つのリーダーエンドポイントがあります。

リーダーエンドポイントは、リーダー DB インスタンスの 1 つに各接続リクエストを送信します。セッションにリーダーエンドポイントを使用する場合、そのセッションのSELECTなどで読み取り専用ステートメントのみを実行できます。

マルチ AZ DB クラスターのリーダーエンドポイントを次の例に示します。リーダーエンドポイントの読み取り専用インテントは、クラスターエンドポイント名内の `-ro` で示されます。

```
mydbcluster.cluster-ro-123456789012.us-east-1.rds.amazonaws.com
```

リーダーエンドポイントへの接続に関する詳細は、「リーダーエンドポイント」を参照してください。

インスタンスエンドポイント

インスタンスエンドポイントは、マルチ AZ DB クラスター内の特定の DB インスタンスに接続します。DB クラスターの各 DB インスタンスには、独自のインスタンスエンドポイントがあります。したがって、DB クラスター内の現在のライター DB インスタンスに1つのインスタンスエンドポイントがあり、DB クラスター内のリーダー DB ごとに 1 つのインスタンスエンドポイントがあります。

インスタンスエンドポイントは、DB クラスターへの接続の直接制御を提供します。この制御は、クラスターエンドポイントやリーダーエンドポイントの使用が適切でないシナリオに対処するのに役立ちます。例えば、ワークロードタイプに基づき、さらにきめ細かいロードバランシングがアプリケーションに必要になる場合があります。この場合、DB クラスター内の異なるリーダー DB インスタンスに接続して読み取りワークロードを配信するように、複数のクライアントを設定できます。

次の例では、 マルチ AZ DB クラスターの DB インスタンスのインスタンスエンドポイントを示します。

```
mydbinstance.123456789012.us-east-1.rds.amazonaws.com
```

インスタンスエンドポイントへの接続の詳細については、「インスタンスエンドポイント」を参照してください。

エンドポイントの表示

コンソール、AWS CLI、または Amazon RDS API を使用して、クラスター、リーダー、インスタンスエンドポイントを表示します。

Console

AWS マネジメントコンソールでは、それぞれのマルチ AZ DB クラスターの詳細ページにクラスターエンドポイントとリーダーエンドポイントが表示されます。インスタンスエンドポイントは、各 DB インスタンスの詳細ページに表示されます。

AWS CLI

AWS CLIでは、describe-db-clusters コマンドの出力にライターエンドポイントとリーダーエンドポイントが表示されます。例えば、次のコマンドは、現在の AWS リージョンにあるすべてのクラスターのエンドポイント属性を表示します。

```
aws rds describe-db-cluster-endpoints
```

Amazon RDS API

Amazon RDS API では、DescribeDBClusterEndpoints アクションを呼び出してエンドポイントを取得します。出力には Amazon Aurora DB クラスターエンドポイントが存在する場合も表示されます。

クラスターエンドポイント

それぞれのマルチ AZ DB クラスターには単一の組み込みクラスターエンドポイントがあり、その名前とその他の属性は Amazon RDS によって管理されます。ユーザーが、この種のエンドポイントを作成、削除、または変更することはできません。

クラスターエンドポイントは、DB クラスターの管理、抽出/変換/ロード (ETL) オペレーションの実行、およびアプリケーションの開発やテストに使用します。クラスターエンドポイントは、クラスターのライター DB インスタンスに接続します。ライター DB インスタンスは、テーブルとインデックスの作成、ステートメントINSERTの実行、およびその他の DDL および DML 操作を実行できる唯一の DB インスタンスです。

フェイルオーバーメカニズムが新しい DB インスタンスをクラスターのライター DB インスタンスに昇格させると、クラスターエンドポイントが指す物理 IP アドレスが変更されます。何らかの形式の接続プールや他の多重化を使用している場合は、キャッシュされた DNS 情報の有効時間をフラッシュまたは削減する必要があります。これにより、フェイルオーバー後に使用不可または読み取り専用になった DB インスタンスに読み取り/書き込み接続を試行できないようにします。

リーダーエンドポイント

マルチ AZ DBクラスターへの読み取り専用接続にはリーダーエンドポイントを使用します。このエンドポイントは、DB クラスターでクエリを大量に消費するワークロードを処理するのに役立ちます。リーダーエンドポイントは、クラスターに対してレポートや他の読み取り専用のオペレーションを実行するアプリケーションに指定します。リーダーエンドポイントは、マルチ AZ DB クラスターで使用可能なリーダーDBインスタンスへの接続を送信します。

それぞれのマルチ AZ クラスターごとに組み込まれている単一のリーダーエンドポイントの名前や他の属性は Amazon RDS で管理されます。ユーザーが、この種のエンドポイントを作成、削除、または変更することはできません。

インスタンスエンドポイント

マルチ AZ DBクラスターの DB インスタンスごとに個別に組み込まれているインスタンスエンドポイントの名前や他の属性は Amazon RDS で管理されます。ユーザーが、この種のエンドポイントを作成、削除、または変更することはできません。マルチ AZ DB クラスターでは、通常、インスタンスエンドポイントよりもライターエンドポイントとリーダーエンドポイントを頻繁に使用します。

日常的なオペレーションでインスタンスエンドポイントを主に使用するのは、マルチ AZ DBクラスター内の特定の DB インスタンスに影響している容量やパフォーマンスの問題を診断する場合です。特定の DB インスタンスに接続しているときに、そのステータス可変、メトリクスなどを調査できます。これを行うと、クラスター内の他の DB インスタンスで起こっていることは異なる、その DB インスタンスで起こっていることを判断するのに役立ちます。

高可用性接続

高可用性が重要であるマルチ AZ DB クラスターでは、ライターエンドポイントを読み取り/書き込み接続や汎用接続に使用し、リーダーエンドポイントを読み取り専用接続に使用します。ライターエンドポイントとリーダーエンドポイントは、インスタンスエンドポイントよりも DB インスタンスのフェイルオーバーを適切に管理します。インスタンスエンドポイントとは異なり、ライターエンドポイントとリーダーエンドポイントは、クラスター内の DB インスタンスが利用できなくなった場合に、接続先の DB インスタンスを自動的に変更します。

DB クラスターのライター DBインスタンスが失敗した場合、 Amazon RDS は新しいライター DB インスタンスに自動的にフェイルオーバーします。これは、リーダー DB インスタンスを新しいライター DB インスタンスに昇格させることによって行われます。フェイルオーバーが発生した場合、ライターエンドポイントを使用して、新しく昇格させたライター DB インスタンスに再接続できます。または、リーダーエンドポイントを使用して DB クラスター内のリーダー DB インスタンスの 1 つに再接続することもできます。フェイルオーバー中に、リーダー DB インスタンスが新しいライター

DB インスタンスに昇格された後、リーダーエンドポイントが DB クラスターの新しいライター DB インスタンスへの接続を短時間指示する場合があります。インスタンスエンドポイントへの接続を管理するように独自のアプリケーションロジックを設計する場合は、DB クラスター内の使用可能な DB インスタンスの結果セットを手動またはプログラムで検出できます。

Amazon RDS の AWS ドライバーを使用したマルチ AZ DB クラスターへの接続

AWS のドライバースイートは、スイッチオーバーとフェイルオーバーの時間の短縮、AWS Secrets Manager、AWS Identity and Access Management (IAM)、フェデレーティッド ID での認証をサポートするように設計されています。AWS ドライバーは、DB クラスターステータスをモニタリングし、クラスタートポロジを認識して新しいライターを決定することを前提としています。このアプローチにより、スイッチオーバーとフェイルオーバーの時間が 1 桁秒に短縮されます (オープンソースドライバーの場合は数十秒)。

新しいサービス機能が導入されるにあたって、こうしたサービス機能を標準でサポートすることが AWS のドライバースイートの目標です。

Amazon Web Services (AWS) JDBC ドライバーを使用したマルチ AZ DB クラスターへの接続

Amazon Web Services (AWS) JDBC ドライバーは、アプリケーションでクラスター化されたデータベースの機能を利用する際に役立つ高度な JDBC ラッパーとして設計されています。このラッパーは、既存の JDBC ドライバーの機能を補完し、拡張します。このドライバーには、以下のコミュニティドライバーとドロップイン互換性があります。

- MySQL Connector/J
- MariaDB Connector/J
- pgJDBC

AWS JDBC ドライバーをインストールするには、AWS JDBC ドライバーの .jar ファイル (CLASSPATH アプリケーション内) を追加して、それぞれのコミュニティドライバーへの参照を保持します。対応する接続 URL プレフィックスを次のように更新します。

- `jdbc:mysql://` ~ `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` ~ `jdbc:aws-wrapper:mariadb://`
- `jdbc:postgresql://` ~ `jdbc:aws-wrapper:postgresql://`

AWS JDBC ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) JDBC ドライバー GitHub リポジトリ」を参照してください。

Amazon Web Services (AWS) Python ドライバーを使用したマルチ AZ DB クラスターへの接続

Amazon Web Services (AWS) Python ドライバーは、高度な Python ラッパーとして設計されています。このラッパーは、オープンソースの Psycopg ドライバーの機能を補完し、拡張します。AWS Python ドライバーは Python バージョン 3.8 以降をサポートしています。`aws-advanced-python-wrapper` パッケージは、`pip` コマンドと `psycopg` オープンソースパッケージを使用してインストールできます。

AWS Python ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) Python Driver GitHub repository」を参照してください。

AWS コンピューティングリソースと Amazon RDS のマルチ AZ DB クラスターを自動的に接続する

マルチ AZ DB クラスターと、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスや AWS Lambda 関数などの AWS コンピューティングリソースを自動的に接続できます。

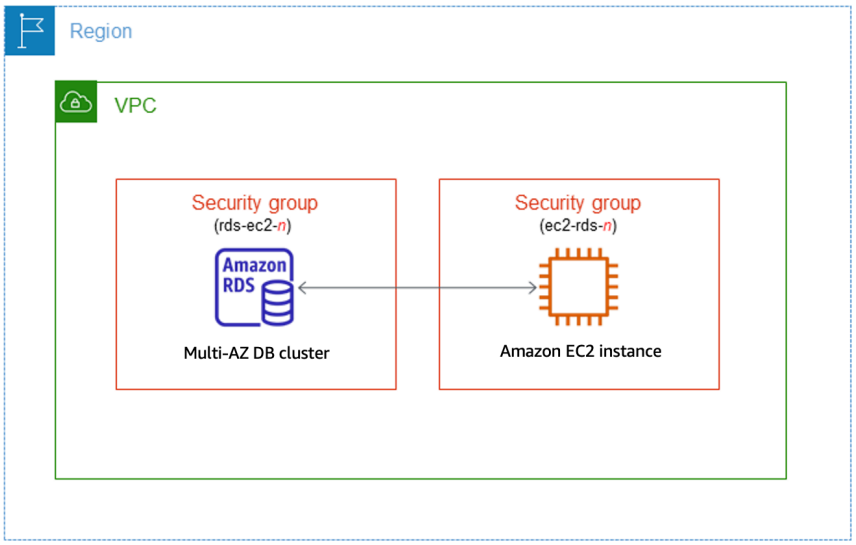
以下のトピックでは、マルチ AZ DB クラスターデプロイ内の Amazon RDS DB インスタンスへの信頼性の高い接続を確立するために、ネットワーク設定、セキュリティグループ、および接続パラメータを設定する詳細な手順について説明します。これらは、マルチ AZ DB クラスターとやり取りするアプリケーションのネットワーク接続とパフォーマンスを最適化し、安全で効率的なデータオペレーションを保証することに重点を置いています。

トピック

- EC2 インスタンスとマルチ AZ DB クラスターを自動的に接続する
- Lambda 関数とマルチ AZ DB クラスターを自動的に接続する

EC2 インスタンスとマルチ AZ DB クラスターを自動的に接続する

Amazon RDS コンソールを使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスとマルチ AZ DB クラスターとの接続を簡単に設定できます。多くの場合、マルチ AZ DB クラスターはプライベートサブネットにあり、EC2 インスタンスは VPC 内のパブリックサブネットにあります。EC2 インスタンスの SQL クライアントを使用すると、マルチ AZ DB クラスターに接続できます。EC2 インスタンスは、プライベートマルチ AZ DB クラスターにアクセスするウェブサーバーやアプリケーションを実行することも可能です。



マルチ AZ DB クラスターと同じ VPC にない EC2 インスタンスに接続する場合は、the section called “VPC の DB インスタンスにアクセスするシナリオ” のシナリオを参照してください。

トピック

- EC2 インスタンスとの自動接続の概要
- EC2 インスタンスとマルチ AZ DB クラスターを自動的に接続する
- 接続中のコンピューティングリソースを表示する

EC2 インスタンスとの自動接続の概要

EC2 インスタンスとマルチ AZ DB クラスター間の自動接続をセットアップすると、Amazon RDS は、EC2 インスタンスと DB クラスターの VPC セキュリティグループを設定します。

EC2 インスタンスとマルチ AZ DB クラスターを接続するための要件は次のとおりです。

- EC2 インスタンスはマルチ AZ DB クラスター と同じ VPC に存在する必要があります。

  同じ VPC に EC2 インスタンスが存在しない場合、コンソールには EC2 インスタンス作成用のリンクが表示されます。
- 接続を設定するユーザーには、次の EC2 オペレーションを実行する権限が必要です。
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

EC2 インスタンスへの接続を設定すると、次の表で示されているように、Amazon RDS は、マルチ AZ DB クラスターと EC2 インスタンスに関連付けられているセキュリティグループの現在の設定に従って動作します。

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* (*n* は数字) に一致する名前を持つのセキュリティグループが 1 つ以上あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	パターン `rds-ec2-`*n* (*n* は数字) に一致する名前の EC2 インスタンスに関連付けられたセキュリティグループが 1 つまたは複数存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが 1 つのみ存在します。	Amazon RDS は何もしません。 EC2 インスタンスとマルチ AZ DB クラスターとの接続は、既に自動で設定されています。EC2 インスタンスと RDS データベースの間には既に接続が存在するため、セキュリティグループは変更されません。
次の条件のいずれかが適用されます。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループはありません。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループが 1 つ以上あります。ただし、これらのセキュリティグループは、いずれも EC2 インスタンスとの接続には使用できません。	次の条件のいずれかが適用されます。 • EC2 インスタンスに関連付けされた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループは存在しません。 • EC2 インスタンスに関連付けられた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、これらのセキュリティグループは、いずれもマルチ AZ DB クラスターとの接続には使用できません	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
セキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しない場合は使用できません。また、セキュリティグループが変更されている場合は使用できません。変更の例としては、ルールの追加や、既存ルールのポート変更などがあります。	。ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが 1 つも存在しない場合、セキュリティグループを使用できません。また、セキュリティグループが変更されている場合は使用できません。	
マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループが 1 つ以上あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	EC2 インスタンスに関連付けられた、パターン `ec2-rds-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。ただし、これらのセキュリティグループは、いずれもマルチ AZ DB クラスターとの接続には使用できません。ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが 1 つも存在しない場合、セキュリティグループを使用できません。また、セキュリティグループが変更されている場合は使用できません。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループが 1 つ以上あります。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	接続に有効な EC2 セキュリティグループは存在しますが、EC2 インスタンスに関連付けられていません。このセキュリティグループには、パターン `rds-ec2-`*n* に一致する名前が付いています。このセキュリティグループには変更が加えられていません。ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが 1 つのみ存在します。	RDS action: associate EC2 security group

現在の RDS セキュリティグループの設定	現在の EC2 セキュリティグループの設定	RDS アクション
次の条件のいずれかが適用されます。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループはありません。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-ec2-`*n* に一致する名前を持つセキュリティグループが 1 つ以上あります。ただし、これらのセキュリティグループは、いずれも EC2 インスタンスとの接続には使用できません。セキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しない場合は使用できません。また、セキュリティグループが変更されている場合は使用できません。	EC2 インスタンスに関連付けられた、パターン `rds-ec2-`*n* に一致する名前のセキュリティグループが 1 つまたは複数存在します。パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが 1 つのみ存在します。	RDS action: create new security groups

RDS アクション: 新しいセキュリティグループを作成する

Amazon RDS は以下のアクションを実行します。

- パターン `rds-ec2-n` に一致する新しいセキュリティグループを作成します。このセキュリティグループには、EC2 インスタンスの VPC セキュリティグループをソースとするインバウンドルールが存在します。このセキュリティグループはマルチ AZ DB クラスターに関連付けられていて、これにより、EC2 インスタンスはマルチ AZ DB クラスターにアクセスできます。
- パターン `ec2-rds-n` に一致する新しいセキュリティグループを作成します。このセキュリティグループには、ソースとしてのマルチ AZ DB クラスター の VPC セキュリティグループにアウトバウンドルールが存在します。このセキュリティグループは EC2 インスタンスに関連付けられ、これにより EC2 インスタンスはマルチ AZ DB クラスターにトラフィックを送信できます。

RDS アクション: EC2 セキュリティグループを関連付ける

Amazon RDS は、有効な既存の EC2 セキュリティグループを EC2 インスタンスに関連付けます。このセキュリティグループにより、EC2 インスタンスはマルチ AZ DB クラスターにトラフィックを送信できます。

EC2 インスタンスとマルチ AZ DB クラスターを自動的に接続する

EC2 インスタンスと RDS データベース との接続を設定する前に、「EC2 インスタンスとの自動接続の概要」で説明されている要件を満たしていることを確認してください。

接続の設定後にこれらのセキュリティグループを変更すると、EC2 インスタンスと RDS データベース との接続に影響する可能性があります。

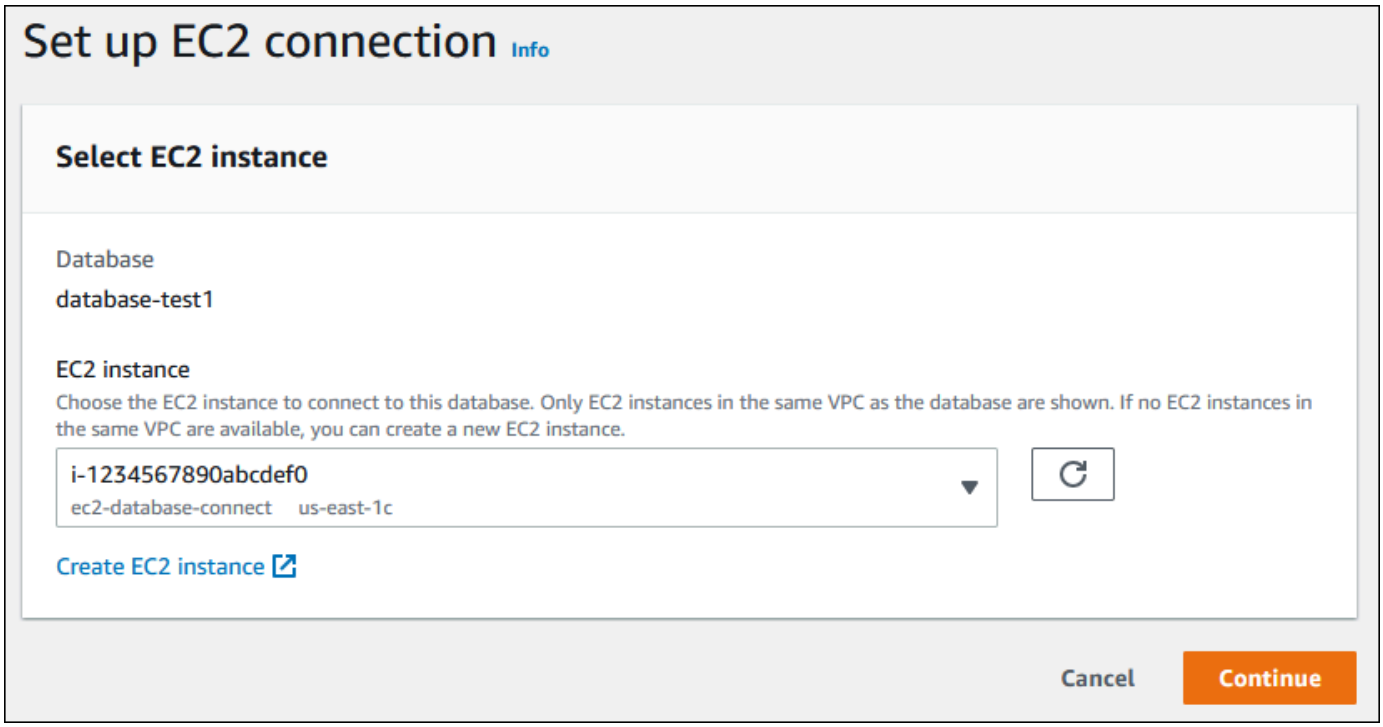
> **Note**
>
> AWS マネジメントコンソール を使用することでのみ、EC2 インスタンスと RDS データベース との接続を自動で設定できます。AWS CLI または RDS API を使用して自動で接続を設定することはできません。

EC2 インスタンスと RDS データベース を自動的に接続するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、RDS データベース のリンクを選択します。
3. [アクション] から [EC2 接続の設定] を選択します。

[Set up EC2 connection] (EC2 接続の設定) ページが表示されます。

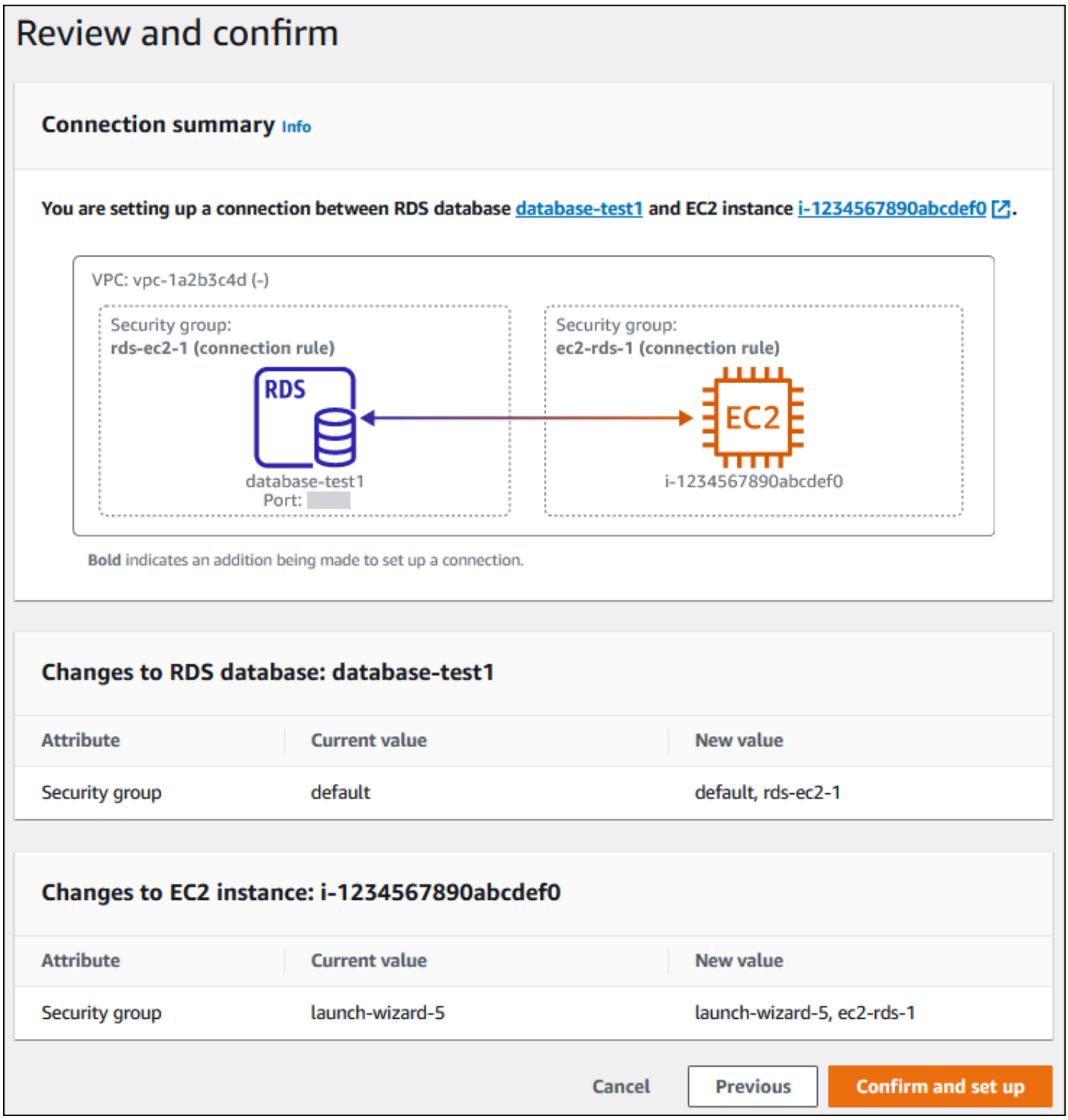
4. [Set up EC2 connection] (EC2 接続の設定) ページで、[EC2 instance] (EC2 インスタンス) を選択します。



同じ VPC に EC2 インスタンスが存在しない場合は、[Create EC2 instance] (EC2 インスタンスの作成) を選択します。この場合、新しい EC2 インスタンスが RDS データベース と同じ VPC にあることを確認してください。

5. [続行] をクリックしてください。

[Review and confirm] (確認と確定) ページが表示されます。



6. [Review and confirm] (確認と確定) ページで、EC2 インスタンスとの接続を設定するために RDS が行う変更を確認します。

   変更が正しければ、[確認とセットアップ] を選択します。

   変更内容が正しくない場合は、[Previous] (前へ) または [Cancel] (キャンセル) を選択します。

接続中のコンピューティングリソースを表示する

AWS マネジメントコンソール を使用して、RDS データベースに接続されているコンピューティングリソースを表示できます。表示されるリソースには、自動的に設定されたコンピューティングリソース接続が含まれます。コンピューティングリソースとの接続は、次の方法で自動的に設定できます。

- データベースを作成するときに、コンピューティングリソースを選択できます。

  詳細については、「Amazon RDS DB インスタンスの作成」および「Amazon RDS 用のマルチ AZ DB クラスターの作成」を参照してください。
- 既存のデータベースとコンピューティングリソース間の接続を設定できます。

  詳細については、「EC2 インスタンスと RDS データベースを自動的に接続する」を参照してください。

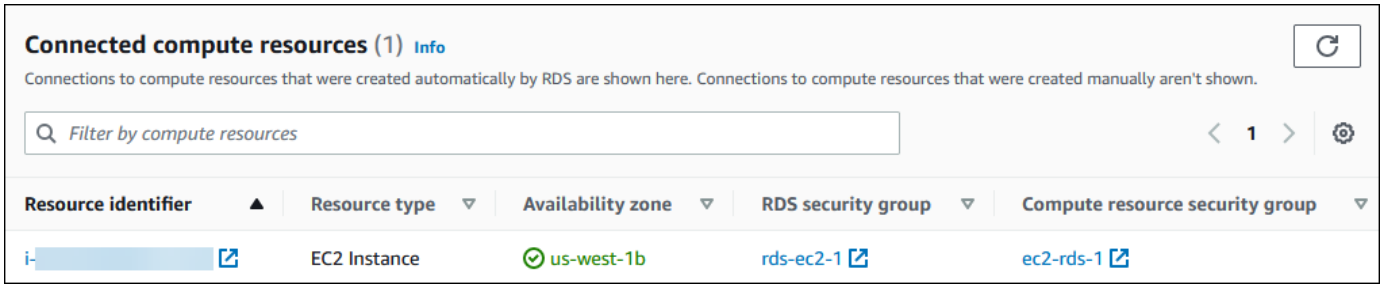
コンピューティングリソースリストには、手動でデータベースに接続されたものは含まれていません。例えば、データベースに関連付けられた VPC セキュリティグループにルールを追加することで、コンピューティングリソースがデータベースに手動でアクセスできるようになります。

コンピューティングリソースをリスト化するには、次の条件を満たしている必要があります。

- コンピューティングリソースに関連付けられているセキュリティグループの名前がパターン `ec2-rds-`*n* (*n* は数字) と一致する。
- コンピューティングリソースに関連付けられたセキュリティグループには、ポート範囲が RDS データベースが使用するポートに設定されたアウトバウンドルールがあります。
- コンピューティングリソースに関連付けられたセキュリティグループには、ソースが RDS データベース に関連付けられたセキュリティグループに設定されたアウトバウンドルールがある。
- RDS データベースに関連付けられたセキュリティグループの名前が、パターン `rds-ec2-`*n* (*n* は数字) に一致する。
- RDS データベース に関連付けられたセキュリティグループには、ポート範囲が RDS データベース が使用するポートに設定されたインバウンドルールがあります。
- RDS データベース に関連付けられたセキュリティグループには、ソースがコンピューティングリソースに関連付けられたセキュリティグループに設定されたインバウンドルールがある。

RDS データベースに接続されているコンピューティングリソースを表示するには

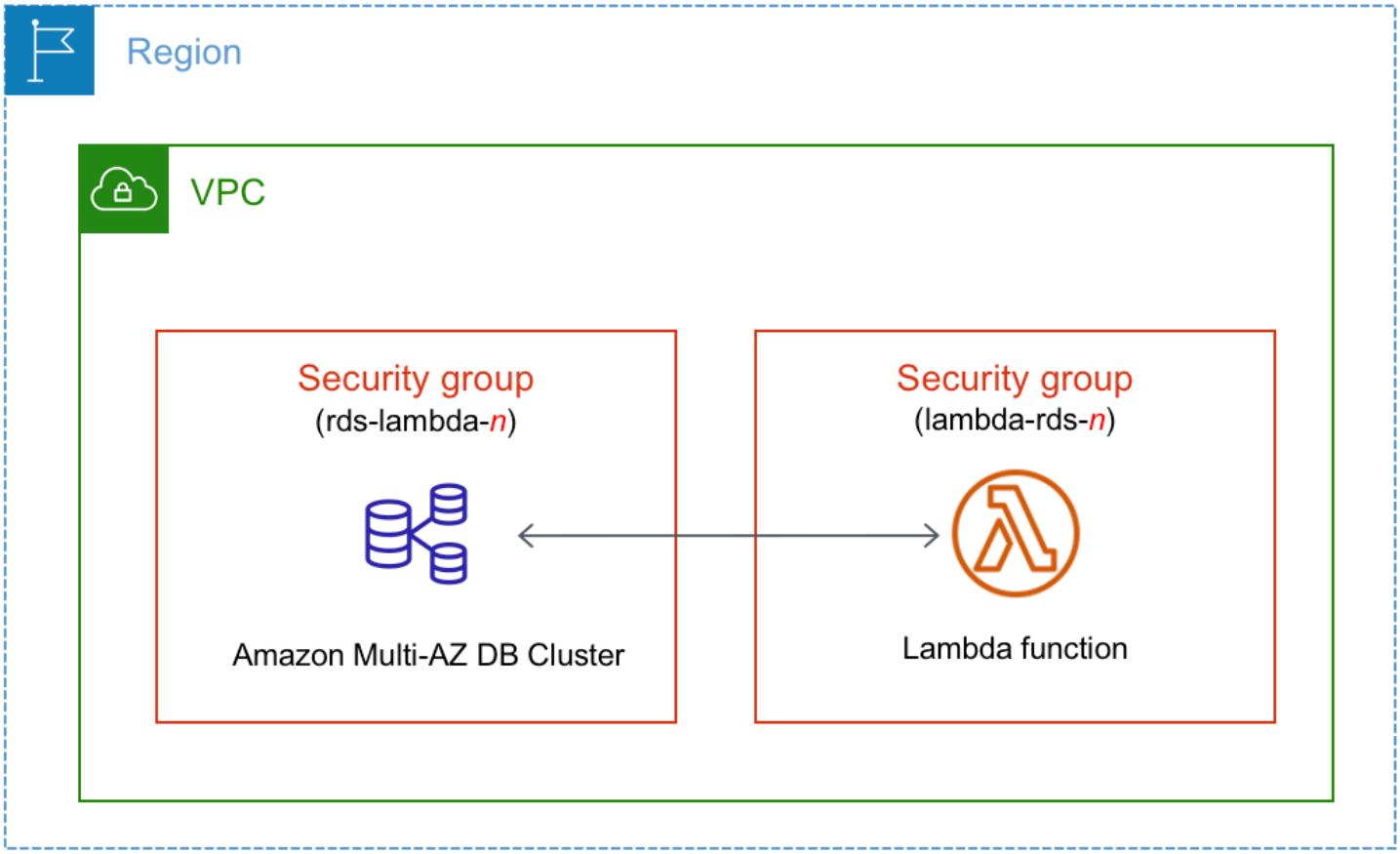
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、RDS データベース の名前を選択します。
3. [Connectivity & security] (接続とセキュリティ) タブの [Connected compute resources] (接続されたコンピューティングリソース) にコンピューティングリソースが表示されます。



Lambda 関数とマルチ AZ DB クラスターを自動的に接続する

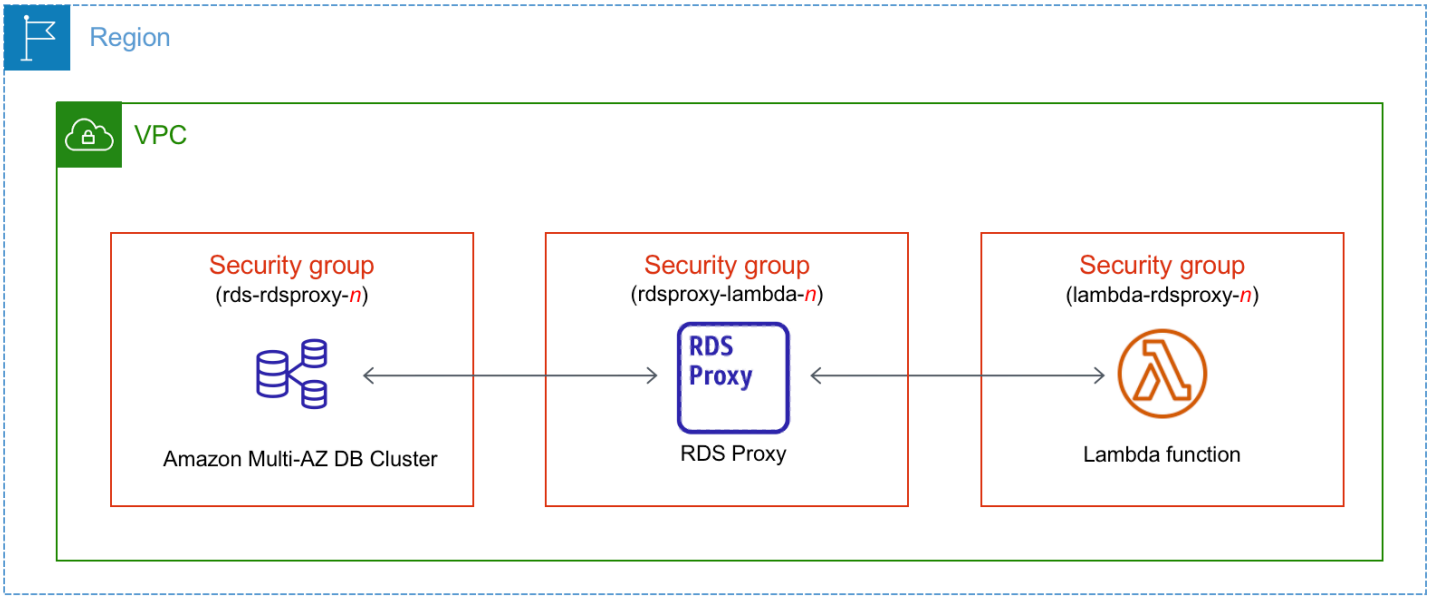
RDS コンソールを使用して、Lambda 関数とマルチ AZ DB クラスターとの接続を簡単に設定できます。RDS コンソールを使用して、Lambda 関数とマルチ AZ DB クラスターとの接続を簡単に設定できます。多くの場合、マルチ AZ DB クラスターは、VPC 内のプライベートサブネットにあります。アプリケーションで Lambda 関数を使用すると、プライベートマルチ AZ DB クラスターにアクセスできます。

次の画像は、マルチ AZ DB クラスターと Lambda 関数の間の直接接続を示しています。



Lambda 関数とデータベース間の RDS プロキシ経由の接続を設定して、データベースのパフォーマンスと耐障害性を改善できます。多くの場合、Lambda 関数は短いデータベース接続を頻繁に行い、RDS プロキシが提供する接続プールを使用することで利点を得られます。データベース認証情報を Lambda アプリケーションコードで管理する代わりに、Lambda 関数に設定済みの IAM 認証を利用できます。詳細については、「Amazon RDS Proxy」を参照してください。

コンソールを使用して、接続用のプロキシを自動的に作成できます。既存のプロキシを選択することもできます。コンソールはプロキシセキュリティグループを更新して、データベースと Lambda 関数からの接続を許可します。データベースの認証情報を入力するか、データベースへのアクセスに必要な Secrets Manager シークレットを選択できます。



トピック

- Lambda 関数との自動接続の概要
- Lambda 関数とマルチ AZ DB クラスターを自動的に接続する
- 接続中のコンピューティングリソースを表示する

Lambda 関数との自動接続の概要

Lambda 関数とマルチ AZ DB クラスター間の自動接続をセットアップすると、Amazon RDS は、Lambda 関数と DB クラスター の VPC セキュリティグループを設定します。

Lambda 関数とマルチ AZ DB クラスターを接続するための要件は次のとおりです。

- Lambda 関数は、マルチ AZ DB クラスターと同じ VPC に存在する必要があります。

  同じ VPC に Lambda 関数が存在しない場合、コンソールには Lambda 関数を作成するためのリンクが表示されます。
- 接続を設定するユーザーには、以下の Amazon RDS、Amazon EC2、Lambda、Secrets Manager、および IAM 操作を実行するアクセス許可が必要です。
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBInstances`
    - `rds:DescribeDBProxies`

    - rds:ModifyDBInstance
    - rds:ModifyDBProxy
    - rds:RegisterProxyTargets
- Amazon EC2
    - ec2:AuthorizeSecurityGroupEgress
    - ec2:AuthorizeSecurityGroupIngress
    - ec2:CreateSecurityGroup
    - ec2:DeleteSecurityGroup
    - ec2:DescribeSecurityGroups
    - ec2:RevokeSecurityGroupEgress
    - ec2:RevokeSecurityGroupIngress
- Lambda
    - lambda:CreateFunctions
    - lambda:ListFunctions
    - lambda:UpdateFunctionConfiguration
- Secrets Manager
    - sercetsmanager:CreateSecret
    - secretsmanager:DescribeSecret
- IAM
    - iam:AttachPolicy
    - iam:CreateRole
    - iam:CreatePolicy
- AWS KMS
    - kms:describeKey

Lambda 関数とマルチ AZ DB クラスター間の接続をセットアップすると、Amazon RDS は、Lambda 関数とマルチ AZ DB クラスター の VPC セキュリティグループを設定します。RDS プロキシを使用する場合、Amazon RDS はプロキシの VPC セキュリティグループも設定します。Amazon RDS は、次の表で説明されているように、マルチ AZ DB クラスター、Lambda 関数、およびプロキシに関連付けられたセキュリティグループの現在の設定に従って動作します。

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
すべてのリソースのセキュリティグループは正しい命名パターンに従い、適切なインバウンドルールとアウトバウンドルールを持っているため、Amazon RDS は何もしません。	マルチ AZ DB クラスターに関連付けられ、パターン `rds-lambda-` *n* (*n* は数字) に一致する名前を持つのセキュリティグループが 1 つ以上あります (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* (*n* は数字) に一致します。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、マルチ AZ DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つのみ存在します。	プロキシに関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `rdsproxy-lambda-` *n* (*n* は数字) に一致します。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数とマルチ AZ DB クラスターの VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。
次の条件のいずれかが適用されます。 • マルチ AZ DB クラスターに関連付けられ、パター	次の条件のいずれかが適用されます。 • Lambda 関数に関連付けられ、パターン `lambda-`	次の条件のいずれかが適用されます。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
ン rds-lambda- *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの TargetHealth が AVAILABLE の場合)。 • マルチ AZ DB クラスターに関連付けられ、パターン rds-lambda- *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの TargetHealth が AVAILABLE の場合)。ただし、Amazon RDS は、これらのセキュリティグループを Lambda 関数との接続に使用できません。 Amazon RDS は、Lambda 関数またはプロキシの VPC	rds- *n* または lambda-rdsproxy- *n* に一致する名前のセキュリティグループはありません。 • Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン lambda-rds- *n* または lambda-rdsproxy- *n* に一致します。ただし、Amazon RDS は、これらのセキュリティグループをルチ AZ DB クラスターとの接続に使用できません。 Amazon RDS は、マルチ AZ DB クラスターまたはプロキシの VPC セキュリティグループをソースとするアウトバウンドルールが 1 つも	する名前のセキュリティグループはありません。 • プロキシに関連付けられ、パターン rdsproxy-lambda- *n* に一致する名前のセキュリティグループが 1 つ以上あります。ただし、Amazon RDS は、マルチ AZ DB クラスターまたは Lambda 関数との接続にこれらのセキュリティグループを使用できません。 Amazon RDS は、マルチ AZ DB クラスターおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキ	

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。変更の例としては、ルールの追加や、既存ルールのポート変更などがあります。	存在しない場合、セキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	ユリティグループを使用できません。	

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
マルチ AZ DB クラスターに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。 ただし、Amazon RDS は、これらのセキュリティグループをルチ AZ DB クラスターとの接続に使用できません。Amazon RDS は、マルチ AZA DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。 ただし、Amazon RDS は、マルチ AZ DB クラスターまたは Lambda 関数との接続にこれらのセキュリティグループを使用できません。Amazon RDS は、マルチ AZ DB クラスターおよび Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
マルチ AZ DB クラスターに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つのみ存在します。	接続用の有効な Lambda セキュリティグループが存在しますが、Lambda 関数に関連付けられていません。このセキュリティグループには、パターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致する名前が付いています。このセキュリティグループには変更が加えられていません。マルチ AZ DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つだけあります。	接続に有効なプロキシセキュリティグループは存在しますが、プロキシに関連付けられていません。このセキュリティグループには、パターン `rdsproxy-lambda-` *n* に一致する名前が付いています。このセキュリティグループには変更が加えられていません。マルチ AZ DB クラスターと Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。	RDS action: associate Lambda security group

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
次の条件のいずれかが適用されます。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。 • マルチ AZ DB クラスターに関連付けられ、パターン `rds-lambda-` *n* に一致する名前を持つセキュリティグループはありません (または関連するプロキシの `TargetHealth` が `AVAILABLE` の場合)。ただし、Amazon RDS は、これらのセキュリティグルー	Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、マルチ AZ DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つだけ存在します。	プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。 パターンに一致するセキュリティグループは変更されていません。このセキュリティグループには、マルチ AZ DB クラスターと Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
プを Lambda 関数またはプロキシとの接続に使用できません。 Amazon RDS は、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールが 1 つも存在しないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。			

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
マルチ AZ DB クラスターに関連付けられ、パターン `rds-rdsproxy-` *n* (*n* は数字) に一致する名前を持つのセキュリティグループが 1 つ以上あります。	次の条件のいずれかが適用されます。 • Lambda 関数に関連付けられ、パターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致する名前のセキュリティグループはありません。 • Lambda 関数に関連付けられたセキュリティグループが 1 つ以上あり、その名前はパターン `lambda-rds-` *n* または `lambda-rdsproxy-` *n* に一致します。ただし、Amazon RDS は、これらのセキュリティグループをルチ AZ DB クラスターとの接続に使用できません。	次の条件のいずれかが適用されます。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループはありません。 • プロキシに関連付けられ、パターン `rdsproxy-lambda-` *n* に一致する名前のセキュリティグループが 1 つ以上あります。ただし、Amazon RDS は、マルチ AZ DB クラスターまたは Lambda 関数との接続にこれらのセキュリティグループを使用できません。 Amazon RDS は、マルチ AZ DB クラスターおよび Lambda 関数の VPC セキュリティグループに	RDS action: create new security groups

現在の RDS セキュリティグループの設定	現在の Lambda セキュリティグループ設定	現在のプロキシセキュリティグループ設定	RDS アクション
	Amazon RDS は、マルチ AZA DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールが 1 つもないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	関するインバウンドルールとアウトバウンドルールがないセキュリティグループを使用できません。また、Amazon RDS は、変更されたセキュリティグループを使用できません。	

RDS アクション: 新しいセキュリティグループを作成する

Amazon RDS は以下のアクションを実行します。

- パターン `rds-lambda-`*n* に一致する新しいセキュリティグループを作成します。このセキュリティグループには、Lambda 関数またはプロキシの VPC セキュリティグループをソースとするインバウンドルールがあります。このセキュリティグループはマルチ AZ DB クラスターに関連付けられていて、これにより、関数またはプロキシがマルチ AZ DB クラスターにアクセスするのを許可します。
- パターン `lambda-rds-`*n* に一致する新しいセキュリティグループを作成します。このセキュリティグループには、マルチ AZ DB クラスターまたはプロキシの VPC セキュリティグループをデスティネーションとするアウトバウンドルールがあります。このセキュリティグループは Lambda 関数に関連付けられ、Lambda 関数がマルチ AZ DB クラスターにトラフィックを送信するか、プロキシ経由でトラフィックを送信することを許可します。
- パターン `rdsproxy-lambda-`*n* に一致する新しいセキュリティグループを作成します。このセキュリティグループには、マルチ AZ DB クラスターと Lambda 関数の VPC セキュリティグループに関するインバウンドルールとアウトバウンドルールがあります。

RDS アクション: Lambda セキュリティグループを関連付ける

Amazon RDS は、有効な既存の Lambda セキュリティグループを Lambda 関数に関連付けます。このセキュリティグループは、関数がマルチ AZ DB クラスター にトラフィックを送信するか、プロキシ経由でトラフィックを送信することを許可します。

Lambda 関数とマルチ AZ DB クラスターを自動的に接続する

Amazon RDS コンソールを使用して、Lambda 関数をマルチ AZ DB クラスターに自動的に接続することができます。これにより、これらのリソース間の接続を設定するプロセスが簡単になります。

RDS プロキシを使用して、接続にプロキシを含めることもできます。Lambda 関数は短いデータベース接続を頻繁に行うため、RDS プロキシが提供する接続プールを使用することで利点を得られます。Lambda アプリケーションコードでデータベース認証情報を管理する代わりに、Lambda 関数用に設定済みの IAM 認証を使用することもできます。

[Lambda 接続の設定] ページを使用して、既存のマルチ AZ DB クラスターを新規および既存の Lambda 関数に接続できます。セットアッププロセスでは、必要なセキュリティグループが自動的にセットアップされます。

Lambda 関数とマルチ AZ DB クラスターの間の接続を設定する前に、次のことを確認してください。

- Lambda 関数とマルチ AZ DB クラスターが同じ VPC にあります。
- ユーザーアカウントに適切なアクセス許可があります。要件の詳細については、「Lambda 関数との自動接続の概要」を参照してください。

接続の設定後にセキュリティグループを変更すると、Lambda 関数とマルチ AZ DB クラスターとの接続に影響する可能性があります。

> **Note**
>
> AWS マネジメントコンソール でのみ、マルチ AZ DB クラスターと Lambda 関数の間の接続を自動的に設定できます。Lambda 関数を接続するには、マルチ AZ DB クラスターのすべてのインスタンスが使用可能状態である必要があります。

Lambda 関数とマルチ AZ DB クラスターを自動的に接続するには

<result>

設定を確認した後、Amazon RDS は、Lambda 関数、RDS プロキシ (プロキシを使用した場合)、およびマルチ AZ DB クラスターの接続プロセスを開始します。コンソールに [接続の詳細] ダイアログボックスが表示され、リソース間の接続を許可するセキュリティグループの変更が一覧表示されます。
</result>

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、Lambda 関数に接続するマルチ AZ DB クラスターを選択します。
3. [アクション] として、[Lambda 接続の設定] を選択します。
4. [Lambda 接続の設定] ページの [Lambda 関数の選択] で、次のいずれかを実行します。
   - マルチ AZ DB クラスターと同じ VPC に既存の Lambda 関数がある場合は、[既存の関数を選択] を選択し、関数を選択します。
   - 同じ VPC に Lambda 関数がない場合は、[新しい関数を作成] を選択し、[関数名] を入力します。デフォルトのランタイムは Nodejs.18 に設定されています。接続設定が完了した後、Lambda コンソールで新しい Lambda 関数の設定を変更できます。
5. (オプション) [RDS プロキシ] で、[RDS プロキシを使用して接続] を選択し、次のいずれかを実行します。
   - 使用する既存のプロキシがある場合は、[既存のプロキシを選択] を選択し、プロキシを選択します。
   - プロキシがなく、Amazon RDS にプロキシを自動的に作成させる場合は、[新しいプロキシの作成] を選択します。次に、[データベース認証情報] として、次のいずれかを実行します。

     a. [データベースのユーザー名とパスワード] を選択し、マルチ AZ DB クラスターの [ユーザー名] と [パスワード] を入力します。

     b. [Secrets Manager シークレット] を選択します。次に、[シークレットを選択] で、AWS Secrets Manager シークレットを選択します。Secrets Manager シークレットがない場合は、[新しい Secrets Manager シークレットを作成] を選択して、新しいシークレットを作成します。シークレットを作成した後、[シークレットを選択] で、新しいシークレットを選択します。

     新しいプロキシを作成した後、[既存のプロキシを選択] を選択し、プロキシを選択します。プロキシが接続可能になるまでに時間がかかる場合があることに注意してください。

6. (オプション) [接続の概要] を展開し、強調表示されているリソースの最新情報を確認します。
7. [設定] を選択します。

接続中のコンピューティングリソースを表示する

AWS マネジメントコンソール を使用して、マルチ AZ DB クラスターに接続されているコンピューティングリソースを表示できます。表示されるリソースには、Amazon RDS が自動的に設定したコンピューティングリソース接続が含まれます。

一覧表示されるコンピューティングリソースには、マルチ AZ DB クラスターに手動で接続されたリソースは含まれていません。例えば、クラスターに関連付けられた VPC セキュリティグループにルールを追加することで、コンピューティングリソースがマルチ AZ DB クラスター に手動でアクセスするのを許可できます。

コンソールに Lambda 関数を一覧表示するには、以下の条件が適用される必要があります。

- コンピューティングリソースに関連付けられているセキュリティグループの名前がパターン `lambda-rds-`*n* または `lambda-rdsproxy-`*n* (*n* は数字) と一致します。
- コンピューティングリソースに関連付けられているセキュリティグループに、マルチ AZ DB クラスターまたは該当するプロキシが使用するポートにポート範囲が設定されたアウトバウンドルールがあります。アウトバウンドルールのデスティネーションは、マルチ AZ DB クラスターまたは関連するプロキシに関連付けられているセキュリティグループに設定されている必要があります。
- データベースに関連付けられたプロキシにアタッチされたセキュリティグループの名前は、パターン `rds-rdsproxy-`*n* (*n* は数字) と一致します。
- 関数に関連付けられているセキュリティグループには、マルチ AZ DB クラスターまたは該当するプロキシが使用するポートにポート範囲が設定されたアウトバウンドルールがあります。デスティネーションは、マルチ AZ DB クラスターまたは関連するプロキシに関連付けられているセキュリティグループに設定されている必要があります。

マルチ AZ DB クラスターに自動的に接続されたコンピューティングリソースを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、マルチ AZ DB クラスターを選択します。
3. [接続とセキュリティ] タブの [接続されたコンピューティングリソース] にコンピューティングリソースが表示されます。

Amazon RDS のマルチ AZ DB クラスターの変更

マルチ AZ DB クラスターには、3 つの別々のアベイラビリティーゾーンに 1 つのライターDB インスタンスと 2 つのリーダー DB インスタンスがあります。マルチ AZ DB クラスターは、マルチ AZ 配置と比較して、高可用性、読み取りワークロードの容量の増加、および低レイテンシーを提供します。マルチ AZ DBクラスターの詳細ついては、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。

マルチ AZ DB クラスターを変更して、設定を変更することができます。スナップショットの作成など、マルチ AZ DB クラスターで操作を実行することもできます。

> **Important**
>
> マルチ AZ DB クラスター内の DB インスタンスは変更できません。変更はすべて DB クラスターレベルで行う必要があります。マルチ AZ DB クラスター内の DB インスタンスで実行できるのは、再起動のみです。

マルチ AZ DB クラスターは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して変更できます。

コンソール

マルチ DB クラスターを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「データベース」 を選択し、変更したいマルチ AZ DB クラスターを選択します。
3. [Modify] を選択します。[DB クラスターの変更] ページが表示されます。
4. 必要に応じて任意の設定を変更してください。各設定の詳細については、「マルチ AZ DB クラスターの変更の設定」を参照してください。
5. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
6. (省略可能) 変更をすぐに適用するには、[すぐに適用] を選択します。このオプションを選択すると、ダウンタイムを発生させる場合があります。(詳しくは、「今すぐ変更を適用する」を参照してください。)
7. 確認ページで、変更内容を確認します。正しい場合は、「DB クラスターの変更」を選択して変更を保存します。

または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

AWS CLI

AWS CLIを使用してマルチAZ DBクラスターを変更するには、modify-db-cluster コマンドを呼び出します。DB クラスター識別子と、変更したいオプションの値を指定します。各オプションの詳細については、「マルチ AZ DB クラスターの変更の設定」を参照してください。

Example

次のコードでは、バックアップ保存期間を 1 週間 (7 日間) に設定して、my-multi-az-dbcluster を変更します。このコードは、--deletion-protection を使用して削除保護を有効にします。--no-deletion-protectionを使用して、削除保護を無効にするには、変更は、--no-apply-immediately を使用して次のメンテナンスウィンドウ中に適用されます。今すぐ変更を適用するには、--apply-immediately を使用します。詳細については、「今すぐ変更を適用する」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-multi-az-dbcluster \
    --backup-retention-period 7 \
    --deletion-protection \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-multi-az-dbcluster ^
    --backup-retention-period 7 ^
    --deletion-protection ^
    --no-apply-immediately
```

RDS API

Amazon RDS API を使用して マルチAZ DB クラスターを変更するには、ModifyDBCluster オペレーションを呼び出します。DB クラスター識別子と、変更したい設定のパラメータを指定します。各パラメータの詳細については、「マルチ AZ DB クラスターの変更の設定」を参照してください。

今すぐ変更を適用する

マルチ AZ DB クラスターを変更する際に、変更内容を即時に適用することができます。変更をすぐに適用するには、AWS マネジメントコンソール で [Apply Immediately (すぐに適用)] オプションを選択します。または、AWS CLI を呼び出す際に`--apply-immediately`オプションを使用するか、`true`Amazon RDS API を使用する際に`ApplyImmediately`のパラメータを設定します。

変更の即時適用を選択しない場合、この変更は保留中の変更キューに保存されます。次のメンテナンスウィンドウ実行中に、キューのすべての保留中の変更が適用されます。変更の即時適用を選択した場合は、新しい変更と、保留中の変更キューにあるすべての変更が適用されます。

> **Important**
>
> 保留中の変更のいずれかで DB クラスターをテンポラリに使用不可にする必要がある場合 (ダウンタイム)、すぐに適用オプションを選択すると、予期しないダウンタイムが発生する可能性があります。
> 変更をすぐに適用することを選択した場合、保留中の変更も、次のメンテナンス時間中ではなく、すぐに適用されます。
> 次のメンテナンスウィンドウで保留中の変更を適用しない場合は、変更を元に戻すように DB インスタンスを変更できます。これを行うには、AWS CLI を使用し、`--apply-immediately` オプションを指定します。

変更の延期を選択した場合でも、一部のデータベース設定に対する変更は即時に適用されます。さまざまなデータベース設定が [Apply immediately (すぐに適用)] の設定とどのように相互作用するかについては、「マルチ AZ DB クラスターの変更の設定」を参照してください。

マルチ AZ DB クラスターの変更の設定

マルチ AZ DB クラスターの変更に使用できる設定の詳細については、次の表を参照してください。AWS CLIのオプションの詳細については、modify-db-clusterを参照してください。RDS API パラメータの詳細については、ModifyDBClusterを参照してください。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

ストレージ割り当て	DB クラスター内の各 DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。(詳しくは、「Amazon RDS DB インスタンスストレージ」を参照してください。)	CLI オプション:
		
		`--allocated-storage`
		
		RDS API パラメータ:
		
		`AllocatedStorage`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更時にダウンタイムは発生しません。

マイナーバージョン自動アップグレード	自動マイナーバージョンアップグレードを有効にして、DB クラスターが優先マイナー DB エンジンバージョンアップグレードを利用可能になったときに自動的に受信するようにします。Amazon RDS では、メンテナンスウィンドウでマイナーバージョンの自動アップグレードが実行されます。	CLI オプション:
		
		`--auto-minor-version-upgrade`
		
		`--no-auto-minor-version-upgrade`
		
		RDS API パラメータ:
		
		`AutoMinorVersionUpgrade`
			変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更中に、ダウンタイムが発生します。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

バックアップの保存期間	DB クラスターの自動バックアップを保持する日数。この値はゼロより大きくなければなりません。
	
	詳細については、「バックアップの概要」を参照してください。
		CLI オプション:
		
		`--backup-retention-period`
		
		RDS API パラメータ:
		
		`BackupRetentionPeriod`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更時にダウンタイムは発生しません。

バックアップ期間:	Amazon RDS が DB クラスターのバックアップを自動的に作成する期間。データベースのバックアップを保持する期間を指定しない場合は、デフォルトの「指定なし」 を使用します。
	
	詳細については、「バックアップの概要」を参照してください。
		CLI オプション:
		
		`--preferred-backup-window`
		
		RDS API パラメータ:
		
		`PreferredBackupWindow`
			変更は、可能な限り早く非同期的に適用されます。	この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

認証局	DB クラスターによって使用されるサーバー証明書の認定機関 (CA)。
	
	詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
		CLI オプション:
		
		`--ca-certificate-identifier`
		
		RDS API パラメータ:
		
		`CACertificateIdentifier`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				ダウンタイムは、DB エンジンが再起動なしのローテーションをサポートしていない場合にのみ発生します。describe-db-engine-versions AWS CLI コマンドを使用すると、DB エンジンが再起動なしのローテーションをサポートしているかどうかを判断できます。

Copy tags to snapshot	このオプションは、DB スナップショットの作成時に、DB クラスターの任意のタグを、そのスナップショットにコピーします。
	
	(詳しくは、「Amazon RDS リソースのタグ付け」を参照してください。)
		CLI オプション:
		
		`-copy-tags-to-snapshot`
		
		`-no-copy-tags-to-snapshot`
		
		RDS API パラメータ:
		
		`CopyTagsToSnapshot`
			変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
データベース認証	マルチ AZ DB クラスターの場合、パスワード認証のみがサポートされています。	パスワード認証がデフォルトであるため、なし。	変更をすぐに適用するように選択した場合は、直ちに適用されます。 変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。	この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

DB cluster identifier (DB クラスター識別子	DB クラスター識別子。この値は小文字で保存されます。
	
	DB クラスター識別子を変更すると、DB クラスターエンドポイントが変更されます。DB クラスター内の識別子とエンドポイントも変更されます。新しい DB クラスタークラスターの名前は一意である必要があります。最大長は 63 文字です。
	
	DB クラスター内の DB インスタンスの名前が、DB クラスターの新しい名前に対応するように変更されます。新しい DB インスタンスと既存の DB インスタンスを同じ名前にすることはできません。例えば、DB
		CLI オプション:
		
		`--new-db-cluster-identifier`
		
		RDS API パラメータ:
		
		`NewDBClusterIdentifier`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
	クラスターの名前を maz に変更すると、DB インスタンス名は maz-instance-1 に変更されることがあります。この場合、maz-instance-1 という名前の既存の DB インスタンスは存在できません。 詳細については、「Amazon RDS のマルチ AZ DB クラスターの名前の変更」を参照してください。			

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

DB クラスターインスタンスクラス	マルチ AZ DB クラスター内の各 DB インスタンス (`db.r6gd.xlarge` など) の計算容量とメモリ容量。
	
	可能であれば、一般的なクエリの作業セットをメモリに保持できる十分な大きさの DB インスタンスクラスを選択します。作業セットがメモリに保持されていると、システムによるディスクへの書き込みが回避され、これによりパフォーマンスが向上します。
	
	詳細については、「the section called “マルチ AZ DB クラスターで利用できるインスタンスクラス”」を参照してください。
		CLI オプション:
		
		`--db-cluster-instance-class`
		
		RDS API パラメータ:
		
		`DBClusterInstanceClass`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更中に、ダウンタイムが発生します。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

DB クラスターのパラメータグループ	DB クラスターに関連付ける DB クラスターのパラメータグループ。
	
	(詳しくは、「マルチ AZ DB クラスターのパラメータグループ」を参照してください。)
		CLI オプション:
		
		`--db-cluster-parameter-group-name`
		
		RDS API パラメータ:
		
		`DBClusterParameterGroupName`
			パラメータグループの変更は直ちに行われます。	この変更時にダウンタイムは発生しません。パラメータグループを変更すると、一部のパラメータの変更は、再起動なしで マルチ AZ DB クラスター内の DB インスタンスに即時に適用されます。他のパラメータへの変更は、DB インスタンスの再起動後にのみ適用されます。

DB エンジンバージョン	使用するデータベースエンジンのバージョン。	CLI オプション:
		
		`--engine-version`
		
		RDS API パラメータ:
		
		`EngineVersion`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更中に、ダウンタイムが発生します。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

削除保護	削除保護を有効にして、DB クラスターが削除されないようにします。
	
	(詳しくは、「DB インスタンスを削除する」を参照してください。)
		CLI オプション:
		
		`--deletion-protection`
		
		`--no-deletion-protection`
		
		RDS API パラメータ:
		
		`DeletionProtection`
			変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。

メンテナンスウィンドウ	DB クラスターへの変更保留が適用される 30 分単位のウィンドウ。期間が重要ではない場合は、「No Preference」 を選択します。
	
	詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。
		CLI オプション:
		
		`--preferred-maintenance-window`
		
		RDS API パラメータ:
		
		`PreferredMaintenanceWindow`
			変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	ダウンタイムを引き起こす保留中のアクションが 1 つ以上あり、現在の時刻を含むようにメンテナンス時間を変更した場合、それらの保留中のアクションはすぐに適用され、ダウンタイムが発生します。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

でマスター認証情報を管理する AWS Secrets Manager	[Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択して、Secrets Manager でユーザーのパスワードをシークレットに管理します。
	
	オプションで、シークレットを保護するために使用する KMS キーを選択します。お客様のアカウントの KMS キーから選択するか、別のアカウントからキーを入力します。
	
	RDS によって既に DB クラスターのマスターユーザーのパスワードを管理している場合は、[Rotate secret immediately] (すぐにシークレットをローテーションする) を選
		CLI オプション:
		
		`--manage-master-user-password | --no-manage-master-user-password`
		
		`--master-user-secret-kms-key-id`
		
		`--rotate-master-user-password | --no-rotate-master-user-password`
		
		RDS API パラメータ:
		
		`ManageMasterUserPassword`
		
		`MasterUserSecretKmsKeyId`
		
		`RotateMasterUserPassword`
			マスターユーザーパスワードの自動管理をオンまたはオフにすると、すぐに変更が行われます。この変更は、[Apply immediately] (すぐに適用) の設定を無視します。
			
			マスターユーザーのパスワードをローテーションする場合は、変更がすぐに適用されるように指定する必要があります。
				この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
	択してマスターユーザーパスワードをすぐにローテーションできます。 詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。			

新しいマスターパスワード	マスターユーザーアカウントのパスワード。	CLI オプション:
		
		`--master-user-password`
		
		RDS API パラメータ:
		
		`MasterUserPassword`
			変更は、可能な限り早く非同期的に適用されます。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。

プロビジョンド IOPS	初めに DB クラスターに割り当てられるプロビジョンド IOPS (1 秒あたりの入力/出力操作数) の合計です。	CLI オプション:
		
		`--iops`
		
		RDS API パラメータ:
		
		`Iops`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
パブリックアクセス	パブリック IP アドレスを DB クラスターに付与する場合は パブリックアクセス可能を選択します。これにより、DB インスタンスは Virtual Private Cloud (VPC) 外からアクセス可能になります。パブリックにアクセス可能となるよう、DB クラスターは、VPC のパブリックサブネット内にある必要があります。 VPC 内からのみ DB クラスターにアクセス可能にするには、「パブリックアクセス不可」 です。 (詳しくは、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。)	DB クラスターの変更時には使用できません。	変更はただちに発生します。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
	VPC の外部から DB クラスターに接続するには、DB クラスターがパブリックにアクセスできる必要があります。また、DB クラスターのセキュリティグループのインバウンドルールを使用してアクセスを許可し、その他の要件を満たしている必要があります。(詳しくは、「Amazon RDS DB インスタンスに接続できない」を参照してください。) DB クラスターがパブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続またはDirect Connect接続を使用してプライベートネットワークからアクセスすることもできます。			

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意
	詳細については、「ネットワーク間のトラフィックのプライバシー」を参照してください。			

ストレージタイプ	DB クラスターのストレージタイプ。
	
	汎用 SSD (gp3)、プロビジョンド IOPS (io1)、およびプロビジョンド IOPS SSD (io2) ストレージのみがサポートされています。
	
	詳細については、「Amazon RDS ストレージタイプ」を参照してください。
		CLI オプション:
		
		`--storage-type`
		
		RDS API パラメータ:
		
		`StorageType`
			変更をすぐに適用するように選択した場合は、直ちに適用されます。
			
			変更をすぐに適用ように選択しない場合は、次のメンテナンス期間中に適用されます。
				この変更時にダウンタイムは発生しません。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ	変更を行った場合	ダウンタイムに関する注意

VPC セキュリティグループ	DB クラスターに関連付けるセキュリティグループ。
	
	(詳しくは、「VPC セキュリティグループの概要」を参照してください。)
		CLI オプション:
		
		`--vpc-security-group-ids`
		
		RDS API パラメータ:
		
		`VpcSecurityGroupIds`
			変更は、可能な限り早く非同期的に適用されます。この設定は、[Apply immediately (すぐに適用)] 設定を無視します。	この変更時にダウンタイムは発生しません。

マルチ AZ DB クラスターの変更時に適用されない設定

AWS CLI コマンド modify-db-cluster および RDS API オペレーション ModifyDBCluster の次の設定は、マルチAZ DBクラスターには適用されません。

コンソールでマルチ AZ DB クラスターの設定を変更することもできません。

AWS CLI の設定	RDS API の設定
`--backtrack-window`	`BacktrackWindow`
`--cloudwatch-logs-export-configuration`	`CloudwatchLogsExportConfiguration`
`--copy-tags-to-snapshot | --no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--db-instance-parameter-group-name`	`DBInstanceParameterGroupName`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`

AWS CLI の設定	RDS API の設定
--enable-global-write-forwarding | --no-enable-global-write-forwarding	EnableGlobalWriteForwarding
--enable-http-endpoint | --no-enable-http-endpoint	EnableHttpEndpoint
--option-group-name	OptionGroupName
--port	Port
--scaling-configuration	ScalingConfiguration
--storage-type	StorageType

Amazon RDS のマルチ AZ DB クラスターのエンジンバージョンのアップグレード

Amazon RDS では、マルチ AZ DB クラスターを最新の状態に維持するため、サポートされている各エンジンの新しいバージョンを提供します。このトピックでは、マルチ AZ DB クラスターを新しいバージョンにアップグレードするプロセスについて説明します。

マルチ AZ DB クラスターのアップグレードには、互換性のある新しいエンジンバージョンを選択し、潜在的なダウンタイムを計画する必要があります。このプロセスでは、マルチ AZ アーキテクチャのフェイルオーバー機能を活用することで、中断を最小限に抑えます。ベストプラクティスには、トラフィックの少ない期間のアップグレードの実行、非本番環境でのテスト、新しいバージョンとのアプリケーションの互換性の検証などがあります。

トピック

- マイナーバージョンのアップグレード
- メジャーバージョンのアップグレード
- マルチ AZ DB クラスターのアップグレード
- マルチ AZ DB クラスターリードレプリカのアップグレード
- イベントによるマルチ AZ DB クラスターのアップグレードのモニタリング

マイナーバージョンのアップグレード

マイナーバージョンアップグレードには、既存のアプリケーションとの下位互換性がある変更のみが含まれます。マイナーバージョンアップグレードを開始すると、Amazon RDS は最初にリーダー DB インスタンスを一度に 1 つずつアップグレードします。その後、リーダー DB インスタンスの 1 つが新しいライター DB インスタンスに切り替わります。次に、Amazon RDS は、古いライターインスタンスをアップグレードします (リーダーインスタンスになります) 。

アップグレード中のダウンタイムは、リーダー DB インスタンスの 1 つが新しいライター DB インスタンスになるのにかかる時間に制限されます。このダウンタイムは、自動フェイルオーバーのように切り替わります。詳細については、「the section called “マルチ AZ DB クラスターをフェイルオーバーする”」を参照してください。マルチ AZ DB クラスターのレプリカの遅延は、ダウンタイムに影響する可能性があることに注意してください。詳細については、「the section called “レプリカの遅延とマルチ AZ DB クラスター”」を参照してください。

RDS for PostgreSQL マルチ AZ DB クラスターのリードレプリカの場合、Amazon RDS はクラスターのメンバーであるインスタンスを一度に 1 つずつアップグレードします。リーダークラス

ターロールとライタークラスターのロールは、アップグレード中に切り替えられません。したがって、Amazon RDS がクラスターライターインスタンスをアップグレードしている間に、DB クラスターでダウンタイムが発生する可能性があります。

> **Note**
>
> マルチ AZ DB クラスターのマイナーバージョンアップグレードのダウンタイムは、通常 35 秒です。RDS Proxy と併用すると、ダウンタイムをさらに 1 秒以下に短縮できます。詳細については、「Amazon RDS Proxy 」を参照してください。または、ProxySQL、PgBouncer、AWS Advanced JDBC Wrapper Driver などのオープンソースデータベースプロキシを使用することもできます。

メジャーバージョンのアップグレード

メジャーバージョンのアップグレードによって、既存のアプリケーションと互換性のない変更が導入されることがあります。

RDS for PostgreSQL マルチ AZ DB クラスターのメジャーバージョンのアップグレードを開始すると、Amazon RDS によってリーダーとライターのインスタンスが同時にアップグレードされます。したがって、アップグレードが完了するまで DB クラスターを使用できない場合があります。

RDS for MySQL マルチ AZ DB クラスターのメジャーバージョンアップグレードを開始すると、Amazon RDS はクラスターメンバーインスタンスを一度に 1 つずつアップグレードするため、レプリケーションはエンジンの下位バージョンから上位バージョンの順に行われます。エンジンのバージョンは構文と機能が異なる可能性があるため、メジャーバージョンのアップグレード中に、ワークロードがソースとターゲットのエンジンバージョンの間で互換性があるかを確認することが重要です。

> **Note**
>
> マイナーバージョンアップグレードと同様に、RDS for MySQL メジャーバージョンアップグレードのダウンタイムは通常 35 秒です。RDS Proxy と併用すると、ダウンタイムをさらに 1 秒以下に短縮できます。詳細については、「Amazon RDS Proxy 」を参照してください。

マルチ AZ DB クラスターのアップグレード

マルチ AZ DB クラスターのエンジンバージョンをアップグレードするプロセスは、DB インスタンスのエンジンバージョンをアップグレードするプロセスと同じです。手順については、「the section called “エンジンバージョンのアップグレード”」を参照してください。唯一の違いは、AWS Command Line Interface (AWS CLI) を使用する場合、modify-db-cluster コマンドを実行して、`--db-cluster-identifier` パラメータ (および `--allow-major-version-upgrade` パラメータ) を指定するという点です。

メジャーバージョンおよびマイナーバージョンのアップグレードの詳細については、以下の DB エンジンに関するドキュメントを参照してください。

- the section called “PostgreSQL DB エンジンのアップグレード”
- the section called “MySQL DB エンジンのアップグレード”

マルチ AZ DB クラスターリードレプリカのアップグレード

Amazon RDS は、マルチ AZ DB クラスターのリードレプリカを自動的にアップグレードしません。マイナーバージョンアップグレードの場合、最初にすべてのリードレプリカを手動でアップグレードしてからクラスターをアップグレードする必要があります。そうしないと、アップグレードがブロックされます。クラスターのメジャーバージョンを実行すると、すべてのリードレプリカのレプリケーション状態が終了に変わります。アップグレードの完了後、リードレプリカを削除し、再作成する必要があります。詳細については、「the section called “リードレプリケーションのモニタリング”」を参照してください。

イベントによるマルチ AZ DB クラスターのアップグレードのモニタリング

マルチ AZ DB クラスターのエンジンバージョンをアップグレードすると、Amazon RDS はプロセスの各フェーズで特定のイベントを発行します。アップグレードの進行状況を追跡するには、これらのイベントを表示またはサブスクライブします。

RDS イベントの詳細については、「Amazon RDS イベントのモニタリング」を参照してください。

エンジンのアップグレード中に発生する特定の Amazon RDS イベントの詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

Amazon RDS のマルチ AZ DB クラスターの名前の変更

マルチ AZ DB クラスターの名前を変更するには、AWS マネジメントコンソール、AWS CLI、`modify-db-cluster` コマンド、または Amazon RDS API `ModifyDBCluster` オペレーションを使用します。マルチ AZ DB クラスターの名前を変更すると、大きな影響があります。以下は、マルチ AZ DB クラスターの名前を変更する前の考慮事項の一覧です。

- マルチ AZ DB クラスターの名前を変更すると、マルチ AZ DB クラスターのクラスターエンドポイントが変更されます。これらのエンドポイントは、マルチ AZ DB クラスターに割り当てた名前が含まれているために変更されます。古いエンドポイントから新しいエンドポイントにトラフィックをリダイレクトできます。マルチ AZ DB クラスターエンドポイントの詳細については、「Amazon RDS のマルチ AZ DB クラスターへの接続」を参照してください。
- マルチ AZ DB クラスターの名前を変更すると、マルチ AZ DB クラスターが使用していた古い DNS 名は削除されますが、数分間キャッシュされたままになる場合があります。名前を変更したマルチ AZ DB クラスターの新しい DNS 名は、約 2 分後に有効になります。名前を変更したマルチ AZ DB クラスターは、新しい名前が有効になるまで使用できません。
- クラスターの名前を変更する場合、既存のマルチ AZ DB クラスターの名前を使用することはできません。
- マルチ AZ DB クラスターの名前に関連付けられたメトリクスとイベントは、DB クラスターの名前を再利用する場合も維持されます。
- マルチ AZ DB クラスターのタグは、名前の変更にかかわらず、マルチ AZ DB クラスターに残ります。
- DB クラスターのスナップショットは、名前を変更した マルチ AZ DB クラスターに保持されます。

> **Note**
>
> マルチ AZ DB クラスターは、クラウドで実行する独立したデータベース環境です。マルチ AZ DB クラスターは、複数のデータベースをホストすることができます。データベース名の変更の詳細については、DB エンジンのドキュメントを参照してください。

名前を変更して既存のマルチ AZ DB クラスターを置き換える

マルチ AZ DB クラスターの名前を変更する最も一般的なシナリオには、DB クラスターのスナップショットからのデータの復元や、ポイントインタイムリカバリ (PITR) の実行です。マルチ AZ DB ク

ラスターの名前を変更することで、マルチ AZ DB クラスターを参照するアプリケーションコードを変更することなく、マルチ AZ DB クラスターを置き換えることができます。この場合、次のステップを完了します。

1. マルチ AZ DB クラスターに向かうトラフィックを停止します。マルチ AZ DB クラスターでデータベースにアクセスするトラフィックをリダイレクトするか、別の方法を選択してトラフィックのアクセスを回避します。
2. 既存のマルチ AZ DB クラスターの名前を変更します。
3. DB クラスターのスナップショットの復元またはポイントインタイムの復旧によって、新しいマルチ AZ DB クラスターを作成します。次に、新しいマルチ AZ DB クラスターに以前のマルチ AZ DB クラスターの名前を付けます。

古いマルチ AZ DB クラスターを削除した場合、古いマルチ AZ DB クラスターの不要な DB クラスターのスナップショットは、お客様によってすべて削除する必要があります。

コンソール

マルチ AZ DB クラスターの名前を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベースを選択します。
3. 名前を変更するマルチ AZ DB クラスターを選択します。
4. Modify を選択します。
5. [Settings] (設定) で、[DB cluster identifier] (DB クラスター識別子) に新しい名前を入力します。
6. [Continue] (続行) をクリックします。
7. 変更をすぐに反映させるには、[Apply immediately] を選択します。このオプションを選択すると、停止状態になる場合があります。詳細については、「今すぐ変更を適用する」を参照してください。
8. 確認ページで、変更内容を確認します。正しい場合は、[クラスターの変更] を選択して変更を保存します。

   または、[Back] (戻る) を選択して変更を編集するか、[Cancel] (キャンセル) を選択して変更を破棄します。

AWS CLI

マルチ AZ DB クラスターの名前を変更するには、AWS CLI コマンドの modify-db-cluster を使用します。マルチ AZ DB クラスターの新しい名前とともに、現在の `--db-cluster-identifier` 値および `--new-db-cluster-identifier` パラメータを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier DBClusterIdentifier \
    --new-db-cluster-identifier NewDBClusterIdentifier
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier DBClusterIdentifier ^
    --new-db-cluster-identifier NewDBClusterIdentifier
```

RDS API

マルチ AZ DB クラスターの名前を変更するには、以下のパラメータで、Amazon RDS API オペレーションの ModifyDBCluster を呼び出します。

- `DBClusterIdentifier` - DB クラスターの既存の名前。
- `NewDBClusterIdentifier` - DB クラスターの新しい名前。

Amazon RDS のマルチ AZ DB クラスターとリーダー DB インスタンスの再起動

通常はメンテナンスのために、マルチAZ DB クラスターを再起動する必要がある場合があります。例えば、特定の変更を行う場合や DB クラスターに関連付けられた DB クラスターのパラメータグループを変更する場合、DB クラスターを再起動します。これを行うことで、変更が有効になります。

DB クラスターが、その関連付けられた DB クラスターパラメータグループに対する最新の変更を使用していない場合、AWS マネジメントコンソール は、DB クラスターパラメータグループのステータスを 「再起動の保留中」 と表示します。パラメータグループの [再起動の保留中] のステータスにより、次回のメンテナンスウィンドウで自動的に再起動されることはありません。パラメータの最新の変更を DB クラスターに適用するには、DB クラスターを手動で再起動します。パラメータグループの詳細については、「マルチ AZ DB クラスターのパラメータグループ」を参照してください。

DB クラスターを再起動すると、データベースエンジンサービスが再起動されます。DB クラスターを再起動すると一時的に機能停止になります。その間、DB クラスターのステータスは 「rebooting」 に設定されます。

available (利用可能) 状態でない DB クラスターを再起動することはできません。データベースは、バックアップが進行中または、以前の要求による変更、メンテナンス時間のアクションなど、いくつかの理由で使用できない場合があります。

DB クラスターの再起動に要する時間は、クラッシュ回復プロセス、再起動時のデータベースのアクティビティ、および特定の DB クラスターの動作によって異なります。再起動時間を短くするには、再起動プロセス中のデータベースアクティビティをできる限り減らすことをお勧めします。データベースアクティビティを減らすと、未完了のトランザクションのロールバックアクティビティが減少します。

> **Important**
>
> マルチ AZ DB クラスターは、フェイルオーバーによる再起動をサポートしていません。マルチ AZ DB クラスターのライターインスタンスを再起動しても、その DB クラスター内のリーダー DB インスタンスには影響せず、フェイルオーバーは発生しません。リーダー DB インスタンスを再起動すると、フェイルオーバーは発生しません。マルチ AZ DB クラスターをフェイルオーバーするには、コンソールのフェイルオーバーを選択し、AWS CLIコマンド`failover-db-cluster`、または API オペレーション`FailoverDBCluster`をび出します。

コンソール

DB クラスターを再起動するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「データベース」 を選択して、再起動したい マルチ AZ DBクラスターを選択します。
3. [アクション] で、[再起動] を選択します。

   「DB クラスターを再起動」 ページが表示されます。
4. DB クラスターを再起動するには、「Reboot」 を選択します。

   または、[Cancel] (キャンセル) を選択します。

AWS CLI

AWS CLIを使用してマルチ AZ DB クラスターを再起動するには、reboot-db-clusterコマンドを呼び出します。

```
aws rds reboot-db-cluster --db-cluster-identifier mymultiazdbcluster
```

RDS API

Amazon RDS API を使用してマルチ AZ DB クラスターを再起動するには、リブートDBClusterオペレーションを呼び出します。

Amazon RDS 用のマルチ AZ DB クラスターのフェイルオーバー

マルチ AZ DB クラスター内のライター DB インスタンスで計画的な機能停止または計画外の機能停止が発生した場合、Amazon RDS は別のアベイラビリティーゾーンのリーダー DB インスタンスに自動的にフェイルオーバーします。これにより、中断を最小限に抑えながら高可用性を確保できます。フェイルオーバーは、ハードウェア障害、ネットワークの問題、または手動リクエスト中に発生する可能性があります。このトピックでは、障害の自動検出、フェイルオーバー中のイベントのシーケンス、読み取りおよび書き込みオペレーションへの影響について説明します。また、フェイルオーバー時間をモニタリングして最小限に抑えるためのベストプラクティスも提供します。

フェイルオーバーが完了するまでにかかる時間は、データベースアクティビティや、ライターDB インスタンスが使用できなくなった時点の他の条件によって異なります。フェイルオーバーの時間は通常 35 秒未満です。フェイルオーバーは、両方のリーダー DB インスタンスが、失敗したライターからの未処理のトランザクションを適用すると完了します。フェイルオーバーが完了してから、新しいアベイラビリティーゾーンが RDS コンソールに反映されるまでさらに時間がかかる場合があります。

トピック

- 自動フェイルオーバー
- マルチ AZ DB クラスターを手動でフェイルオーバーする
- マルチ AZ DB クラスターがフェイルオーバーしたかどうかの判別
- DNS 名参照用の JVM TTL の設定

自動フェイルオーバー

Amazon RDS はフェイルオーバーを自動的に処理するため、管理者が操作しなくても可能な限りすみやかにデータベース操作を再開できます。フェイルオーバーするには、ライター DB インスタンスがリーダー DB インスタンスに自動的に切り替わります。

マルチ AZ DB クラスターを手動でフェイルオーバーする

マルチ AZ DB クラスターを手動でフェイルオーバーすると、RDS は最初にプライマリ DB インスタンスを終了します。次に、内部モニタリングシステムがプライマリ DB インスタンスに異常があることを検出し、読み取り可能なレプリカ DB インスタンスを昇格させます。フェイルオーバーの時間は通常 35 秒未満です。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、マルチ AZ DB クラスタを手動でフェイルオーバーできます。

コンソール

マルチ AZ DB クラスターを手動でフェイルオーバーするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベースを選択します。
3. フェイルオーバーするマルチ AZ DB クラスターを選択します。
4. 「アクション」で、「フェイルオーバー」を選択します。

   [フェイルオーバー DB クラスター] ページが表示されます。
5. フェイルオーバーを選択して、手動フェイルオーバーを確認します。

AWS CLI

マルチ AZ DB クラスターを手動でフェイルオーバーするには、AWS CLIコマンドフェイルオーバー db-クラスタを使用します。

Example

```
aws rds failover-db-cluster --db-cluster-identifier mymultiazdbcluster
```

RDS API

マルチ AZ DB クラスターを手動でフェイルオーバーするには、Amazon RDS API FailoverDBClusterを呼び出し、DBClusterIdentifierを指定します。

マルチ AZ DB クラスターがフェイルオーバーしたかどうかの判別

マルチ AZ DB クラスターがフェイルオーバーされたかどうかを判断するには、次を実行します。

- フェイルオーバーがスタートされたことを電子メールまたはSMSで通知するようにDBイベントサブスクリプションを設定します。 イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。
- Amazon RDS コンソールまたは API オペレーションを使用して DB イベントを表示します。
- Amazon RDS コンソール、AWS CLIおよび RDS API を使用して、マルチ AZ DB クラスターの現在の状態を表示します。

フェイルオーバーへの応答、復旧時間の短縮、Amazon RDS のその他のベストプラクティスについては、「Amazon RDS のベストプラクティス」を参照してください。

DNS 名参照用の JVM TTL の設定

フェイルオーバーメカニズムでは、リーダー DB インスタンスをポイントするように DB インスタンスのドメインネームシステム (DNS) レコードが自動的に変更されます。したがって、DB インスタンスへの既存の接続の再確立が必要になります。Java 仮想マシン (JVM) 環境では、Java DNS キャッシュ機構がどのように機能するかによって、JVM 設定の再構成が必要になる場合があります。

JVM は DNS 名参照をキャッシュします。JVM がホスト名を IP アドレスに変換するとき、time-to-live (TTL) と呼ばれる指定期間 IP アドレスをキャッシュします。

AWS リソースは、ときどき変更される DNS 名を使用するため、60 秒を超えない TTL 値で JVM を設定することをお勧めします。こうすることにより、リソースの IP アドレスが変更されたときに、アプリケーションは DNS に対して再度クエリを実行することで、リソースの新しい IP アドレスを取得して使用できます。

一部の Java 設定では JVM のデフォルトの TTL が設定されるため、JVM が再起動されるまで、DNS エントリが更新されることはありません。したがって、アプリケーションがまだ実行中に AWS リソースの IP アドレスが変更された場合、JVM を手動で再起動し、キャッシュされた IP 情報が更新されるまで、そのリソースを使用することはできません。この場合、キャッシュされた IP 情報が定期的に更新されるように JVM の TTL を設定することがきわめて重要です。

> **Note**
>
> デフォルト TTL は、JVM のバージョンと、セキュリティマネージャーがインストールされているかどうかに応じて変わります。多くの JVM はデフォルト TTL を 60 秒以下にしています。このような JVM を使用しており、セキュリティマネージャーを使用していない場合、このトピックの残り部分は無視してかまいません。Oracle のセキュリティマネージャーの詳細については、Oracle ドキュメントの 「The Security Manager」を参照してください。

JVM の TTL を変更するには、`networkaddress.cache.ttl` プロパティ値を設定します。ニーズに応じて、次の方法のいずれかを使用します。

- JVM を使用するすべてのアプリケーションのプロパティ値をグローバルに設定するには、`networkaddress.cache.ttl` ファイルで `$JAVA_HOME/jre/lib/security/java.security` を設定します。

```
networkaddress.cache.ttl=60
```

- アプリケーションに対してのみプロパティをローカルに設定するには、ネットワーク接続を確立する前に、アプリケーションの初期化コードで networkaddress.cache.ttl を設定します。

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

Amazon RDS のマルチ AZ DB クラスターを使用した PostgreSQL 論理レプリケーションの設定

マルチ AZ DB クラスターとの PostgreSQL 論理レプリケーションを使用することで、データベースインスタンス全体ではなく、個々のテーブルをレプリケートおよび同期できます。論理レプリケーションでは、パブリケーションおよびサブスクリプションモデルを使用して、ソースからの変更を 1 人または複数の受信者にレプリケートします。これは、PostgreSQL の ログ先行書き込み (WAL) の変更レコードを使用することで動作します。詳細については、「the section called “論理レプリケーションの実行”」を参照してください。

マルチ AZ DB クラスターのライター DB インスタンスに新しい論理レプリケーションスロットを作成すると、そのスロットはクラスター内の各リーダー DB インスタンスに非同期でコピーされます。リーダー DB インスタンスのスロットは、ライター DB インスタンスのスロットと継続的に同期されます。

論理レプリケーションは、RDS for PostgreSQL バージョン 14.8-R2 以降および 15.3-R2 以降を実行しているマルチ AZ DB クラスターでサポートされています。

> **Note**
>
> ネイティブの PostgreSQL 論理レプリケーション機能に加えて、RDS for PostgreSQL を実行しているマルチ AZ DB クラスターは `pglogical` 拡張機能もサポートしています。

PostgreSQL 論理レプリケーションの詳細については、PostgreSQL ドキュメントの「論理レプリケーション」を参照してください。

トピック

- 前提条件
- 論理レプリケーションの設定
- 制限と推奨事項

前提条件

マルチ AZ DB クラスターの PostgreSQL 論理レプリケーションを設定するには、次の前提条件を満たす必要があります。

- ユーザーアカウントは rds_superuser グループのメンバーであり、rds_superuser 権限を持っている必要があります。詳細については、「the section called “PostgreSQL のロールとアクセス権限について”」を参照してください。
- 次の手順で説明するパラメータ値を設定できるように、マルチ AZ DB クラスターをカスタム DB クラスターパラメータグループに関連付ける必要があります。詳細については、「the section called “DB クラスターのパラメータグループ”」を参照してください。

論理レプリケーションの設定

マルチ AZ DB クラスターの論理レプリケーションを設定するには、関連する DB クラスターパラメータグループ内の特定のパラメータを有効にしてから、論理レプリケーションスロットを作成します。

> **Note**
>
> PostgreSQL バージョン 16 から、マルチ AZ DB クラスターのリーダー DB インスタンスを論理レプリケーションに使用できます。

RDS for PostgreSQL マルチ AZ DB クラスターの論理レプリケーションをセットアップするには

1. RDS for PostgreSQL マルチ AZ DB クラスターに関連付けられているカスタム DB クラスターパラメータグループを開きます。
2. [パラメータ] 検索フィールドで、rds.logical_replication 静的パラメータを探し、その値を 1 に設定します。このパラメータの変更により、WAL の生成量が増える可能性があるため、論理スロットを使用している場合にのみ有効にしてください。
3. この変更の一環として、次の DB クラスターパラメータを設定します。

   - max_wal_senders
   - max_replication_slots
   - max_connections

   予想される使用状況に応じて、次のパラメータの値を変更しなければならない場合があります。ただし、多くの場合はデフォルト値で十分です。

   - max_logical_replication_workers

- max_sync_workers_per_subscription

4. マルチ AZ DB クラスターを再起動して、パラメータ値を有効にします。手順については、「the section called “マルチ AZ DB クラスターの再起動”」を参照してください。
5. the section called “論理的なレプリケーションスロットの使用” で説明されているように、マルチ AZ DB クラスターのライター DB インスタンスに論理レプリケーションスロットを作成します。このプロセスでは、デコードプラグインを指定する必要があります。現在、RDS for PostgreSQL は、PostgreSQL に付属する test_decoding、wal2json、および pgoutput プラグインをサポートしています。

   スロットは、クラスター内の各リーダー DB インスタンスに非同期でコピーされます。
6. マルチ AZ DB クラスターのすべてのリーダー DB インスタンスのスロットの状態を確認します。そのためには、すべてのリーダー DB インスタンスの pg_replication_slots ビューを調べて、アプリケーションが論理的な変更を積極的に行っている間に confirmed_flush_lsn 状態が進行していることを確認します。

   以下のコマンドは、リーダー DB インスタンスのレプリケーション状態を調べる方法を示しています。

```
% psql -h test-postgres-instance-2.abcdefabcdef.us-west-2.rds.amazonaws.com

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D0001700
(1 row)

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D8003628
(1 row)

% psql -h test-postgres-instance-3.abcdefabcdef.us-west-2.rds.amazonaws.com

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
```

```
 logical_slot | logical   | 32/D0001700
(1 row)

postgres=> select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
  slot_name   | slot_type | confirmed_flush_lsn
--------------+-----------+---------------------
 logical_slot | logical   | 32/D8003628
(1 row)
```

レプリケーションタスクが完了したら、レプリケーションプロセスを停止し、レプリケーションスロットを削除して、論理レプリケーションをオフにします。論理レプリケーションをオフにするには、DB クラスターのパラメータグループを変更し、rds.logical_replication の値を 0 に戻します。クラスターを再起動して、パラメータの変更を有効にします。

制限と推奨事項

PostgreSQL 16 を実行するマルチ AZ DB クラスターでの論理レプリケーションの使用には、以下の制限事項と推奨事項があります。

- 論理レプリケーションスロットを作成または削除するには、ライター DB インスタンスのみを使用できます。例えば、CREATE SUBSCRIPTION コマンドでは、ホスト接続文字列にクラスターライターエンドポイントを使用する必要があります。
- テーブルの同期または再同期中にクラスターライターエンドポイントを使用する必要があります。例えば、以下のコマンドを使用して、新しく追加されたテーブルを再同期できます。

```
Postgres=>ALTER SUBSCRIPTION subscription-name CONNECTION host=writer-endpoint
Postgres=>ALTER SUBSCRIPTION subscription-name REFRESH PUBLICATION
```

- 論理レプリケーションにリーダー DB インスタンスを使用する前に、テーブルの同期が完了するのを待つ必要があります。テーブルの同期は、pg_subscription_rel カタログテーブルを使用して監視できます。テーブルの同期が完了すると、srsubstate 列が ready (r) に設定されます。
- 最初のテーブル同期が完了したら、論理レプリケーション接続にインスタンスエンドポイントを使用することをお勧めします。次のコマンドは、いずれかのリーダー DB インスタンスにレプリケーションをオフロードすることで、ライター DB インスタンスの負荷を軽減します。

```
Postgres=>ALTER SUBSCRIPTION subscription-name CONNECTION host=reader-instance-
endpoint
```

一度に複数の DB インスタンスで同じスロットを使用することはできません。2 つ以上のアプリケーションがクラスターの異なる DB インスタンスから論理的な変更をレプリケートしている場合、クラスターのフェイルオーバーやネットワークの問題により、一部の変更が失われる可能性があります。このような状況では、ホスト接続文字列で論理レプリケーションのインスタンスエンドポイントを使用できます。同じ設定を使用している他のアプリケーションには、次のエラーメッセージが表示されます。

```
replication slot slot_name is already active for PID x providing immediate feedback.
```

- pglogical 拡張機能を使用している間は、クラスターライターエンドポイントのみを使用できます。拡張機能には、テーブルの同期中に未使用の論理レプリケーションスロットが作成される可能性がある既知の制限があります。古いレプリケーションスロットによって WAL ファイルが予約され、ディスク容量の問題を引き起こす可能性があります。

Amazon RDS のマルチ AZ DB クラスターのリードレプリカの操作

DB クラスターのリードレプリカは、ソース DB インスタンスから作成する特殊なタイプのクラスターです。リードレプリカの作成後、プライマリ DB インスタンスに加えられた更新は、マルチ AZ DB クラスターのリードレプリカに非同期的にコピーされます。読み込みクエリをアプリケーションからリードレプリカにルーティングすることにより、プライマリ DB インスタンスの負荷を軽減できます。リードレプリカを使うと、単一 DB インスタンスのキャパシティ制約にとらわれることなく伸縮自在にスケールアウトし、読み込み負荷の高いデータベースワークロードに対応できます。

また、マルチ AZ DB クラスターから 1 つ以上の DB インスタンスのリードレプリカを作成できます。DB インスタンスのリードレプリカでは、余分な読み取りトラフィックをリードレプリカに送信することで、ソースマルチ AZ DB クラスターのコンピューティングまたは I/O のキャパシティを超えてスケーリングできます。現在、既存のマルチ AZ DB クラスターからマルチ AZ DB クラスターのリードレプリカを作成することはできません。

リードレプリカを使用してマルチ AZ DB クラスターに移行するか、マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成するかを選択する場合は、特定のユースケースとパフォーマンス要件を考慮してください。

リードレプリカを使用してマルチ AZ DB クラスターに移行する

: このアプローチは、ダウンタイムを最小限に抑えながらデータベースの可用性と耐久性を向上させる必要がある場合に最適です。リードレプリカを使用してマルチ AZ DB クラスターに移行することで、継続的なオペレーションとデータ整合性を確保できます。この方法は、可用性を維持し、ライブワークロードへの影響を減らすことが重要な本番環境に特に役立ちます。

マルチ AZ DB クラスターからの DB インスタンスリードレプリカの作成

: この方法は、読み取りオペレーションをスケールしたり、プライマリデータベースインスタンスから読み取りトラフィックをオフロードしたりする場合に適しています。既存のマルチ AZ DB クラスターからリードレプリカを作成することで、読み取り負荷の高いワークロードを分散させ、プライマリインスタンスの安定性に影響を与えることなくパフォーマンスを向上させることができます。

適切なアプローチの選択は、高い可用性と耐久性を確保するか、読み取りパフォーマンスをスケーリングするかによって異なります。ワークロードの特性と運用要件を評価して、情報に基づいた意思決定を行います。

トピック

- リードレプリカを使用してマルチ AZ DB クラスターに移行する
- マルチ AZ DB クラスターからの DB インスタンスリードレプリカの作成

リードレプリカを使用してマルチ AZ DB クラスターに移行する

シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイをマルチ AZ DB クラスターデプロイに少ないダウンタイムで移行するには、マルチ AZ DB クラスターリードレプリカを作成します。ソースとして、シングル AZ デプロイの DB インスタンス、またはマルチ AZ DB インスタンスデプロイのプライマリ DB インスタンスを指定します。DB インスタンスは、マルチ AZ DB クラスターへの移行時に書き込みトランザクションを処理できます。

以下は、マルチ AZ DB クラスターのリードレプリカを作成する前の考慮事項です。

- ソース DB インスタンスは、マルチ AZ DB クラスターをサポートするバージョンである必要があります。詳細については、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。
- マルチ AZ DB クラスターのリードレプリカは、ソースと同じメジャーバージョンで、同じかそれ以上のマイナーバージョンでなければなりません。
- バックアップ保持期間を 0 以外の値に設定することで、ソース DB インスタンスで自動バックアップを有効にする必要があります。
- ソース DB インスタンスに割り当てられるストレージは 100 GiB 以上でなければなりません。
- RDS for MySQL では、`gtid-mode` と `enforce_gtid_consistency` パラメータの両方は、ソース DB インスタンスの `ON` に設定する必要があります。デフォルトのパラメータグループではなく、カスタムパラメータグループを使用する必要があります。詳細については、「the section called “DB パラメータグループ”」を参照してください。
- アクティブな長時間実行トランザクションの場合、リードレプリカの作成プロセスに時間がかかることがあります。長時間実行トランザクションが完了してから、リードレプリカを作成することをお勧めします。
- マルチ AZ DB クラスターのリードレプリカのソース DB インスタンスを削除した場合、リードレプリカはスタンドアロンのマルチ AZ DB クラスターに昇格されます。

マルチ AZ DB クラスターのリードレプリカの作成と昇格

マルチAZ DB クラスターのリードレプリカの作成と昇格には、AWS マネジメントコンソール、AWS CLI、または RDS API を使用します。

> **Note**
>
> ソース DB インスタンスの Amazon VPC に基づいて、すべてのリードレプリカを同じ仮想プライベートクラウド (VPC) に作成することを強くお勧めします。
> リードレプリカをソース DB インスタンスとは異なる VPC に作成すると、クラスレスドメイン間ルーティング (CIDR) の範囲がレプリカと Amazon RDS システムとの間で重複する可能性があります。CIDR が重複すると、レプリカが不安定になり、レプリカに接続するアプリケーションに悪影響を及ぼす可能性があります。リードレプリカの作成時にエラーが発生した場合は、別のターゲット DB サブネットグループを選択します。詳細については、「VPC 内の DB インスタンスの使用」を参照してください。

コンソール

リードレプリカを使用してシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイをマルチ AZ DB クラスターに移行するには、AWS マネジメントコンソール を使用して次の手順を実行します。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. マルチ AZ DB クラスターのリードレプリカを作成します。
   a. ナビゲーションペインで、[データベース] を選択します。
   b. リードレプリカのソースとして使用する DB インスタンスを選択します。
   c. [アクション] で [リードレプリカの作成] を選択します。
   d. [Availability and durability] (可用性と耐久性) で、[Multi-AZ DB cluster] (マルチ AZ DB クラスター) を選択します。
   e. DB インスタンス識別子に、リードレプリカの名前を入力します。
   f. 残りのセクションで、DB クラスター設定を指定します。設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。
   g. [Create read replica] を選択します。
3. 準備ができたら、リードレプリカをスタンドアロンのマルチ AZ DB クラスターに昇格します。
   a. トランザクションがソース DB インスタンスに書き込まれるのを停止して、すべての更新がリードレプリカに反映されるまで待ちます。

データベースの更新は、プライマイ DB インスタンスで行われた後にリードレプリカで行われます。このレプリケーションのラグは大きく異なる可能性があります。`ReplicaLag` メトリクスを使用して、リードレプリカにすべての更新がいつ加えられたかを確認できます。レプリカラグの詳細については、「リードレプリケーションのモニタリング」を参照してください。

b. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

c. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

[Databases (データベース)] ペインが表示されます。各リードレプリカには、[Role (ロール)] 列に [Replica (レプリカ)] があります。

d. 昇格するマルチ AZ DBクラスターのリードレプリカを選択します。

e. [アクション] で、[Promote (昇格)] を選択します。

f. [Promote read replica] (リードレプリカの昇格) ページで、新しく昇格されたマルチ AZ DB クラスターのバックアップ保持期間とバックアップウィンドウを入力します。

g. 希望通りの設定になったら、[Promote read replica] (リードレプリカの昇格) を選択します。

h. 昇格されたマルチ AZ DB クラスターのステータスが `Available` になるまで待ちます。

i. 昇格されたマルチ AZ DB クラスターを使用するようにアプリケーションに指示します。

必要に応じて、シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイが不要になった場合は削除します。手順については、DB インスタンスを削除する を参照してください。

AWS CLI

リードレプリカを使用してシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイをマルチ AZ DB クラスターに移行するには、AWS CLI を使用して次の手順を実行します。

1. マルチ AZ DB クラスターのリードレプリカを作成します。

ソース DB インスタンスからリードレプリカを作成するには、AWS CLI コマンド `create-db-cluster` を使用します。`--replication-source-identifier` として、ソース DB インスタンスの Amazon リソースネーム (ARN) を指定します。

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster \
  --db-cluster-identifier mymultiazdbcluster \
  --replication-source-identifier arn:aws:rds:us-
east-2:123456789012:db:mydbinstance
  --engine postgres \
  --db-cluster-instance-class db.m5d.large \
  --storage-type io1 \
  --iops 1000 \
  --db-subnet-group-name defaultvpc \
  --backup-retention-period 1
```

Windows の場合:

```
aws rds create-db-cluster ^
  --db-cluster-identifier mymultiazdbcluster ^
  --replication-source-identifier arn:aws:rds:us-
east-2:123456789012:db:mydbinstance
  --engine postgres ^
  --db-cluster-instance-class db.m5d.large ^
  --storage-type io1 ^
  --iops 1000 ^
  --db-subnet-group-name defaultvpc ^
  --backup-retention-period 1
```

2. トランザクションがソース DB インスタンスに書き込まれるのを停止して、すべての更新がリードレプリカに反映されるまで待ちます。

   データベースの更新は、プライマイ DB インスタンスで行われた後にリードレプリカで行われます。このレプリケーションのラグは大きく異なる可能性があります。Replica Lag メトリクスを使用して、リードレプリカにすべての更新がいつ加えられたかを確認できます。レプリカラグの詳細については、「リードレプリケーションのモニタリング」を参照してください。

3. 準備ができたら、リードレプリカをスタンドアロンのマルチ AZ DB クラスターに昇格します。

   マルチ AZ DB クラスターのリードレプリカを昇格するには、AWS CLI コマンド promote-read-replica-db-cluster を使用します。--db-cluster-identifier として、マルチ AZ DB クラスターリードレプリカの ID を指定します。

   ```
   aws rds promote-read-replica-db-cluster --db-cluster-identifier mymultiazdbcluster
   ```

4. 昇格されたマルチ AZ DB クラスターのステータスが Available になるまで待ちます。

5. 昇格されたマルチ AZ DB クラスターを使用するようにアプリケーションに指示します。

必要に応じて、シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイが不要になった場合は削除します。手順については、DB インスタンスを削除する を参照してください。

RDS API

リードレプリカを使用してシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイをマルチ AZ DB クラスターに移行するには、RDS AAPI を使用して次の手順を実行します。

1. マルチ AZ DB クラスターのリードレプリカを作成します。

   マルチ AZ DB クラスターのリードレプリカを作成するには、必須パラメータ `DBClusterIdentifier` を指定して、`CreateDBCluster` オペレーションを使用します。`ReplicationSourceIdentifier` として、ソース DB インスタンスの Amazon リソースネーム (ARN) を指定します。
2. トランザクションがソース DB インスタンスに書き込まれるのを停止して、すべての更新がリードレプリカに反映されるまで待ちます。

   データベースの更新は、プライマイ DB インスタンスで行われた後にリードレプリカで行われます。このレプリケーションのラグは大きく異なる可能性があります。`Replica Lag` メトリクスを使用して、リードレプリカにすべての更新がいつ加えられたかを確認できます。レプリカラグの詳細については、「リードレプリケーションのモニタリング」を参照してください。
3. 準備ができたら、リードレプリカをスタンドアロンのマルチ AZ DB クラスターに昇格します。

   マルチ AZ DB クラスターのリードレプリカを昇格するには、必須パラメータ `DBClusterIdentifier` を指定して、`PromoteReadReplicaDBCluster` オペレーションを使用します。マルチ AZ DB クラスターリードレプリカの ID を指定します。
4. 昇格されたマルチ AZ DB クラスターのステータスが `Available` になるまで待ちます。
5. 昇格されたマルチ AZ DB クラスターを使用するようにアプリケーションに指示します。

必要に応じて、シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイが不要になった場合は削除します。手順については、DB インスタンスを削除する を参照してください。

マルチ AZ DB クラスターのリードレプリカの作成に関する制限事項

次の制限は、シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイから マルチ AZ DB クラスターのリードレプリカを作成する際に適用されます。

- ソース DB インスタンスを所有する AWS アカウント とは異なる AWS アカウント では、マルチ AZ DB クラスターのリードレプリカを作成することはできません。
- マルチ AZ DB クラスターのリードレプリカは、ソース DB インスタンスとは異なる AWS リージョン で作成することはできません。
- マルチ AZ DB クラスターのリードレプリカを指定の時点に復元することはできません。
- ストレージ暗号化は、ソース DB インスタンスとマルチ AZ DB クラスターで同じ設定にする必要があります。
- ソース DB インスタンスが暗号化されている場合、マルチ AZ DB クラスターのリードレプリカは同じ KMS キーを使用して暗号化する必要があります。
- ソース DB インスタンスが汎用 SSD (gp3) ストレージを使用しており、割り当てられたストレージが 400 GiB 未満の場合、マルチ AZ DB クラスターのリードレプリカのプロビジョンド IOPS は変更できません。
- ソース DB インスタンスでマイナーバージョンアップグレードを実行するには、まず、マルチ AZ DB クラスターリードレプリカでマイナーバージョンアップグレードを実行する必要があります。
- RDS for PostgreSQL マルチ AZ DB クラスターのリードレプリカでマイナーバージョンアップグレードを実行すると、アップグレード後にリーダー DB インスタンスがライター DB インスタンスに切り替えられません。したがって、Amazon RDS がライターインスタンスをアップグレードしている間に、DB クラスターでダウンタイムが発生する可能性があります。
- マルチ AZ DB クラスターのリードレプリカでメジャーバージョンアップグレードを実行することはできません。
- マルチ AZ DB クラスターリードレプリカのソース DB インスタンスでメジャーバージョンアップグレードを実行できますが、リードレプリカへのレプリケーションは停止し、再開できません。
- マルチ AZ DB クラスターのリードレプリカは、カスケードリードレプリカをサポートしていません。
- RDS for PostgreSQL では、マルチ AZ DB クラスターのリードレプリカはフェイルオーバーできません。

マルチ AZ DB クラスターからの DB インスタンスリードレプリカの作成

クラスターのコンピューティングまたは I/O のキャパシティを超えてスケーリングするために、マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成できます。このような過度の読み込みトラフィックを 1 つまたは複数の DB インスタンスのリードレプリカに割り振ることができます。また、リードレプリカを使用して、マルチ AZ DB クラスターから DB インスタンスに移行することもできます。

リードレプリカを作成するには、マルチ AZ DB クラスターをレプリケーションソースとして指定します。ライターインスタンスではなく、マルチ AZ DB クラスターのリーダーインスタンスのいずれかが常にレプリケーションのソースです。この条件により、フェイルオーバーが発生した場合でも、レプリカは常にソースクラスターと同期されます。

トピック

- リーダー DB インスタンスと DB インスタンスのリードレプリカの比較
- 考慮事項
- DB インスタンスのリードレプリカの作成
- DB インスタンスのリードレプリカの昇格
- マルチ AZ DB クラスターからの DB インスタンスリードレプリカの作成に関する制限事項

リーダー DB インスタンスと DB インスタンスのリードレプリカの比較

マルチ AZ DB クラスターの DB インスタンスのリードレプリカは、マルチ AZ DB クラスターのリーダー DB インスタンスと次の点で異なります。

- リーダー DB インスタンスは自動フェイルオーバーターゲットとして機能しますが、DB インスタンスのリードレプリカにその機能はありません。
- リーダー DB インスタンスは、変更をコミットする前に、ライター DB インスタンスからの変更を確認する必要があります。ただし、DB インスタンスのリードレプリカの場合、更新は確認を必要とせずにリードレプリカに非同期的にコピーされます。
- リーダー DB インスタンスは、マルチ AZ DB クラスターのライター DB インスタンスと常に同じインスタンスクラス、ストレージタイプ、およびエンジンバージョンを共有します。ただし、DB インスタンスのリードレプリカは、必ずしもソースクラスターと同じ設定を共有する必要はありません。
- DB インスタンスのリードレプリカをスタンドアロン DB インスタンスに昇格させることができます。マルチ AZ DB クラスターの DB インスタンスをスタンドアロンインスタンスに昇格することはできません。
- リーダーエンドポイントは、マルチ AZ DB クラスターのリーダー DB インスタンスにのみリクエストをルーティングします。DB インスタンスのリードレプリカにリクエストをルーティングすることはありません。

リーダーおよびライター DB インスタンスの詳細については、「the section called “マルチ AZ DB クラスターアーキテクチャ”」を参照してください。

考慮事項

マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成する前に以下を検討してください。

- DB インスタンスのリードレプリカを作成するとき、ソースクラスターと同じメジャーバージョンで、同じかそれ以上のマイナーバージョンでなければなりません。作成後、オプションでリードレプリカをソースクラスターよりも上位のマイナーバージョンにアップグレードできます。
- DB インスタンスのリードレプリカを作成する場合、割り当てられるストレージは、ソースのマルチ AZ DB クラスターに割り当てられたストレージと同じである必要があります。割り当てられたストレージは、リードレプリカの作成後に変更できます。
- RDS for MySQL の場合、ソースマルチ AZ DB クラスターでは、`gtid-mode` パラメータを `ON` に設定する必要があります。詳細については、「the section called “DB クラスターのパラメータグループ”」を参照してください。
- アクティブな長時間実行トランザクションの場合、リードレプリカの作成プロセスに時間がかかることがあります。長時間実行トランザクションが完了してから、リードレプリカを作成することをお勧めします。
- DB インスタンスのリードレプリカのソースマルチ AZ DB インスタンスを削除した場合、書き込み先のリードレプリカはスタンドアロン DB インスタンスに昇格されます。

DB インスタンスのリードレプリカの作成

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成できます。

> **Note**
>
> ソースマルチ AZ DB クラスターの Amazon VPC に基づいて、すべてのリードレプリカを同じ仮想プライベートクラウド (VPC) に作成することを強くお勧めします。
> リードレプリカをソースマルチ AZ DB クラスターとは異なる VPC に作成すると、Classless Inter-Domain Routing (CIDR) の範囲がレプリカと RDS システムとの間で重複する可能性があります。CIDR が重複すると、レプリカが不安定になり、レプリカに接続するアプリケーションに悪影響を及ぼす可能性があります。リードレプリカの作成時にエラーが発生した場合は、別のターゲット DB サブネットグループを選択します。詳細については、「the section called “VPC 内の DB インスタンスの使用”」を参照してください。

コンソール

マルチ AZ DB インスタンスリードレプリカを作成するには、AWS マネジメントコンソール を使用して次のステップを実行します。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. リードレプリカのソースとして使用するマルチ AZ DB クラスターを選択します。
4. [アクション] で [リードレプリカの作成] を選択します。
5. レプリカソースに、正しいマルチ AZ DB クラスターが選択されていることを確認してください。
6. DB 識別子に、リードレプリカの名前を入力します。
7. 残りのセクションで、DB インスタンス設定を指定します。設定の詳細については、「the section called “使用できる設定”」を参照してください。

   > **Note**
   >
   > DB インスタンスのリードレプリカに割り当てられたストレージは、ソースマルチ AZ DB クラスターに割り当てられたストレージと同じである必要があります。

8. [Create read replica] を選択します。

AWS CLI

マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成するには、AWS CLI コマンド `create-db-instance-read-replica` を使用します。`--source-db-cluster-identifier` には、マルチ AZ DB クラスターの識別子を指定します。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
  --db-instance-identifier myreadreplica \
  --source-db-cluster-identifier mymultiazdbcluster
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
```

```
  --db-instance-identifier myreadreplica ^
  --source-db-cluster-identifier mymultiazdbcluster
```

RDS API

マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成するには、CreateDBInstanceReadReplica オペレーションを使用します。

DB インスタンスのリードレプリカの昇格

DB インスタンスのリードレプリカが不要になった場合は、スタンドアロン DB インスタンスに昇格できます。リードレプリカを昇格させると、使用可能になる前に DB インスタンスが再起動されます。手順については、「the section called “リードレプリカの昇格”」を参照してください。

リードレプリカを使用してシングル AZ DB クラスターデプロイをシングル AZ またはマルチ AZ DB インスタンスデプロイに移行している場合は、ソース DB クラスターに書き込まれるトランザクションを必ず停止してください。次に、リードレプリカにすべての更新が行われるまで待ちます。データベースの更新は、マルチ AZ DB クラスターのリーダー DB インスタンスのいずれかで行われた後にリードレプリカで実行されます。このレプリケーションのラグは大きく異なる可能性があります。ReplicaLag メトリクスを使用して、リードレプリカにすべての更新がいつ加えられたかを確認できます。レプリカラグの詳細については、「the section called “リードレプリケーションのモニタリング”」を参照してください。

リードレプリカを昇格させたら、昇格した DB インスタンスのステータスが Available になるのを待ってから、昇格した DB インスタンスを使用するようアプリケーションに指示します。必要に応じて、マルチ AZ DB クラスターデプロイが不要になった場合は削除することもできます。手順については、「the section called “マルチ AZ DB クラスターの削除”」を参照してください。

マルチ AZ DB クラスターからの DB インスタンスリードレプリカの作成に関する制限事項

次の制限事項は、マルチ AZ DB インスタンスデプロイから DB インスタンスのリードレプリカを作成する際に適用されます。

- ソースマルチ AZ DB クラスターを所有する AWS アカウント とは異なる AWS アカウント では、DB インスタンスのリードレプリカを作成することはできません。
- ソースマルチ AZ DB クラスターとは異なる AWS リージョン で DB インスタンスのリードレプリカを作成することはできません。
- DB インスタンスのリードレプリカを指定の時点に復元することはできません。
- ストレージ暗号化は、ソースマルチ AZ DB クラスターと DB インスタンスのリードレプリカで同じ設定にする必要があります。

- ソースマルチ AZ DB クラスターが暗号化されている場合、DB インスタンスのリードレプリカは同じ KMS キーを使用して暗号化する必要があります。
- ソースマルチ AZ DB クラスターでマイナーバージョンアップグレードを実行するには、まず、DB インスタンスのリードレプリカでマイナーバージョンアップグレードを実行する必要があります。
- DB インスタンスのリードレプリカは、カスケードリードレプリカをサポートしていません。
- RDS for PostgreSQL の場合、DB インスタンスのリードレプリカを作成するには、ソースのマルチ AZ DB クラスターは PostgreSQL バージョン 13.11、14.8、または 15.2.R2 以降を実行している必要があります。
- DB インスタンスリードレプリカのマルチ AZ DB クラスターでメジャーバージョンアップグレードを実行できますが、リードレプリカへのレプリケーションが停止し、再開できません。

Amazon RDS のマルチ AZ DB クラスターからの外部レプリケーションの設定

マルチ AZ DB クラスターと Amazon RDS の外部にあるデータベースとの間のレプリケーションを設定できます。

外部レプリケーションにより、マルチ AZ DB クラスターは、オンプレミスまたは別のクラウド環境の RDS DB インスタンスと外部データベース間でデータをレプリケートできます。これは、ディザスタリカバリ、データ移行、および異なる場所のシステム間の整合性の維持に役立ちます。このセクションでは、レプリケーション設定の前提条件、プロセスの設定方法、レプリケーションのレイテンシー、帯域幅、さまざまなデータベースエンジンとの互換性などの重要な考慮事項について説明します。

RDS for MySQL

RDS for MySQL マルチ AZ DB クラスターの外部レプリケーションを設定するには、Amazon RDS がバイナリログファイルを削除する前に、変更がレプリカに適用されるように、クラスター内の DB インスタンスにバイナリログファイルを十分長く保持する必要があります。そのためには、`mysql.rds_set_configuration` ストアドプロシージャを呼び出し、`binlog retention hours` パラメータを指定することによって、バイナリログの保持を設定します。詳細については、「the section called “バイナリログの保持時間”」を参照してください。

`binlog retention hours` のデフォルト値は `NULL` です。つまり、バイナリログは保持されません (0 時間)。マルチ AZ DB クラスターの外部レプリケーションを設定する場合は、 パラメータを `NULL` 以外の値に設定する必要があります。

マルチ AZ DB クラスターのライター DB インスタンスからのみバイナリログ保持を設定でき、設定はすべてのリーダー DB インスタンスに非同期的に伝播されます。

さらに、外部レプリカで GTID ベースのレプリケーションを有効にすることを強くお勧めします。その後、いずれかの DB インスタンスに障害が発生した場合は、クラスター内の別の正常な DB インスタンスからのレプリケーションを再開できます。詳細については、MySQL ドキュメントの「グローバルトランザクション識別子を使用したレプリケーション」を参照してください。

RDS for PostgreSQL

RDS for PostgreSQL マルチ AZ DB クラスターの外部レプリケーションをセットアップするには、論理レプリケーションを有効にする必要があります。手順については、「the section called “マルチ AZ DB クラスターとの PostgreSQL 論理レプリケーション”」を参照してください。

Amazon RDS 用のマルチ AZ DB クラスターの削除

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB マルチ AZ DB クラスターを削除できます。

マルチ AZ DB クラスターの削除に要する時間は、次の要因によって異なります。

- バックアップ保持期間 (削除するバックアップの数) 。
- 削除されるデータの量。
- 最終スナップショットを作成するかどうか。

マルチ AZ DB クラスターを削除する前に、削除保護を無効にする必要があります。詳細については、「the section called “DB インスタンスの削除の前提条件”」を参照してください。DB クラスターの設定を変更することで、削除保護を無効にできます。詳細については、「the section called “マルチ AZ DB クラスターの変更”」を参照してください。

コンソール

マルチ AZ DB クラスターを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、「データベース」 を選択して、削除したいマルチ AZ DB クラスターを選択します。
3. [アクション] で、[削除] を選択します。
4. マルチ AZ DB クラスターの最終 DB スナップショットを作成するには、「最終スナップショットを作成しますか」 を選択します。

   最終スナップショットを作成する場合は、[Final snapshot name] (最終スナップショット名)を入力します。
5. 自動バックアップを保持するには、[Retain automated backups] (自動バックアップの保持) を選択します。
6. ボックスに「**`delete me`**」と入力します。
7. [削除] を選択します。

AWS CLI

AWS CLIを使用して マルチAZ DB クラスターを削除するには、以下のオプションを使用して delete-db-instance コマンドを呼び出します。

- `--db-cluster-identifier`
- `--final-db-snapshot-identifier`、または `--skip-final-snapshot`

Example最終スナップショットで

Linux、macOS、Unix の場合:

```
aws rds delete-db-cluster \
    --db-cluster-identifier mymultiazdbcluster \
    --final-db-snapshot-identifier mymultiazdbclusterfinalsnapshot
```

Windows の場合:

```
aws rds delete-db-cluster ^
    --db-cluster-identifier mymultiazdbcluster ^
    --final-db-snapshot-identifier mymultiazdbclusterfinalsnapshot
```

Example最終スナップショットなし

Linux、macOS、Unix の場合:

```
aws rds delete-db-cluster \
    --db-cluster-identifier mymultiazdbcluster \
    --skip-final-snapshot
```

Windows の場合:

```
aws rds delete-db-cluster ^
    --db-cluster-identifier mymultiazdbcluster ^
    --skip-final-snapshot
```

RDS API

Amazon RDS API を使用して マルチ AZ DB インスタンスを削除するには、以下のパラメータを指定してDeleteDBCluster オペレーションを呼び出します。

- DBClusterIdentifier
- FinalDBSnapshotIdentifier、または SkipFinalSnapshot

Amazon RDS のマルチ AZ DB クラスターの制限事項

マルチ AZ DB クラスターには、3 つの別々のアベイラビリティーゾーンに 1 つのライター DB インスタンスと 2 つのリーダー DB インスタンスがあります。マルチ AZ DB クラスターは、マルチ AZ 配置と比較して、高可用性、読み取りワークロードの容量の増加、および低レイテンシーを提供します。マルチ AZ DB の配置の詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。

マルチ AZ DB クラスターには、次の制約事項が適用されます。

- マルチ AZ DB クラスターでは、次の機能はサポートされていません。
  - IPv6 接続 (デュアルスタックモード)
  - クロスリージョン自動バックアップ
  - Kerberos 認証
  - ポートの変更。別の方法として、マルチ AZ DB クラスターを特定の時点に復元し、別のポートを指定することもできます。
  - オプショングループ
  - 削除されたクラスターのポイントインタイムリカバリ (PITR)
  - 割り当てられる最大ストレージを設定することによるストレージの自動スケーリング。または、ストレージを手動でスケールすることもできます。
  - マルチ AZ DB クラスターの停止と起動
  - マルチ AZ DB クラスターのスナップショットをコピーする
  - 暗号化されていないマルチ AZ DB クラスターを暗号化する
- RDS for MySQL マルチ AZ DB クラスターでは、次のシステムストアドプロシージャのみがサポートされています。
  - `mysql.rds_rotate_general_log`
  - `mysql.rds_rotate_slow_log`
  - `mysql.rds_show_configuration`
  - `mysql.rds_set_external_master_with_auto_position`
  - `mysql.rds_set_configuration`
- RDS for PostgreSQL マルチ AZ DB クラスターは、`aws_s3` と `pg_transport` の PostgreSQL の拡張機能をサポートしていません。
- RDS for PostgreSQL マルチ AZ DB クラスターは、アウトバウンドネットワークアクセスでのカスタム DNS サーバーの使用をサポートしていません。

Amazon RDS の Amazon RDS 延長サポート

RDS 延長サポートを利用すると、RDS 標準サポート終了日以降も、データベースを以前のメジャーエンジンバージョンで引き続き実行できます (追加料金がかかります)。

DB インスタンスを初めて作成または復元するときに、データベースを RDS 延長サポートに登録できます。AWS CLI または RDS API を使用して `EngineLifecycleSupport` パラメータを変更することで、既存の DB インスタンスまたは DB クラスターの登録ステータスをいつでも変更できます。この変更はダウンタイムなしですぐに有効になります。Aurora およびマルチ AZ DB クラスターの場合は、クラスターレベルで設定を変更します。

標準サポート終了日を既に過ぎている DB インスタンスまたは DB クラスターの登録ステータスを無効にすると、インスタンスまたはクラスターはサポートされている次のメジャーバージョンに自動的にアップグレードされます。詳細については、「ModifyDBInstance」および「ModifyDBCluster」を参照してください。

DB インスタンスの作成または復元中に RDS 延長サポートを有効にした場合、RDS の標準サポート終了日以降、Amazon RDS は自動的に DB インスタンスを RDS 延長サポートに登録します。RDS 延長サポートへの自動登録は、データベースエンジンを変更せず、DB インスタンスの稼働時間やパフォーマンスにも影響しません。

RDS 延長サポートでは、以下の更新とテクニカルサポートが提供されます。

- データベースエンジンを含む DB インスタンスまたは DB クラスターの重大および高 CVE に対するセキュリティ更新
- 重大な問題のバグ修正とパッチ
- 標準 Amazon RDS サービスレベル契約の範囲内でサポートケースを開き、トラブルシューティングのヘルプを受け取る機能

この有料サービスを利用すると、サポートされているメジャーエンジンバージョンへのアップグレードにかかる時間が長くなります。たとえば、RDS for MySQL 5.7 バージョンおよび RDS for MySQL バージョンのサポート終了日は 2024 年 2 月 29 日です。しかし、その日までに RDS for MySQL バージョン 8.0 への手動アップグレードの準備ができていないとします。この場合、Amazon RDS は、2024 年 2 月 29 日にデータベースを RDS 延長サポートに自動的に登録するため、引き続き RDS for MySQL バージョン 5.7 を実行できます。2024 年 3 月 1 日以降、Amazon RDS によって RDS 延長サポートの料金が自動的に請求されます。

RDS 延長サポートは、メジャーエンジンバージョンのRDS の標準サポート終了日 から最大 3 年間ご利用いただけます。この期間が過ぎてもメジャーエンジンバージョンをサポート対象バージョンにアップグレードしていない場合、Amazon RDS によってメジャーエンジンバージョンが自動的にアップグレードされます。サポート対象のメジャーエンジンバージョンへできるだけ早くアップグレードすることをお勧めします。

RDS 標準サポート終了日と RDS 拡張サポート終了日の詳細については、「Amazon RDS でサポートされている MySQL のメジャーバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。

トピック

- Amazon RDS 延長サポートの概要
- Amazon RDS 延長サポート料金
- Amazon RDS 延長サポートが適用されるバージョン
- Amazon RDS 延長サポートにおける Amazon RDS とお客様の責任
- Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの作成
- Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの登録を確認します。
- Amazon RDS 延長サポートでのエンジンバージョンのサポート日を表示
- Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元

Amazon RDS 延長サポートの概要

RDS の標準サポート終了日が過ぎて、DB インスタンスおよび DB クラスターの作成時または復元時に RDS 延長サポートを無効にしなかった場合、Amazon RDS は DB インスタンスおよび DB クラスターを自動的に RDS 延長サポートに登録します。既存の DB インスタンスまたは DB クラスターの `EngineLifecycleSupport` パラメータを変更することで、自動登録を防ぐこともできます。Amazon RDS は、DB インスタンスを、RDS 標準サポート終了日より前にリリースされた最新のマイナーバージョンに自動的にアップグレードします (該当バージョンをまだ実行していない場合)。Amazon RDS は、メジャーエンジンバージョンの RDS 標準サポート終了日が過ぎるまでは、マイナーバージョンをアップグレードしません。

RDS 標準サポート終了日に達したメジャーエンジンバージョンで新しいデータベースを作成できます。RDS は、これらの新しいデータベースを RDS 延長サポートに自動的に登録し、このサービスの料金を請求します。

RDS 標準サポート終了日前に RDS 標準サポートがまだ適用されているエンジンにアップグレードすると、Amazon RDS はエンジンを RDS 延長サポートに登録しません。

RDS 標準サポート終了日を過ぎているが、RDS 延長サポートに登録していないエンジンと互換性があるデータベースのスナップショットを復元しようとすると、Amazon RDS は、RDS 標準サポートがまだ適用されている最新のエンジンバージョンとの互換性を持つようにスナップショットをアップグレードしようとします。復元に失敗すると、Amazon RDS は、スナップショットと互換性があるバージョンを使用してエンジンを RDS 延長サポートに自動的に登録します。

RDS 延長サポートへの登録はいつでも終了できます。登録を終了するには、AWS CLI または RDS API を使用して `EngineLifecycleSupport` パラメータを変更できます。標準サポート終了日を既に過ぎている DB インスタンスまたは DB クラスターの登録ステータスを無効にすると、インスタンスまたはクラスターはサポートされている次のメジャーバージョンに自動的にアップグレードされます。

登録を終了するには、登録した各エンジンを、RDS 標準サポートがまだ適用されている、より新しいエンジンバージョンにアップグレードすることもできます。RDS 延長サポートへの登録の終了は、RDS 標準サポートがまだ適用されている、より新しいエンジンバージョンへのアップグレードを完了した日から有効になります。

RDS 標準サポート終了日と RDS 拡張サポート終了日の詳細については、「Amazon RDS でサポートされている MySQL のメジャーバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。

Amazon RDS 延長サポート料金

RDS 延長サポートに登録しているすべてのエンジンには、RDS 標準サポート終了日の翌日から、料金が発生します。RDS 標準サポート終了日については、「メジャーバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。RDS 延長サポート料金は、マルチ AZ 配置でのスタンバイインスタンスに適用されます。

RDS 延長サポートの追加料金は、以下のいずれかのアクションを実行すると、自動的に停止します。

- 標準サポートの対象となるエンジンバージョンにアップグレードします。
- RDS 標準サポート終了日を過ぎてメジャーバージョンを実行しているデータベースを削除します。

将来、ターゲットエンジンバージョンが延長サポートに移行すると、料金が再び発生します。

例えば、RDS for PostgreSQL 11 は 2024 年 3 月 1 日に延長サポートが開始されますが、2024 年 4 月 1 日まで課金は開始されません。2024 年 4 月 30 日に、RDS for PostgreSQL 11 データベースを RDS for PostgreSQL 12 にアップグレードします。RDS for PostgreSQL 11 の延長サポートに対しては 30 日間のみ課金されます。RDS 標準サポート終了日である 2025 年 2 月 28 日が過ぎても、この DB インスタンスで RDS for PostgreSQL 12 を引き続き実行します。データベースには、2025 年 3 月 1 日から RDS 延長サポート料金が再び発生します。

詳細については、「Amazon RDS for MySQL の価格設定」と「Amazon RDS for PostgreSQL の価格設定」「」を参照してください。

Amazon RDS 延長サポート料金の回避

RDS 延長サポート料金を回避するには、RDS 標準サポート終了日が過ぎたら、RDS で DB インスタンスまたはマルチ AZ DB クラスターを作成または復元できないようにします。これには、AWS CLI または RDS API を使用します。

AWS CLI または RDS API を使用して `EngineLifecycleSupport` パラメータを変更することで、既存の DB インスタンスまたは DB クラスターの登録ステータスをいつでも変更し、RDS 延長サポートの課金を回避できます。標準サポート終了日を既に過ぎている DB インスタンスまたは DB クラスターの登録ステータスを無効にすると、インスタンスまたはクラスターはサポートされている次のメジャーバージョンに自動的にアップグレードされます。

AWS CLI の `--engine-lifecycle-support` オプションで、`open-source-rds-extended-support-disabled` と指定します。RDS API で、`EngineLifecycleSupport` パラメータに `open-source-rds-extended-support-disabled` を指定します。詳細については「DB インスタンスまたはマルチ AZ DB クラスターの作成」または「DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

Amazon RDS 延長サポートが適用されるバージョン

RDS 延長サポートは、メジャーバージョンでのみ利用できます。マイナーバージョンでは利用できません。

RDS 延長サポートは、RDS for MySQL および RDS for PostgreSQL で利用できます。詳細については、「メジャーバージョン」および「Amazon RDS for PostgreSQL リリースノート」の「Release calendar for Amazon RDS for PostgreSQL」を参照してください。

AWS CLI または RDS API を使用して、エンジンバージョンのサポート日に関する情報を表示することもできます。詳細については、「Amazon RDS 延長サポートでのエンジンバージョンのサポート日を表示」を参照してください。

Amazon RDS 延長サポートバージョンの命名規則

Amazon RDS は、RDS 延長サポートにおいてエンジンの修正と CVE パッチを含む新しいマイナーバージョンをリリースします。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」および「Amazon RDS for PostgreSQL リリースノート」の「Amazon RDS Extended Support updates for RDS for PostgreSQL」を参照してください。

このようなマイナーリリースの名前は major.minor-RDS.YYYYMMDD.patch.YYYYMMDD という形式になり、例えば RDS for MySQL の場合は 5.7.44-RDS.20240208.R2.20240210、RDS for PostgreSQL の場合11.22-RDS.20240208.R2.20240210 となります。

major

MySQL の場合、メジャーバージョン番号はバージョン番号の整数部分と 1 つ目の小数部分の両方です (8.0 など)。メジャーバージョンのアップグレードでは、バージョン番号の主要な部分が大きくなります。例えば、5.7.44 から 8.0.33 へのアップグレードはメジャーバージョンアップグレードであり、5.7 と 8.0 はメジャーバージョン番号です。

PostgreSQL の場合、メジャーバージョン番号は整数、例えば 11 です。

minor-RDS.YYYYMMDD

MySQL の場合、マイナーバージョン番号は、バージョン番号の 3 番目の部分です。例えば、5.7.44-RDS.20240208 では 44-RDS.20240208 です。

PostgeSQL の場合、マイナーバージョン番号は、バージョン番号の 2 番目の部分です。例えば、11.22-RDS.20240208 では 22-RDS.20240208 です。

日付は、Amazon RDS によって Amazon RDS マイナーバージョンが作成された日です。

patch

Amazon RDS によって Amazon RDS マイナーバージョンが作成された日付の後にくるのがパッチバージョンです。例えば、5.7.44-RDS.20240208.R2 または 11.22-RDS.20240208.R2 では R2 です。

Amazon RDS パッチバージョンには、リリース後に Amazon RDS マイナーバージョンに追加された重要なバグ修正が含まれています。

YYYYMMDD

Amazon RDS によってパッチバージョンが作成された日が日付になります。例えば、5.7.44-RDS.20240208.R2.20240210 または 11.22-RDS.20240208.R2.20240210 では20240210 です。

Amazon RDS 日付バージョンは、リリース後にマイナーバージョンに追加された重要なセキュリティ修正を含むセキュリティパッチです。エンジンの動作を変更する可能性のある修正は含まれていません。

Amazon RDS 延長サポートにおける Amazon RDS とお客様の責任

RDS 延長サポートにおける Amazon RDS とお客様の責任について以下に説明します。

トピック

- Amazon RDS の責任
- お客様の責任

Amazon RDS の責任

RDS の標準サポート終了日以降、Amazon RDS は RDS 延長サポートに登録しているエンジンに対してパッチ、バグ修正、アップグレードを提供します。この期間は、最長で 3 年間、またはエンジンの使用を停止するまでのいずれか早いほうとなります。

パッチは、国家脆弱性データベース (NVD) の CVSS 重大度評価で定義されている重大および高 CVE が対象となります。詳細については、「脆弱性メトリクス」を参照してください。

お客様の責任

お客様は、RDS 延長サポートに登録している DB インスタンスまたはマルチ AZ DB クラスターに対して提供されるパッチ、バグ修正、アップグレードを適用する責任があります。Amazon RDS は、このようなパッチ、バグ修正、アップグレードをいつでも変更、置換、または撤回する権利を留保します。セキュリティまたは重大な安定性の問題に対処するためにパッチが必要な場合、Amazon RDS は、お客様の DB インスタンスまたはマルチ AZ DB クラスターをパッチで更新するか、お客様にパッチのインストールを要求する権利を留保します。

また、お客様は、RDS 延長サポート終了日より前にお客様のエンジンをより新しいエンジンバージョンにアップグレードする責任もあります。RDS 延長サポート終了日は、通常、RDS 標準サポート終了日から 3 年です。データベースのメジャーエンジンバージョンの RDS 延長サポート終了日については、「メジャーバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。

エンジンをアップグレードしない場合、RDS 延長サポート終了日が過ぎると、Amazon RDS は、エンジンを RDS 標準サポート対象のより新しいエンジンバージョンにアップグレードしようとしま

す。アップグレードに失敗すると、Amazon RDS は、RDS 標準サポート終了日を過ぎてエンジンを実行している DB インスタンスまたはマルチ AZ DB クラスターを削除する権利を留保します。ただし、その前に、Amazon RDS はそのエンジンのデータを保存します。

Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの作成

DB インスタンスまたはマルチ AZ DB クラスターを作成する場合、コンソールで [RDS 延長サポートを有効にする] を選択するか、AWS CLI の拡張サポートオプションまたは RDS API のパラメータを使用します。DB インスタンスまたはマルチ AZ DB クラスターを Amazon RDS 延長サポートに登録すると、DB インスタンスまたはマルチ AZ DB クラスターの存続期間中、RDS 延長サポートに永続的に登録されます。

コンソールを使用する場合は、[RDS 延長サポートを有効にする] を選択する必要があります。この設定はデフォルトでは選択されていません。

AWS CLI または RDS API を使用し、RDS 延長サポートの設定を指定しなかった場合、Amazon RDS はデフォルトで RDS 延長サポートを有効にします。CloudFormation またはその他のサービスを使用することによって自動化すると、このデフォルトの動作により、RDS の標準サポート終了日を過ぎても、データベースの可用性が維持されます。

AWS CLI または RDS API を使用して DB インスタンスまたはマルチ AZ DB クラスターを作成することにより、RDS 延長サポートの登録を防止できます。

トピック

- RDS 延長サポートの動作
- RDS 延長サポートに関する考慮事項
- RDS 延長サポートで DB インスタンスまたはマルチ AZ DB クラスターを作成する

RDS 延長サポートの動作

次の表は、メジャーエンジンバージョンが RDS の標準サポート終了に達したときの動作をまとめたものです。

RDS 延長サポートのステータス*	行動
有効	RDS 延長サポートの Amazon RDS 料金。
Disabled	Amazon RDS は DB インスタンスまたはマルチ AZ DB クラスターをサポートされるエンジンバージョンへアップグレードします。このアップグレードは、RDS の標準サポート終了日に、またはその直後に行われます。

* RDS コンソールでは、RDS 延長サポートのステータスは「はい」または「いいえ」と表示されます。AWS CLI または RDS API では、RDS 延長サポートのステータスは `open-source-rds-extended-support` または `open-source-rds-extended-support-disabled` と表示されます。

RDS 延長サポートに関する考慮事項

DB インスタンスまたはマルチ AZ DB クラスターを作成する前に、次の点を考慮してください。

- RDS 標準サポート終了日が過ぎた後、新しい DB インスタンスまたは新しいマルチ AZ DB クラスターの作成を阻止し、RDS 延長サポート料金を回避できます。これには、AWS CLI または RDS API を使用します。AWS CLI の `--engine-lifecycle-support` オプションで、`open-source-rds-extended-support-disabled` と指定します。RDS API で、`EngineLifecycleSupport` パラメータに `open-source-rds-extended-support-disabled` を指定します。`open-source-rds-extended-support-disabled` と指定して、RDS 標準サポート終了日が過ぎると、DB インスタンスまたはマルチ AZ DB クラスター の作成は常に失敗します。
- RDS 延長サポートはクラスターレベルで設定されます。クラスターのメンバーの RDS 延長サポート設定は、RDS コンソール、AWS CLI の `--engine-lifecycle-support`、RDS API の `EngineLifecycleSupport` で常に同じになります。

詳細については、「MySQL のバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。

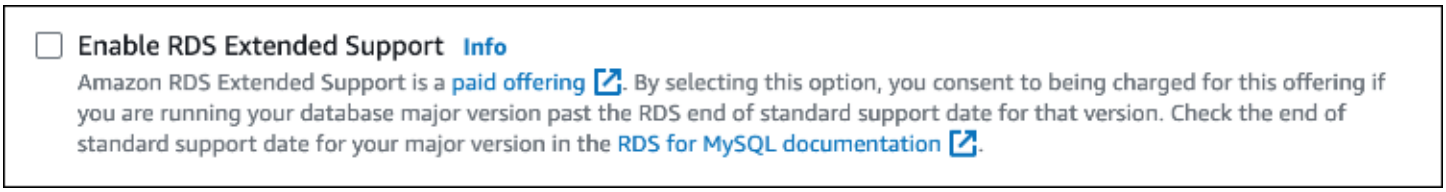
RDS 延長サポートで DB インスタンスまたはマルチ AZ DB クラスターを作成する

AWS マネジメントコンソール、AWS CLI、または RDS API で RDS 延長サポートを使用して、DB インスタンスまたはマルチ AZ DB クラスターを作成する方法について説明します。

コンソール

DB インスタンスまたはマルチ AZ DB クラスターを作成する場合、[エンジンのオプション] セクションで、[RDS 延長サポートの有効化] を選択します。この設定はデフォルトでは選択されていません。

次の画像は、[RDS 延長サポートを有効にする] 設定を示しています。



AWS CLI

create-db-instance または create-db-cluster (マルチ AZ DB クラスター) AWS CLI コマンドを実行する場合は、`--engine-lifecycle-support` オプションに `open-source-rds-extended-support` を指定して RDS 延長サポートを選択します。このオプションはデフォルトで `open-source-rds-extended-support` に設定されています。

RDS 標準サポート終了日以降に、新しい DB インスタンスやマルチ AZ DB クラスターが作成されないようにするには、`--engine-lifecycle-support` オプションを `open-source-rds-extended-support-disabled` と指定します。これにより、関連する RDS 延長サポート料金は発生しません。

RDS API

CreateDBInstance または CreateDBCluster (マルチ AZ DB クラスター) Amazon RDS API オペレーションを使用する場合は、`EngineLifecycleSupport` パラメータを `open-source-rds-extended-support` に設定して RDS 延長サポートを選択します。デフォルトでは、このパラメータは `open-source-rds-extended-support` に設定されます。

RDS 標準サポート終了日以降に、新しい DB インスタンスまたはマルチ AZ DB クラスター が作成されないようにするには、`EngineLifecycleSupport` パラメータに `open-source-rds-`

`extended-support-disabled` を指定します。これにより、関連する RDS 延長サポート料金は発生しません。

詳細については、以下の各トピックを参照してください。

- DB インスタンスを作成するには、「Amazon RDS DB インスタンスの作成」の DB エンジンの手順に従ってください。
- マルチ AZ DB クラスターを作成するには、「Amazon RDS 用のマルチ AZ DB クラスターの作成」の DB エンジンの手順に従ってください。

Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの登録を確認します。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの登録を表示できます。

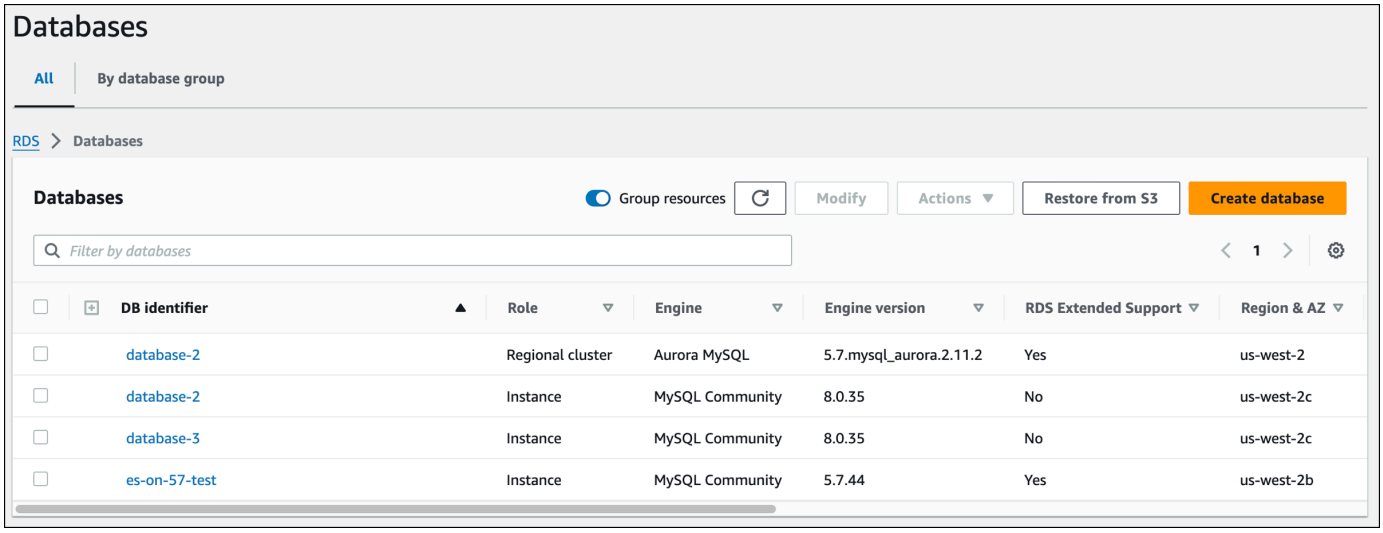
> **Note**
>
> コンソールの [RDS 延長サポート] 列、AWS CLI の [`--engine-lifecycle-support`] オプション、および RDS API の `EngineLifecycleSupport` パラメータは、RDS 延長サポートへの登録のみを示します。RDS 延長サポートの料金は、DB エンジンバージョンが RDS 標準サポートの終了に達した場合にのみ開始されます。詳細については、「メジャーバージョン」および「Amazon RDS for PostgreSQL リリースノート」の「Release calendar for Amazon RDS for PostgreSQL」を参照してください。
> 例えば、RDS 延長サポートに登録されている RDS for MySQL 5.7 データベースがあるとします。2024 年 3 月 1 日、Amazon RDS はこのデータベースの RDS 延長サポートの課金を開始しました。2024 年 7 月 31 日、このデータベースを RDS for MySQL 8.0 にアップグレードしました。このデータベースの RDS 延長サポートのステータスは引き続き有効です。ただし、RDS for MySQL 8.0 はまだ RDS 標準サポートの終了に達していないため、このデータベースの RDS 延長サポート料金は停止しました。Amazon RDS は、RDS for MySQL 8.0 の RDS 標準サポートが終了する 2026 年 8 月 1 日まで、このデータベースの RDS 延長サポートの料金を請求しません。

コンソール

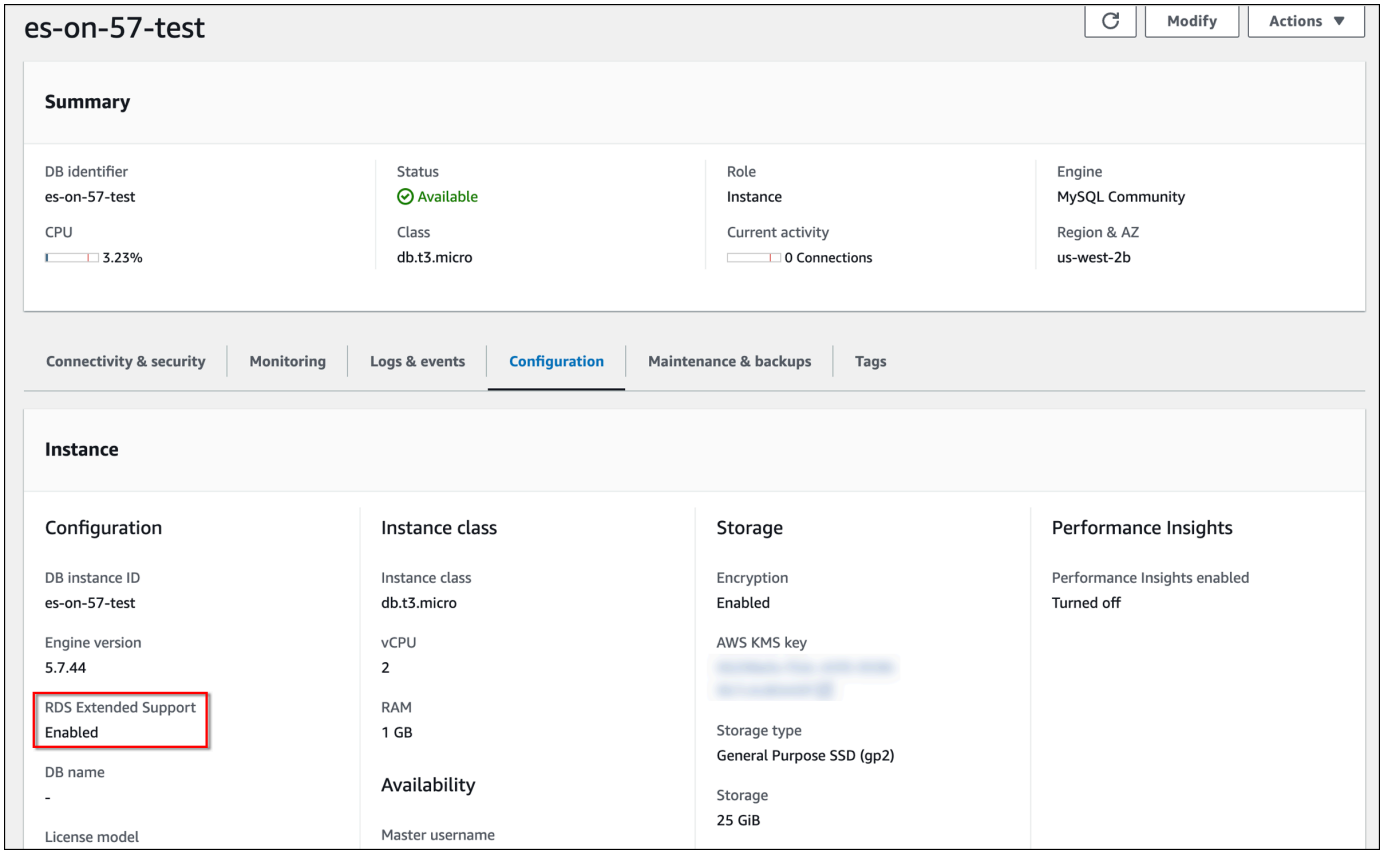
RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターAurora DB クラスターまたはグローバルクラスターの登録を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。[RDS 延長サポート] の値は、DB インスタンスまたはマルチ AZ DB クラスターが RDS 延長サポートに登録されているかどうかを示します。値が表示されない場合は、データベースの RDS 延長サポートを利用できません。

> **Tip**
>
> [RDS 延長サポート] 列が表示されない場合は、[設定] アイコンを選択し、[RDS 延長サポート] をオンにします。



3. 各データベースの [設定] タブで登録を表示することもできます。[DB 識別子] でデータベースを選択します。[設定] タブの [延長サポート] で、データベースが登録されているかどうかを確認します。



AWS CLI

AWS CLI を使用して RDS 延長サポートへのデータベースの登録を表示するには、describe-db-instances または describe-db-clusters (マルチ AZ DB クラスター) コマンドを実行します。

RDS 延長サポートがデータベースで利用可能な場合、レスポンスにはパラメータ `EngineLifecycleSupport` が含まれます。値 `open-source-rds-extended-support` は、DB インスタンスまたはマルチ AZ DB クラスターが RDS 延長サポートに登録されていることを示します。値 `open-source-rds-extended-support-disabled` は、DB インスタンスまたはマルチ AZ DB クラスターが RDS 延長サポートでは無効化されたことを示します。

例

次のコマンドは、すべての DB インスタンスの情報を返します。

```
aws rds describe-db-instances
```

次のレスポンスは、DB インスタンス `database-1` で実行されている PostgreSQL エンジンが RDS 延長サポートに登録されていることを示しています。

```
{
    "DBInstanceIdentifier": "database-1",
    "DBInstanceClass": "db.t3.large",
    "Engine": "postgres",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

RDS API

Amazon RDS API を使用して RDS 延長サポートへのデータベースの登録を表示するには、DescribeDBInstances または DescribeDBClusters オペレーションを使用します。

RDS 延長サポートがデータベースで利用可能な場合、レスポンスにはパラメータ `EngineLifecycleSupport` が含まれます。値 `open-source-rds-extended-support` は、DB インスタンスまたはマルチ AZ DB クラスターが RDS 延長サポートに登録されていることを示します。値 `open-source-rds-extended-support-disabled` は、DB インスタンスまたはマルチ AZ DB クラスターが RDS 延長サポートでは無効化されたことを示します。

Amazon RDS 延長サポートでのエンジンバージョンのサポート日を表示

AWS CLI または RDS API を使用して、Amazon RDS 延長サポートの DB インスタンスまたはマルチ AZ DB クラスター のエンジンバージョンのサポート日に関する情報を表示できます。この情報は、アップグレードの計画に役立ちます。

AWS CLI コマンドと RDS API オペレーションは、RDS 標準サポートと RDS 延長サポートの開始日と終了日を返します。これらの日付は、メジャーエンジンバージョンテーブルでも確認できます。詳細については、「Amazon RDS でサポートされている MySQL のメジャーバージョン」および「Amazon RDS for PostgreSQL リリースノート」の「Release calendar for Amazon RDS for PostgreSQL」を参照してください。

AWS CLI

AWS CLI を使用して、メジャーエンジンバージョンの RDS 標準サポートと RDS 延長サポートの開始日と終了日を表示するには、describe-db-major-engine-versions コマンドを実行します。

このコマンドは、オープンソースエンジン (MariaDB、MySQL、PostgreSQL) に関連する以下の関連パラメータを返します。商用エンジン (Db2、SQL Server、Oracle) では、これらのパラメータは返されません。

- SupportedEngineLifecycles – このパラメータは、LifecycleSupportName、LifecycleSupportStartDate、LifecycleSupportEndDate および を含むオブジェクトの配列です。
- LifecycleSupportName – このパラメータは、エンジンバージョンがサポートされているタイプを示します。RDS 標準サポート (open-source-rds-standard-support) または RDS 延長サポート (open-source-rds-extended-support)。MariaDB の場合、このパラメータは RDS 標準サポート (open-source-rds-standard-support) のみを返します。
- LifecycleSupportStartDate – このパラメータは、LifecycleSupportName の値に応じて、RDS 標準サポートまたはメジャーエンジンバージョンの RDS 延長サポートの開始日を一覧表示します。
- LifecycleSupportEndDate – このパラメータは、LifecycleSupportName の値に応じて、RDS 標準サポートまたはメジャーエンジンバージョンの RDS 延長サポートの終了日を一覧表示します。

例

レスポンスの例は、次の情報を示しています。

- MySQL 5.7 のサポートされているエンジンライフサイクル open-source-rds-standard-support と open-source-rds-extended-support の開始日と終了日。MySQL 5.7 では RDS 延長サポートが利用可能です。
- MariaDB 10.6 でサポートされているエンジンライフサイクル open-source-rds-standard-support の開始日と終了日。RDS 延長サポートは MariaDB 10.6 では使用できません。
- SQL Server は商用エンジンであるため、SQL Server Enterprise Edition 13 でサポートされているエンジンライフサイクルに関する情報はありません。

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
```

```
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2016-02-22T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-02-29T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-03-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        {
            "Engine": "mariadb",
            "MajorEngineVersion": "10.6",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2022-02-03T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2026-07-31T23:59:59.999000+00:00"
                }
            ]
        },
        {
            "Engine": "sqlserver-ee",
            "MajorEngineVersion": "13.00"
        },
        ...
    ]
}
```

RDS API

RDS API を使用して、メジャーエンジンバージョンの RDS 標準サポートと RDS 延長サポートの開始日と終了日を表示するには、DescribeDBMajorEngineVersions オペレーションを使用します。

エンジンバージョンで RDS 延長サポートが利用可能な場合、レスポンスには、パラメータ `SupportedEngineLifeCycles` が 2 つのオブジェクトを持つ配列として含まれます。1 つのオブジェクトには、RDS 標準サポートの開始日と終了日が含まれます。2 つ目のオブジェクトには、RDS 延長サポートの開始日と終了日が含まれます。

RDS 延長サポートがオープンソースエンジンバージョン (MariaDB、MySQL、PostgreSQL) で利用できない場合、レスポンスには、単一のオブジェクトを持つ配列としてパラメータ

SupportedEngineLifeCycles のみが含まれます。このオブジェクトには、RDS 標準サポートの開始日と終了日が含まれます。

エンジンバージョンが商用エンジン (Db2、SQL Server、Oracle) の場合、レスポンスにはパラメータ SupportedEngineLifeCycles は含まれません。

Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元

DB インスタンス、マルチ AZ DB クラスターを復元するときは、コンソールで [RDS 延長サポートを有効にする] を選択するか、AWS CLI の拡張サポートオプションまたは RDS API のパラメータを使用します。RDS 延長サポートに DB インスタンスまたはマルチ AZ DB クラスターを登録すると、DB インスタンスまたはマルチ AZ DB クラスターの存続期間中、RDS 延長サポートに永続的に登録されます。

RDS 延長サポート設定のデフォルトは、コンソール、AWS CLI、または RDS API を使用してデータベースを復元するかどうかによって異なります。コンソールを使用し、[RDS 延長サポートを有効にする] を選択せず、復元するメジャーエンジンバージョンが RDS の標準サポートの終了を過ぎている場合、Amazon RDS は DB インスタンスを新しいエンジンバージョンに自動的にアップグレードします。AWS CLI または RDS API を使用し、RDS 延長サポートの設定を指定しなかった場合、Amazon RDS はデフォルトで RDS 延長サポートを有効にします。CloudFormation またはその他のサービスを使用することによって自動化すると、このデフォルトの動作により、RDS の標準サポート終了日を過ぎても、データベースの可用性が維持されます。AWS CLI または RDS API を使用することによって、RDS 延長サポートを無効にできます。

トピック

- RDS 延長サポートの動作
- RDS 延長サポートに関する考慮事項
- RDS 延長サポートを使用した DB インスタンスまたはマルチ AZ DB クラスター を復元する

RDS 延長サポートの動作

次の表は、復元する DB インスタンスまたはマルチ AZ DB クラスターのメジャーエンジンバージョンが RDS の標準サポート終了に達した場合の動作をまとめたものです。

RDS 延長サポートのステータス*	行動
有効	RDS 延長サポートの Amazon RDS 料金。
無効**	復元が完了すると、Amazon RDS は DB インスタンスまたはマルチ AZ DB クラスターを新しいエンジンバージョンに自動的にアップグレードします (将来のメンテナンスウィンドウで)。

* RDS コンソールでは、RDS 延長サポートのステータスは「はい」または「いいえ」と表示されます。AWS CLI または RDS API では、RDS 延長サポートのステータスは `open-source-rds-extended-support` または `open-source-rds-extended-support-disabled` と表示されます。

** このオプションは、PostgreSQL 12 以降または MySQL 8 以降を実行している DB インスタンスまたはマルチ AZ DB クラスターを復元する場合にのみ使用できます。

RDS 延長サポートに関する考慮事項

DB インスタンスまたはマルチ AZ DB クラスターを復元する前に、次の点を考慮してください。

- RDS 標準サポート終了日を過ぎた後で、DB インスタンスまたはマルチ AZ DB クラスター を Amazon S3 から復元するには、AWS CLI または RDS API のみを使用できます。restore-db-cluster-from-s3 AWS CLI コマンドの `--engine-lifecycle-support` オプション、または RestoreDBClusterFromS3 RDS API オペレーションの `EngineLifecycleSupport` パラメータを使用します。
- RDS によってデータベースが RDS 延長サポートバージョンに復元されないようにするには、AWS CLI または RDS API で `open-source-rds-extended-support-disabled` を指定します。これにより、関連する RDS 延長サポート料金は発生しません。

  この設定を指定すると、Amazon RDS は復元されたデータベースを、サポートされている新しいメジャーバージョンに自動的にアップグレードします。アップグレードでアップグレード前の検証が失敗した場合、Amazon RDS は安全に RDS 延長サポートエンジンのバージョンにロールバックします。このデータベースは延長サポートモードのままとなり、Amazon RDS ではデータベースを手動でアップグレードするまで RDS 延長サポートの料金が発生します。

  例えば、RDS 延長サポートを使用せずに MySQL 5.7 スナップショットを復元すると、Amazon RDS はデータベースを MySQL 8.0 に自動的にアップグレードしようとします。解決すべき問題が

原因でこのアップグレードが失敗した場合、Amazon RDS によってデータベースが MySQL 5.7 にロールバックされます。Amazon RDS は、問題を解決できるまで RDS 延長サポートでデータベースを維持します。例えば、ストレージ容量不足が原因で、アップグレードが失敗する場合があります。この問題を修正した後に、アップグレードを開始する必要があります。データベースのアップグレードの最初の試行後、Amazon RDS がデータベースのアップグレードを再度試行することはありません。

- RDS 延長サポートはクラスターレベルで設定されます。クラスターのメンバーの RDS 延長サポート設定は、RDS コンソール、AWS CLI の `--engine-lifecycle-support`、RDS API の `EngineLifecycleSupport` で常に同じになります。

詳細については、「MySQL のバージョン」と「Amazon RDS for PostgreSQL のリリースカレンダー」を参照してください。

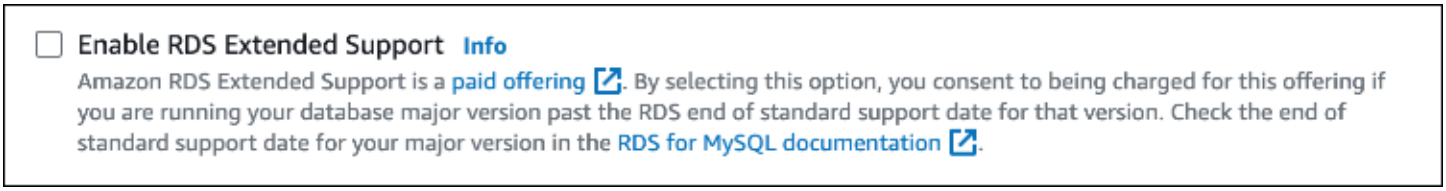
RDS 延長サポートを使用した DB インスタンスまたはマルチ AZ DB クラスター を復元する

RDS 延長サポートバージョンで DB インスタンスまたはマルチ AZ DB クラスターを復元するには、AWS マネジメントコンソール、AWS CLI または RDS API を使用します。

コンソール

DB インスタンスまたはマルチ AZ DB クラスターを復元する際に、[エンジンオプション] セクションで [RDS 延長サポートを有効にする] を選択します。この設定を選択せず、復元するメジャーエンジンバージョンが RDS の標準サポートの終了を過ぎている場合、Amazon RDS は DB インスタンスまたはマルチ AZ DB クラスターを RDS の標準サポートのバージョンに自動的にアップグレードします。

次の画像は、[RDS 延長サポートを有効にする] 設定を示しています。



AWS CLI

restore-db-instance-from-db-snapshot または restore-db-cluster-from-snapshot AWS CLI コマンドを実行する場合は、`--engine-lifecycle-support` オプションに `open-source-rds-extended-support` を指定して RDS 延長サポートを選択します。

RDS 延長サポートに関連する課金を避けたい場合は、`--engine-lifecycle-support` オプションを `open-source-rds-extended-support-disabled` に設定します。このオプションはデフォルトで `open-source-rds-extended-support` に設定されています。

以下の AWS CLI コマンドを使用してこの値を指定することもできます。

- restore-db-cluster-from-s3
- restore-db-cluster-to-point-in-time
- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

RDS API

RestoreDBInstanceFromDBSnapshot または RestoreDBClusterFromSnapshot Amazon RDS API オペレーションを使用する場合は、`EngineLifecycleSupport` パラメータを `open-source-rds-extended-support` に設定して RDS 延長サポートを選択します。

RDS 延長サポートに関連する課金を避けたい場合は、`EngineLifecycleSupport` パラメータを `open-source-rds-extended-support-disabled` に設定します。デフォルトでは、このパラメータは `open-source-rds-extended-support` に設定されます。

以下の RDS API オペレーションを使用してこの値を指定することもできます。

- RestoreDBClusterFromS3
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

DB インスタンスまたはマルチ AZ DB クラスターを復元する方法の詳細については、「DB インスタンスへの復元」の該当 DB エンジンの手順に従ってください。

データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する

ブルー/グリーンデプロイは、本稼働データベース環境を別の同期されたステージング環境にコピーします。Amazon RDS ブルー/グリーンデプロイを使用すると、本稼働環境に影響を与えずにステージング環境のデータベースを変更できます。例えば、DB エンジンのメジャーまたはマイナーバージョンのアップグレードまたはデータベースパラメータの変更をステージング環境で行うことができます。準備ができたら、ステージング環境を新しい本番稼働データベース環境に昇格でき、通常、ダウンタイムは 1 分未満です。

> **Note**
>
> 現在、ブルー/グリーンデプロイは、RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL でのみサポートされています。Amazon Aurora の可用性については、「Amazon Aurora ユーザーガイド」の「データベース更新のために Amazon Aurora ブルー/グリーンデプロイを使用する」を参照してください。

トピック

- Amazon RDS ブルー/グリーンデプロイの概要
- Amazon RDS でのブルー/グリーンデプロイの作成
- Amazon RDS でのブルー/グリーンデプロイの表示
- Amazon RDS でのブルー/グリーンデプロイの切り替え
- Amazon RDS でのブルー/グリーンデプロイの削除

Amazon RDS ブルー/グリーンデプロイの概要

Amazon RDS ブルー/グリーンデプロイを使用すると、データベースに加えた変更をテストしてから、本番稼働環境に実装できます。ブルー/グリーンのデプロイは、本稼働環境をコピーするステージング環境を作成します。ブルー/グリーンデプロイでは、ブルー環境が現在の本稼働環境です。グリーン環境はステージング環境であり、現在の本番環境との同期を維持します。

本稼働環境のワークロードに影響を与えずに、グリーン環境の RDS DB インスタンスに変更を加えることができます。例えば、DB エンジンのメジャーまたはマイナーバージョンのアップグレード、基礎となるファイルシステム設定のアップグレード、またはデータベースパラメータの変更をステージング環境で行うことができます。グリーン環境での変化を徹底的にテストできます。準備ができたら、環境をスイッチオーバーして、グリーン環境を新しい本番稼働環境に移行できます。切り替えには通常 1 分もかからず、データが失われることはなく、アプリケーションを変更する必要もありません。

グリーン環境は本稼働環境のトポロジのコピーであるため、グリーン環境には DB インスタンスが使用する機能が含まれます。これらの機能には、リードレプリカ、ストレージ設定、DB スナップショット、自動バックアップ、パフォーマンスインサイト、拡張モニタリングが含まれます。ブルー DB インスタンスがマルチ AZ DB インスタンスデプロイの場合、グリーン DB インスタンスも、マルチ AZ DB インスタンスデプロイです。

> **Note**
>
> 現在、ブルー/グリーンデプロイは、RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL でのみサポートされています。Amazon Aurora の可用性については、「Amazon Aurora ユーザーガイド」の「Amazon Aurora ブルー/グリーンデプロイの概要」を参照してください。
>
> 特定の条件下では、RDS for PostgreSQL は物理レプリケーションの代わりに論理レプリケーションを使用して、グリーン環境とブルー環境の同期を維持します。詳細については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

Amazon RDS ブルー/グリーンデプロイは、Amazon RDS Proxy とスマートドライバーをサポートしています。これらのソリューションは、トポロジの変更を検出し、DNS の伝播を待たずに接続を新しい本番環境にリダイレクトすることで、スイッチオーバー中のライターノードのアップグレードのダウンタイムを短縮します。

トピック

- リージョンとバージョンの可用性
- Amazon RDS ブルー/グリーンデプロイを使用する利点
- ブルー/グリーンデプロイのワークフロー
- Amazon RDS のブルー/グリーンデプロイオペレーションへのアクセスの承認
- Amazon RDS のブルー/グリーンデプロイの制限と考慮事項
- Amazon RDS のブルー/グリーンデプロイのベストプラクティス

リージョンとバージョンの可用性

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。詳細については、「the section called “ブルー/グリーンデプロイ”」を参照してください。

Amazon RDS ブルー/グリーンデプロイを使用する利点

Amazon RDS ブルー/グリーンデプロイを使用すると、セキュリティパッチを最新の状態に保ち、データベースのパフォーマンスを向上させ、短い予測可能なダウンタイムで新しいデータベース機能を導入できます。ブルー/グリーンデプロイでは、エンジンのメジャーバージョンまたはマイナーバージョンのアップグレードなど、データベース更新のリスクとダウンタイムが軽減されます。

ブルー/グリーンデプロイには次の利点があります。

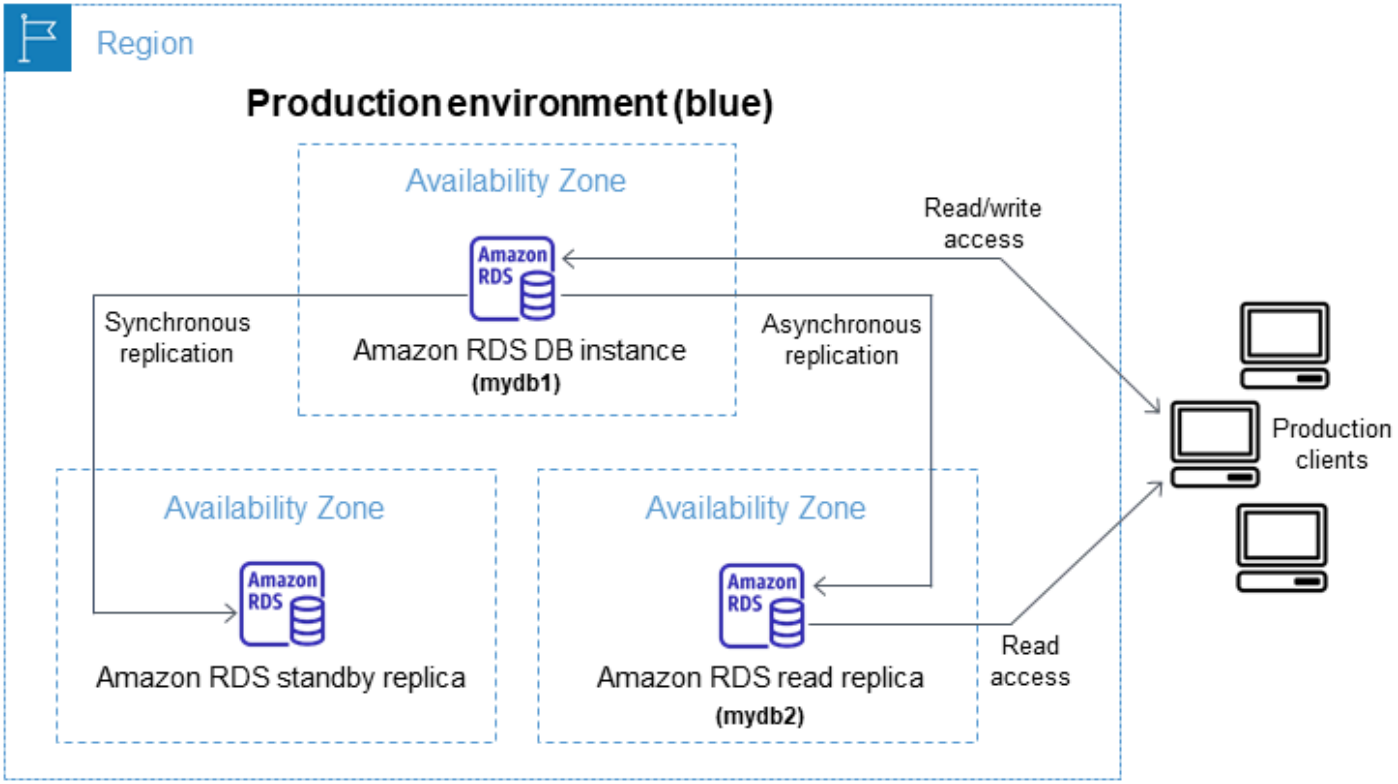
- 本稼働環境に対応したステージング環境を簡単に作成できます。
- データベースの変更を本稼働環境からステージング環境に自動的にレプリケートします。
- 本稼働環境に影響を与えずに、安全なステージング環境でデータベースの変更をテストします。
- データベースパッチとシステムアップデートを最新の状態に保ちます。
- 新しいデータベース機能を実装してテストします。
- アプリケーションを変更することなく、ステージング環境を新しい本稼働環境に切り替えます。
- 組み込みの切り替えガードレールを使用して安全に切り替えることができます。
- 切り替え中のデータ損失をなくします。
- ワークロードにもよりますが、通常は 1 分以内にすばやく切り替えることができます。

ブルー/グリーンデプロイのワークフロー

データベースの更新にブルー/グリーンデプロイを使用する場合は、次の主要なステップを実行します。

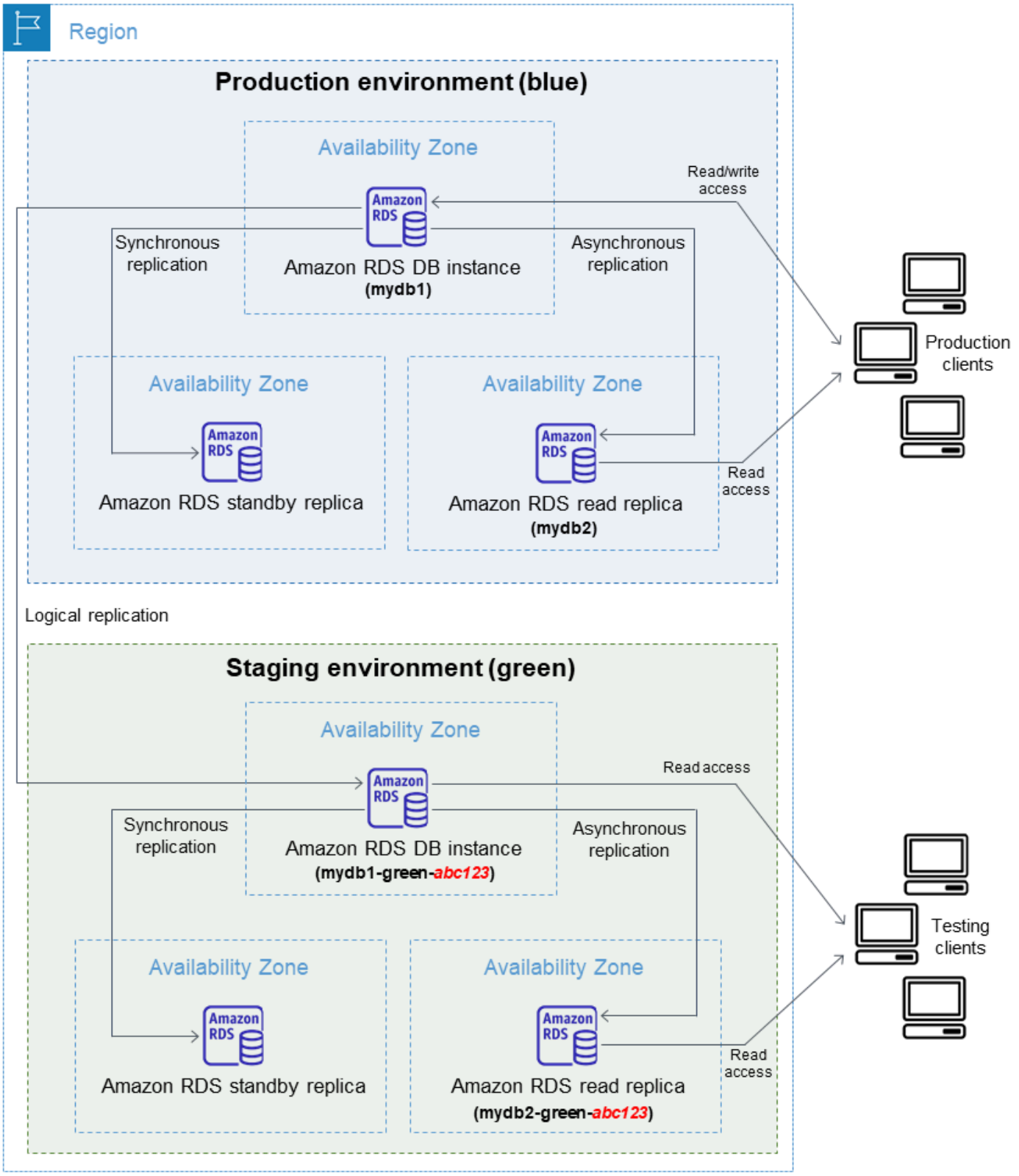
1. 更新が必要な本稼働環境を特定します。

   例えば、この画像の本稼働環境には、マルチ AZ DB インスタンスデプロイ (mydb1) とリードレプリカ (mydb2) があります。



2. ブルー/グリーンデプロイを作成します。手順については、「Amazon RDS でのブルー/グリーンデプロイの作成」を参照してください。

   以下の図は、ステップ 1 の本稼働環境のブルー/グリーンデプロイの例を示しています。ブルー/グリーンデプロイを作成する際、RDS はプライマリ DB インスタンスのトポロジと設定の全体をコピーしてグリーン環境を作成します。コピーされた DB インスタンス名には `-green-random-characters` が付加されます。図のステージング環境には、マルチ AZ DB インスタンスデプロイ (mydb1-green-*abc123*) とリードレプリカ (mydb2-green-*abc123*) が含まれています。



ブルー/グリーンデプロイを作成すると、DB エンジンのバージョンをアップグレードして、グリーン環境の DB インスタンスに別の DB パラメータグループを指定できます。RDS は、ブルー

環境のプライマリ DB インスタンスからグリーン環境のプライマリ DB インスタンスへのレプリケーションも設定します。

ブルー/グリーンデプロイを作成すると、グリーン環境の DB インスタンスはデフォルトで読み取り専用になります。

3. 必要に応じて、ステージング環境をさらに変更します。例えば、グリーン環境の 1 つ以上の DB インスタンスが使用する DB インスタンスクラスを変更できます。

   DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

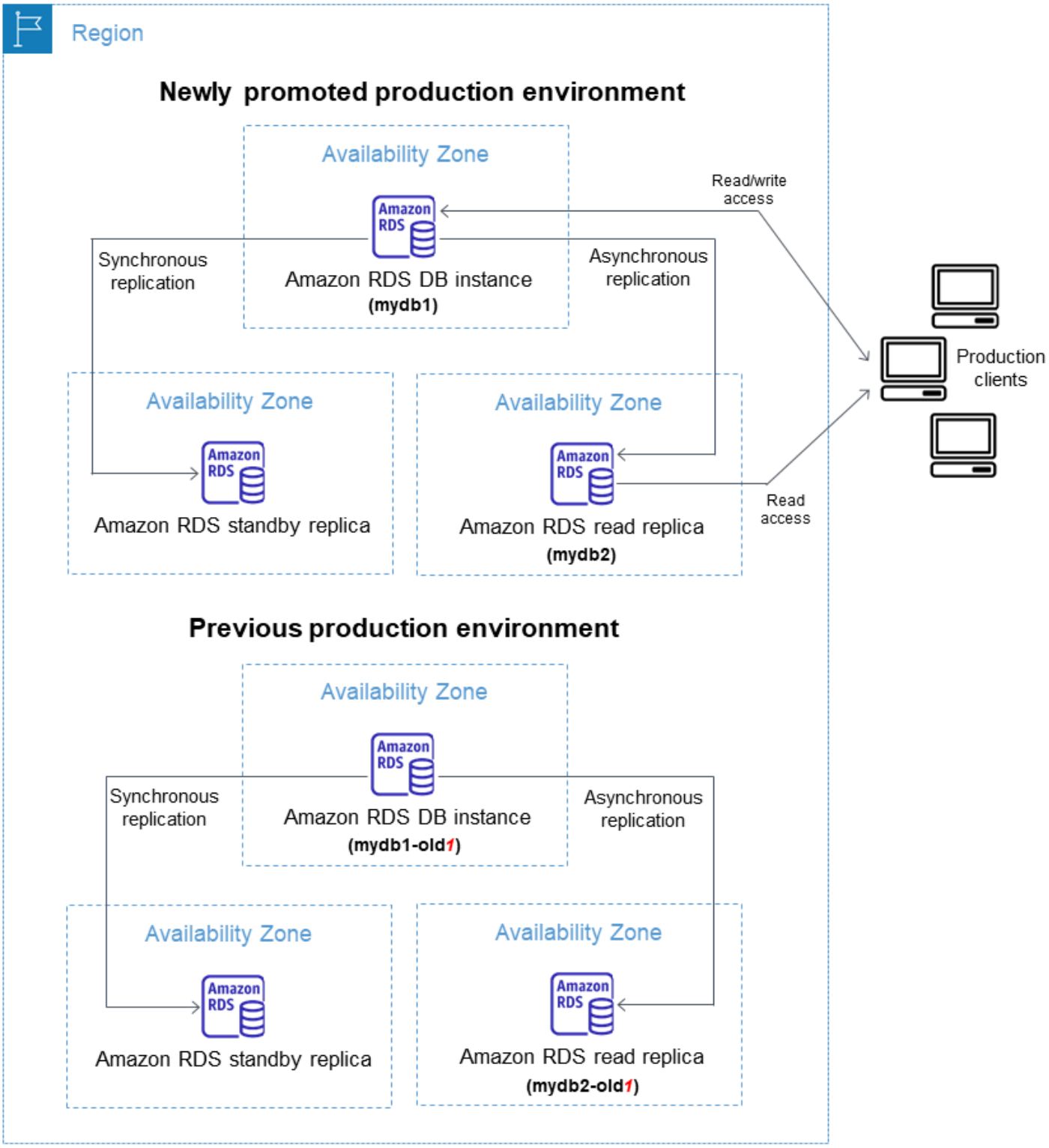
4. ステージング環境をテストします。

   テスト中は、グリーン環境のデータベースを読み取り専用に保つことをお勧めします。グリーン環境ではレプリケーションの競合が発生する可能性があるため、書き込み操作を有効にする場合は注意してください。また、スイッチオーバー後に本稼働データベースに意図しないデータが発生する可能性もあります。RDS for MySQL の書き込みオペレーションを有効にするには、`read_only` パラメータを 1 に設定し、パラメータグループが同期されるまで待ちます。`read_only` は動的パラメータであるため、再起動は必要ありません。同期されたら、`read_only` を 1 から 0 に変更します。論理レプリケーションを使用する RDS for PostgreSQL デプロイの場合、セッションレベルで `default_transaction_read_only` パラメータを `off` に設定します。物理レプリケーションを使用するデプロイの場合、グリーン環境で書き込みオペレーションを有効にすることはできません。

5. 準備ができたら、スイッチオーバーして、ステージング環境を新しい本番稼働環境に移行します。手順については、「Amazon RDS でのブルー/グリーンデプロイの切り替え」を参照してください。

   切り替えによりダウンタイムが発生します。ダウンタイムは通常 1 分未満ですが、ワークロードによってはさらに長くなることもあります。

   次の図は、切り替え後の DB インスタンスを示しています。



切り替え後、グリーン環境にあった DB インスタンスが新しい本稼働 DB インスタンスになります。現在の本番稼働環境内の名前とエンドポイントは、スイッチオーバー後の新しい本番稼働環境に割り当てられるため、アプリケーションを変更する必要はありません。その結果、本稼働ト

ラフィックが新しい本稼働環境に流れるようになります。以前のブルー環境の DB インスタンスは、現在の名前に `-old`*n* を付加することで名前が変更されます (*n* は数字です)。例えば、ブルー環境の DB インスタンスの名前が `mydb1` であるとします。スイッチオーバー後、DB インスタンス名は `mydb1-old1` になります。

図の例では、切り替え中に次の変更が行われます。

- `mydb1-green-abc123` という名前のグリーン環境のマルチ AZ DB インスタンスデプロイが、`mydb1` という名前の本稼働マルチ AZ DB インスタンスデプロイになります。
- `mydb2-green-abc123` という名前が付けられたグリーン環境のリードレプリカが本稼働リードレプリカ `mydb2` になります。
- `mydb1` という名前のブルー環境のマルチ AZ DB インスタンスデプロイは、`mydb1-old1` になります。
- `mydb2` という名前のブルー環境リードレプリカは `mydb2-old1` になります。

6. 不要になったブルー/グリーンデプロイは削除できます。手順については、「Amazon RDS でのブルー/グリーンデプロイの削除」を参照してください。

   切り替え後も以前の本稼働環境は削除されないため、必要に応じてリグレッションテストに使用できます。

Amazon RDS のブルー/グリーンデプロイオペレーションへのアクセスの承認

ブルー/グリーンデプロイに関連する操作を実行するには、ユーザーは必要なアクセス許可があることが求められます。指定されたリソースに対して特定の API 操作を実行するために必要なアクセス許可をユーザーとロールに付与する IAM ポリシーを作成できます。その後、これらのポリシーを、それらのアクセス許可を必要とする IAM アクセス許可セットまたはロールにアタッチできます。詳細については、「Amazon RDS での Identity and Access Management」を参照してください。

ブルー/グリーンデプロイを作成するユーザーには、次の RDS オペレーションを実行するアクセス許可が必要です。

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBInstanceReadReplica`

ブルー/グリーンデプロイを切り替えるユーザーには、次の RDS オペレーションを実行するアクセス許可が必要です。

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBInstance`
- `rds:PromoteReadReplica`

ブルー/グリーンデプロイを削除するユーザーには、次の RDS オペレーション () を実行するアクセス許可が必要です。

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBInstance`

Amazon RDS は、お客様に代わってステージング環境のリソースをプロビジョニングおよび変更します。これらのリソースには、内部で定義された命名規則を使用する DB インスタンスが含まれます。したがって、アタッチされた IAM ポリシーには、`my-db-prefix-*` のような部分的なリソース名パターンを含めることはできません。ワイルドカード (*) のみがサポートされています。一般的に、これらのリソースへのアクセスを制御するには、ワイルドカードではなく、リソースタグやその他のサポートされている属性を使用することをおすすめします。詳細については、「Amazon RDS のアクション、リソース、および条件キー」を参照してください。

Amazon RDS のブルー/グリーンデプロイの制限と考慮事項

Amazon RDS のブルー/グリーンデプロイでは、レプリケーションスロット、リソース管理、インスタンスのサイズ設定、データベースのパフォーマンスへの潜在的な影響などの要素を慎重に検討する必要があります。以下のセクションでは、データベース環境の最小限のダウンタイム、シームレスな移行、効果的な管理を実現するために、デプロイ戦略を最適化するガイダンスを提供します。

トピック

- ブルー/グリーンデプロイの制限事項
- ブルー/グリーンデプロイの考慮事項

ブルー/グリーンデプロイの制限事項

ブルー/グリーンデプロイには、次の制約事項が適用されます。

トピック

- ブルー/グリーンデプロイの一般的な制約事項
- ブルー/グリーンデプロイの RDS for MySQL の制限
- 物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項
- 論理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項

ブルー/グリーンデプロイの一般的な制約事項

ブルー/グリーンデプロイには、次の一般的な制約事項が適用されます。

- ブルー/グリーンデプロイでは、AWS Secrets Manager を使用したマスターユーザーのパスワード管理はサポートされていません。
- ブルーデータベースで専用ログボリューム (DLV) が有効になっている場合は、リードレプリカを含むすべての DB インスタンスで有効にする必要があります。
- 切り替え中、ブルー環境とグリーン環境では Amazon Redshift とのゼロ ETL 統合はできません。最初に統合を削除してから切り替えて、統合を再作成する必要があります。
- ブルー/グリーンデプロイを作成するときには、グリーン環境でイベントスケジューラー (`event_scheduler` パラメーター) を無効にする必要があります。これにより、グリーン環境でイベントが生成されて不整合が発生するのを防ぐことができます。
- 暗号化されていない DB インスタンスを暗号化された DB インスタンスに変更することはできません。さらに、暗号化された DB インスタンスを暗号化されていない DB インスタンスに変更することはできません。
- ブルーの DB インスタンスを、対応するグリーンの DB インスタンスよりも上位のエンジンバージョンに変更することはできません。
- ブルー環境とグリーン環境のリソースは同じ AWS アカウント にある必要があります。
- Amazon RDS Proxy を使用する場合は、ブルー/グリーンデプロイを作成する前に、ブルークラスターをプロキシに登録する必要があります。特定のブルークラスターにブルー/グリーンデプロイが既に存在する場合、そのブルークラスターを Amazon RDS Proxy に登録することはブロックされます。
- Aurora Global Databases では、ブルー/グリーンデプロイの Amazon RDS Proxy はサポートされていません。
- ブルー/グリーンデプロイは、以下の機能ではサポートされていません。
  - カスケードリードレプリカ
  - クロスリージョンリードレプリカ

- CloudFormation
- マルチ AZ DB クラスター配置

  マルチ AZ DB インスタンスのデプロイでは、ブルー/グリーンデプロイがサポートされます。マルチ AZ 配置については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

ブルー/グリーンデプロイの RDS for MySQL の制限

RDS for MySQL ブルー/グリーンデプロイには、以下の制限事項が適用されます。

- ブルーの DB インスタンスを外部のバイナリログレプリカにすることはできません。
- ソースデータベースがカスタムオプショングループに関連付けられている場合、ブルー/グリーンデプロイを作成するときにメジャーバージョンアップグレードを指定することはできません。

  この場合、メジャーバージョンアップグレードを指定せずにブルー/グリーンデプロイを作成できます。その後、グリーン環境のデータベースをアップグレードできます。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。
- ブルー/グリーンデプロイは MySQL 用の AWS JDBC ドライバーをサポートしていません。詳細については、GitHub の「既知の制限事項」を参照してください。

物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項

物理レプリケーションを使用する RDS for PostgreSQL ブルー/グリーンデプロイには、次の制限が適用されます。ブルー/グリーンデプロイが論理レプリケーションではなく物理レプリケーションを使用する状況については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

- グリーン環境の作成後は、メジャーバージョンアップグレードを手動で実行することはできません。
- 物理レプリケーションを使用するブルー/グリーンデプロイは、厳密に読み取り専用であるため、グリーン環境でのスキーマの変更をサポートしていません。
- ブルー DB インスタンスを論理ソース (パブリッシャー) またはレプリカ (サブスクライバー) にすることはできません。
- RDS for PostgreSQL で遅延レプリケーションを設定する場合、ブルー/グリーンデプロイには次の制限があります。

- グリーンソースインスタンス – パラメータグループで設定されていても、`recovery_min_apply_delay parameter` は無視されます。グリーンソースインスタンスの遅延設定は有効になりません。
- グリーンレプリカインスタンス – `recovery_min_apply_delay parameter` は完全にサポートされ、PostgreSQL 設定ファイルに適用されます。遅延設定は、スイッチオーバーワークフロー中に想定どおりに機能します。
- 遅延レプリケーションは、メジャーバージョンアップグレードの RDS ブルー/グリーンデプロイデプロイと互換性がありません。

論理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項

論理レプリケーションを使用する RDS for PostgreSQL のブルー/グリーンデプロイには、次の制限が適用されます。ブルー/グリーンデプロイが物理レプリケーションではなく論理レプリケーションを使用する状況については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

- 。
- ブルー DB インスタンスを論理ソース (パブリッシャー) またはレプリカ (サブスクライバー) にすることはできません。
- ブルー DB インスタンスが外部データラッパー (FDW) 拡張機能の外部サーバーとして設定されている場合は、IP アドレスの代わりにインスタンスエンドポイント名を使用する必要があります。これにより、スイッチオーバー後も設定は機能し続けます。
- ブルー/グリーンデプロイでは、各データベースに論理レプリケーションスロットが必要です。データベースの数が増えると、特に DB インスタンスが十分にスケーリングされていない場合、リソースのオーバーヘッドが増加し、レプリケーションの遅延につながる可能性があります。影響は、データベースワークロードや接続数などの要因によって異なります。影響を軽減するには、DB インスタンスクラスをスケールアップするか、ソースインスタンスのデータベース数を減らすことを検討してください。
- グリーン環境の論理的なレプリケーション適用プロセスは、シングルスレッドです。ブルー環境が大量の書き込みトラフィックを生成した場合、グリーン環境は維持できない可能性があります。これにより、特に書き込みスループットが継続的に高いワークロードでは、レプリケーションの遅延や障害が発生する可能性があります。ワークロードを徹底的にテストしてください。メジャーバージョンのアップグレードが必要で、大量の書き込みワークロードを処理するシナリオでは、AWS Database Migration Service (AWS DMS) などの代替アプローチを検討してください。

- RDS for PostgreSQL で遅延レプリケーションを設定する場合、ブルー/グリーンデプロイには次の制限があります。
  - グリーンソースインスタンス – パラメータグループで設定されていても、`recovery_min_apply_delay parameter` は無視されます。グリーンソースインスタンスの遅延設定は有効になりません。
  - グリーンレプリカインスタンス – `recovery_min_apply_delay parameter` は完全にサポートされ、PostgreSQL 設定ファイルに適用されます。遅延設定は、スイッチオーバーワークフロー中に想定どおりに機能します。
  - 遅延レプリケーションは、メジャーバージョンアップグレードの RDS ブルー/グリーンデプロイデプロイと互換性がありません。
- RDS for PostgreSQL のブルー/グリーンデプロイでは、パーティションテーブルでの新しいパーティションの作成はサポートされていません。新しいパーティションの作成には `CREATE TABLE` などのデータ定義言語 (DDL) オペレーションが含まれますが、これらはブルー環境からグリーン環境にレプリケートされません。ただし、既存のパーティションテーブルとそのデータはグリーン環境にレプリケートされます。
- 以下の制約事項が PostgreSQL 拡張機能に適用されます。
  - ブルー/グリーンデプロイを作成するときは、ブルー環境で `pg_partman` 拡張機能を無効にする必要があります。この拡張機能は `CREATE TABLE` などの DDL オペレーションを実行します。これは、ブルー環境からグリーン環境への論理レプリケーションを中断します。
  - ブルー/グリーンデプロイを作成した後は、すべてのグリーンデータベースで `pg_cron` 拡張機能を無効のままにしておく必要があります。この拡張機能には、スーパーユーザーとして実行されるバックグラウンドワーカーがあり、グリーン環境の読み取り専用設定をバイパスするため、レプリケーションの競合が発生する可能性があります。
  - ブルー/グリーンデプロイを作成するときには、ブルー環境で `pglogical` および `pgactive` 拡張機能を無効にする必要があります。グリーン環境を新しい本番稼働環境にスイッチオーバーしたら、拡張機能を再度有効にできます。また、ブルーデータベースは外部インスタンスの論理サブスクライバーにはなれません。
  - `pgAudit` 拡張機能を使用している場合は、ブルー DB インスタンスとグリーン DB インスタンスの両方のカスタム DB パラメータグループの共有ライブラリ (`shared_preload_libraries`) に残っている必要があります。詳細については、「the section called “pgAudit 拡張機能のセットアップ”」を参照してください。

ブルー/グリーンデプロイの論理レプリケーション固有の制限事項

PostgreSQL には、論理レプリケーションに関連する特定の制約事項があります。これは、 RDS for PostgreSQL DB インスタンスのブルー/グリーンデプロイを作成する際の制約となります。

次の表では、 RDS for PostgreSQL のブルー/グリーンデプロイメントに適用される論理レプリケーションの制約事項について説明しています。詳細については、PostgreSQL の論理レプリケーションドキュメントの「制限」を参照してください。

制限	説明
CREATE TABLE や CREATE SCHEMA などのデータ定義言語 (DDL) ステートメントは、ブルー環境からグリーン環境にはレプリケートされません。	Amazon RDS がブルーの環境で DDL の変更を検出すると、グリーンデータベースはレプリケーションが低下した状態になります。ブルー/グリーンデプロイとすべてのグリーンデータベースを削除してから再作成する必要があります。
GRANT や REVOKE などのデータ制御言語 (DCL) ステートメントは、ブルー環境からグリーン環境にはレプリケートされません。	Amazon RDS PostgreSQL がブルー環境で DCL ステートメントを実行しようとすると、警告メッセージが表示されます。この動作はブルー/グリーンデプロイプロセスの制限であるため、変更できる設定や API はありません。
シーケンスオブジェクトに対する NEXTVAL オペレーションは、ブルー環境とグリーン環境では同期されません。	スイッチオーバー中、 Amazon RDS はグリーン環境のシーケンス値をブルー環境のシーケンス値と一致するようにインクリメントします。シーケンスの量が多い場合は通常、スイッチオーバーを続行できますが、数十万など、非常に多数のシーケンスがあると、完了前にプロセスがタイムアウトする可能性があります。同期により多くの時間を確保できるように、スイッチオーバータイムアウトを増やすことができます。詳細については、「the section called “切り替えタイムアウト”」を参照してください。

制限	説明
ブルー環境のラージオブジェクトは、グリーン環境にはレプリケートされません。これには、既存のラージオブジェクトと、ブルー/グリーンデプロイプロセス中に新しく作成または変更したラージオブジェクトの両方が含まれます。	Amazon RDS が、pg_largeobject システムテーブルに保存されているブルー環境でラージオブジェクトの作成または変更を検出すると、グリーンデータベースはレプリケーションが低下したの状態になります。ブルー/グリーンデプロイとすべてのグリーンデータベースを削除してから再作成する必要があります。
マテリアライズドビューはグリーン環境では自動的に更新されません。	ブルー環境でマテリアライズドビューを更新しても、グリーン環境では更新されません。スイッチオーバー後、REFRESH MATERIALIZED VIEW コマンドを使用して手動で更新することも、更新をスケジュールすることもできます。
UPDATE および DELETE オペレーションは、プライマリキーのないテーブルでは許可されません。	ブルー/グリーンデプロイを作成する前に、すべてのテーブルにプライマリキーがあるか、または REPLICA IDENTITY FULL を使用していることを確認してください。ただし、プライマリキーまたは一意のキーが存在しない場合は、レプリケーションのパフォーマンスに影響するため、REPLICA IDENTITY FULL のみを使用してください。詳細については、PostgreSQL ドキュメントをご参照ください。

ブルー/グリーンデプロイの考慮事項

Amazon RDS は、ブルー/グリーンデプロイのリソースを各リソースの DbiResourceId で追跡します。このリソース ID は、リソースの AWS リージョン 固有のイミュータブルな識別子です。

リソース ID は、DB インスタンス ID とは異なります。それぞれが RDS コンソールのデータベース設定に一覧表示されます。

ブルー/グリーンデプロイを切り替えると、リソースの名前 (インスタンス ID) は変わりますが、各リソースは同じリソース ID を保持します。例えば、DB インスタンス識別子はブルー環境で `mydb` である可能性があります。切り替え後、同じ DB インスタンスの名前が `mydb-old1` になる場合があります。ただし、DB インスタンスのリソース ID は切り替え中に変更されません。そのため、グリーンリソースを新しい本番稼働リソースにスイッチオーバーすると、そのリソース ID は以前に本番稼働環境にあったブルーリソース ID と一致しません。

ブルー/グリーンデプロイをスイッチオーバーしたら、本番稼働リソースで使用していた統合機能やサービスのリソース ID を、新しく移行した本番稼働リソースの ID に更新することを検討してください。具体的には、次のような更新を検討してください。

- RDS API とリソース ID を使用してフィルタリングを実行する場合は、切り替え後にフィルタリングに使用されるリソース ID を調整します。
- CloudTrail をリソースの監査に使用する場合は、切り替え後に新しいリソース ID を追跡するように CloudTrail のコンシューマーを調整します。詳細については、「AWS CloudTrail での Amazon RDS API コールのモニタリング」を参照してください。
- パフォーマンスインサイト API を使用する場合は、切り替え後に API への呼び出しでリソース ID を調整します。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。

  切り替え後に同じ名前のデータベースをモニタリングできますが、切り替え前のデータは含まれていません。
- IAM ポリシーでリソース ID を使用する場合は、必要に応じて新しく移行したリソースのリソース ID を追加してください。詳細については、「Amazon RDS での Identity and Access Management」を参照してください。
- DB インスタンスに IAM ロールが関連付けられている場合は、スイッチオーバー後にそれらを再度関連付けます。アタッチされたロールはグリーンの環境に自動的にコピーされません。
- IAM データベース認証を使用して DB インスタンスを認証する場合は、データベースアクセスに使用する IAM ポリシーの `Resource` 要素の下にブルーとグリーンのデータベースの両方が表示されていることを確認してください。これは、スイッチオーバー後にグリーンのデータベースに接続するために必要です。詳細については、「the section called “IAM データベースアクセス用の IAM ポリシーの作成と使用”」を参照してください。
- AWS Backup を使用してブルー/グリーンデプロイのリソースの自動バックアップを管理する場合は、切り替え後に AWS Backup で使用するリソース ID を調整します。詳細については、「Amazon RDS の自動バックアップの管理に AWS Backup を使用する」を参照してください。

- ブルー/グリーンデプロイに含まれていた DB インスタンスの手動または自動 DB スナップショットを復元する場合は、スナップショットが取られた時間を調べて、正しい DB スナップショットを復元します。詳細については、「DB インスタンスへの復元」を参照してください。
- 以前のブルー環境 DB インスタンスの自動バックアップを記述する場合や、ある時点に復元する場合は、操作のリソース ID を使用します。

  DB インスタンスの名前は切り替え中に変更されるため、以前の名前を `DescribeDBInstanceAutomatedBackups` または `RestoreDBInstanceToPointInTime` 操作に使用することはできません。

  詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。
- ブルー/グリーンデプロイのグリーン環境の DB インスタンスにリードレプリカを追加する場合、切り替え時にブルー環境のリードレプリカが新しいリードレプリカに置き換えられることはありません。ただし、新しいリードレプリカは、切り替え後も新しい本稼働環境に保持されます。
- 切り替え後は、ブルー環境のチェックポイントがグリーン環境で無効であるため、AWS Database Migration Service (AWS DMS) レプリケーションタスクを再開できません。レプリケーションを続行するには、新しいチェックポイントで DMS タスクを再作成する必要があります。
- ブルー/グリーンデプロイのグリーン環境の DB インスタンスを削除すると、ブルー/グリーンデプロイで代わりになる新しい DB インスタンスを作成することはできません。

  削除した DB インスタンスと同じ名前と Amazon リソースネーム (ARN) で新しい DB インスタンスを作成した場合、`DbiResourceId` が異なるため、グリーン環境には含まれません。

  グリーン環境で DB インスタンスを削除すると、次のような動作になります。

  - ブルー環境に同じ名前の DB インスタンスが存在する場合、グリーン環境の DB インスタンスに切り替わりません。この DB インスタンスの名前は、DB インスタンス名に `-old`*n* を付加することによって変更されません。
  - ブルー環境の DB インスタンスを参照するアプリケーションは、切り替え後も同じ DB インスタンスを引き続き使用します。

  同じ動作が DB インスタンスとリードレプリカにも当てはまります。
- アクセスコントロールまたは運用管理にリソースタグを使用する場合は、スイッチオーバーまでタグの変更がブルー環境とグリーン環境の間で同期されないことを理解する必要があります。ブルー/グリーンデプロイを作成すると、ブルー環境のタグがグリーン環境にコピーされます。作成後、いずれかの環境に行ったタグの変更は自動的に同期されません。スイッチオーバー中、ブルー環境タグはグリーン環境内のすべてのタグを置き換えます。ブルー/グリーンデプロイを作成する

前にブルー環境にすべての必要なタグを適用するか、スイッチオーバー後に新しい本番環境に必要なタグを再適用します。タグの詳細については、 Amazon RDS リソースのタグ付けを参照してください。

Amazon RDS のブルー/グリーンデプロイのベストプラクティス

ブルー/グリーンデプロイのベストプラクティスを以下に示します。

トピック

- ブルー/グリーンデプロイの一般的なベストプラクティス
- RDS for MySQL のブルー/グリーンデプロイのベストプラクティス
- RDS for MySQL のブルー/グリーンデプロイのベストプラクティス
- ブルー/グリーンデプロイの PostgreSQL レプリケーション方法

ブルー/グリーンデプロイの一般的なベストプラクティス

ブルー/グリーンデプロイを作成するときは、次の一般的なベストプラクティスを考慮してください。

- 切り替える前に、DB インスタンスをグリーン環境で十分にテストしてください。
- グリーン環境のデータベースは読み取り専用のまま維持してください。グリーン環境ではレプリケーションの競合が発生する可能性があるため、書き込み操作の有効にする場合は注意してください。また、スイッチオーバー後に本稼働データベースに意図しないデータが発生する可能性もあります。
- ブルー/グリーンデプロイを使用してスキーマの変更を実装する場合は、レプリケーション互換の変更のみを行ってください。

  例えば、ブルーデプロイからグリーンデプロイへのレプリケーションを中断することなく、テーブルの最後に新しい列を追加することができます。ただし、列名の変更やテーブル名の変更などのスキーマの変更は、グリーンデプロイへのレプリケーションを中断させます。

  レプリケーションと互換性のある変更の詳細については、MySQL ドキュメントの「ソースとレプリカのテーブル定義が異なるレプリケーション」と PostgreSQL 論理レプリケーションドキュメントの「制限」を参照してください。

> **Note**
>
> この制限は、物理レプリケーションを使用する RDS for PostgreSQL ブルー/グリーンデプロイには適用されません。詳細については、「the section called “物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項”」を参照してください。

- ブルー/グリーンデプロイを作成した後、必要に応じて遅延読み込みを処理します。切り替える前に、データの読み込みが完了していることを確認してください。詳細については、「ブルー/グリーンデプロイの遅延読み込みとストレージの初期化」を参照してください。
- ブルー/グリーンデプロイメントを切り替えるときは、切り替えのベストプラクティスに従ってください。詳細については、「the section called “切り替えのベストプラクティス”」を参照してください。

RDS for MySQL のブルー/グリーンデプロイのベストプラクティス

RDS for MySQL DB インスタンスからブルー/グリーンデプロイを作成するときは、次のベストプラクティスを考慮してください。

- MyISAM など、レプリケーション用に最適化されていない非トランザクションストレージエンジンの使用は避けてください。
- リードレプリカとグリーン環境をバイナリログレプリケーション用に最適化します。DB エンジンでサポートされている場合は、ブルー/グリーンデプロイを作成する前に、GTID、並列、クラッシュセーフレプリケーションを有効にして、データの整合性と耐久性を確保します。詳細については、「GTID ベースレプリケーションを使用する」を参照してください。
- グリーン環境でレプリカの遅延が発生した場合は、以下の点を考慮します。
  - グリーン DB パラメータグループで `innodb_flush_log_at_trx_commit` パラメータを一時的に 2 に設定します。レプリケーションが追い付いたら、スイッチオーバーの前にデフォルト値 1 に戻します。一時的なパラメータ値で予期しないシャットダウンやクラッシュが発生した場合は、検出されないデータ破損を避けるためにグリーン環境を再構築します。詳細については、
  - 書き込みレイテンシーを短縮し、レプリケーションスループットを向上させるには、一時的にグリーンマルチ AZ DB インスタンスをシングル AZ DB インスタンスに変更します。スイッチオーバーの直前にマルチ AZ を再度有効にします。

RDS for MySQL のブルー/グリーンデプロイのベストプラクティス

上記の一般的なベストプラクティスとエンジン固有のベストプラクティスに加えて、RDS for MySQL DB インスタンス の以下のベストプラクティスを検討してください。

- 次の CloudWatch メトリクスをモニタリングして、本番環境でのアクティビティが少ない期間を特定します。
  - `DatabaseConnections`
  - `ActiveTransactions`

  ブルー/グリーンスイッチオーバーは、計画されたメンテナンスウィンドウ中またはアクティビティが少ない期間にスケジュールします。
- ブルー/グリーンスイッチオーバー期間は、ワークロードとセカンダリリージョンの数によって異なります。ブルー/グリーンスイッチオーバーを開始すると、サービスはレプリカの遅延がゼロになるまで待機してから続行します。スイッチオーバーを開始する前に、レプリカの遅延を確認することをお勧めします。
- グリーン環境のデフォルト以外の DB パラメータまたは DB クラスターパラメータグループを使用する場合は、ブルー/グリーンデプロイを開始する前に、すべてのセカンダリリージョンで同じ名前の必要なパラメータグループを作成します。

RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス

RDS for PostgreSQL DB インスタンスからブルー/グリーンデプロイを作成するときは、次のベストプラクティスを考慮してください。

トピック

- RDS for PostgreSQL のブルー/グリーンデプロイの一般的なベストプラクティス
- 物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス
- 論理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス

RDS for PostgreSQL のブルー/グリーンデプロイの一般的なベストプラクティス

RDS for PostgreSQL DB インスタンスからブルー/グリーンデプロイを作成するときは、次の一般的なベストプラクティスを考慮してください。

- ブルー/グリーンデプロイを作成するときには、すべての PostgreSQL 拡張機能を最新バージョンに更新してください。詳細については、「the section called “PostgreSQL 拡張機能のアップグレード”」を参照してください。
- トランザクションの実行時間が長いと、レプリカの遅延が大きくなる可能性があります。レプリカの遅延を減らすには、以下を検討します。
  - グリーン環境がブルー環境に追いつくまで、遅延が発生する可能性があり長時間実行されるトランザクションを減らします。
  - グリーン環境がブルー環境に追いつくまで、ブルー環境での一括オペレーションを減らします。
  - ブルー/グリーンデプロイを作成する前に、ビジーテーブルで手動バキュームフリーズオペレーションを開始します。
  - PostgreSQL バージョン 12 以降では、大きなテーブルまたはビジーテーブルで `index_cleanup` パラメータを無効にして、ブルーデータベースの通常のメンテナンスのレートを高くします。詳細については、「the section called “テーブルをできるだけ早くバキューム処理する”」を参照してください。

    > **Note**
    >
    > バキューム中にインデックスのクリーンアップを定期的にスキップすると、インデックスが肥大化し、スキャンパフォーマンスが低下する可能性があります。ベストプラクティスとして、このアプローチはブルー/グリーンデプロイの使用時にのみ使用してください。デプロイが完了したら、インデックスの定期的なメンテナンスとクリーンアップを再開することをお勧めします。
- レプリケーションが遅い場合、送信者と受信者が頻繁に再起動し、同期が遅れる可能性があります。送信者と受信者が再起動しないようにするには、ブルー環境で `wal_sender_timeout` パラメータを `0` に設定し、グリーン環境で `wal_receiver_timeout` パラメータを `0` に設定してタイムアウトを無効にします。
- ブルー環境からログ先行書き込み (WAL) セグメントが削除されないようにするには、PostgreSQL バージョン 13 以前では `wal_keep_segments` パラメータを 15625 に設定します。バージョン 14 以降では、十分な空きストレージ容量がある場合は、`wal_keep_size` パラメータを 1 TiB に設定します。

物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス

物理レプリケーションを使用して、Amazon RDS はソース DB インスタンスのリードレプリカを作成します。関連するパラメータ、モニタリング、チューニング、トラブルシューティングについては、「the section called “RDS for PostgreSQL のリードレプリカ”」を参照してください。

ブルー/グリーンデプロイが論理レプリケーションではなく物理レプリケーションを使用する状況については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

論理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス

論理レプリケーションを使用するブルー/グリーンデプロイを作成するときは、次のベストプラクティスを考慮してください。ブルー/グリーンデプロイが物理レプリケーションではなく論理レプリケーションを使用する状況については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

- データベースが に十分な空きメモリがある場合は、ブルー環境の `logical_decoding_work_mem` DB パラメータの値を増やします。そうすることで、ディスク上でのデコード回数が減り、代わりにメモリを消費します。詳細については、PostgreSQL ドキュメントをご参照ください。
  - `ReplicationSlotDiskUsage` CloudWatch メトリクスを使用して、ディスクに書き込まれるトランザクションのオーバーフローをモニタリングできます。このメトリクスは、レプリケーションスロットのディスク使用状況に関するインサイトを提供し、トランザクションデータがメモリ容量を超えてディスクに保存されるタイミングを特定するのに役立ちます。`FreeableMemory` CloudWatch メトリクスを使用して空きメモリをモニタリングできます。詳細については、「the section called “Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス”」を参照してください。
  - RDS for PostgreSQL バージョン 14 以降では、`pg_stat_replication_slots` システムビューを使用して論理オーバーフローファイルのサイズをモニタリングできます。
- `aws_s3` 拡張機能を使用している場合は、グリーン環境が作成された後に、必ず IAM ロールを通じてグリーン DB インスタンスに Amazon S3 へのアクセスを許可してください。これにより、インポートコマンドとエクスポートコマンドは、スイッチオーバー後も機能し続けることができます。手順については、「the section called “Amazon S3 バケットへのアクセスを設定する”」を参照してください。
- UPDATE ステートメントと DELETE ステートメントのパフォーマンスを確認し、WHERE 句で使用する列にインデックスを作成することで、これらのクエリを最適化できるかどうかを評価しま

す。これにより、グリーン環境でオペレーションを再生するときのパフォーマンスを向上させることができます。

- トリガーを使用する場合は、名前が「rds」で始まる pg_catalog.pg_publication、pg_catalog.pg_subscription、pg_catalog.pg_replicatio オブジェクトの作成、更新、削除を妨げないようにしてください。
- グリーン環境でより上位のエンジンバージョンを指定する場合は、すべてのデータベースに対して ANALYZE オペレーションを実行して pg_statistic テーブルを更新します。Optimizer の統計情報はメジャーバージョンのアップグレード時に転送されないため、パフォーマンスの問題を回避するためにすべての統計情報を再生成する必要があります。メジャーバージョンアップグレード時のその他のベストプラクティスについては、「the section called “メジャーバージョンのアップグレードを実施する方法”」を参照してください。
- ソースでトリガーを使用してデータを操作している場合は、トリガーを ENABLE REPLICA または ENABLE ALWAYS として設定しないでください。レプリケーションシステムが変更を伝播してトリガーを実行し、重複を引き起こします。

ブルー/グリーンデプロイの PostgreSQL レプリケーション方法

Amazon RDS for PostgreSQL は、ブルー/グリーンデプロイに主に物理レプリケーションを使用します。ただし、ブルー/グリーンデプロイの作成時にメジャーバージョンアップグレードをリクエストし、ソース DB インスタンスが次の表に示す PostgreSQL バージョンのいずれかを実行する場合、Amazon RDS は代わりに論理レプリケーションを使用します。

次の表は、Amazon RDS が PostgreSQL ブルー/グリーンデプロイで物理レプリケーションを使用する状況と論理レプリケーションを使用する状況を示しています。

ソース PostgreSQL DB インスタンスのバージョン	ブルー/グリーンデプロイのアップグレードアクション	レプリケーション方法
• 16.1 以降のすべてのメジャーバージョンとマイナーバージョン • 15.4 以降の 15 バージョン • 14.9 以降の 14 バージョン • 13.12 以降の 13 バージョン	メジャーバージョンアップグレード (ブルーより上位のメジャーエンジンバージョンのグリーンインスタンス)	論理レプリケーション

ソース PostgreSQL DB インスタンスのバージョン	ブルー/グリーンデプロイのアップグレードアクション	レプリケーション方法
• 12.16 以降の 12 バージョン • 11.21 以降の 11 バージョン		
サポートされている全バージョン	マイナーバージョンアップグレード、またはアップグレードなし (ブルーと同じメジャーエンジンバージョンのグリーンインスタンス)	物理レプリケーション。

> **Note**
>
> メジャーバージョンのアップグレードは、ソース RDS for PostgreSQL バージョン 15.3 以前、14.8 以前、13.11 以前、12.15 以前、11.20 以前のブルー/グリーンデプロイではサポートされていません。

物理レプリケーションと論理レプリケーションを使用するブルー/グリーンデプロイの制限については、以下のセクションを参照してください。

- the section called “物理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項”
- the section called “論理レプリケーションを使用した RDS for PostgreSQL のブルー/グリーンデプロイの制限事項”

Amazon RDS でのブルー/グリーンデプロイの作成

ブルー/グリーンデプロイを作成するときには、デプロイにコピーするソース DB インスタンスを指定します。選択する DB インスタンスは本番 DB インスタンスであり、ブルー環境のプライマリ DB インスタンスになります。この DB インスタンスはグリーン環境にコピーされ、RDS はブルー環境の DB インスタンスからグリーン環境の DB インスタンスへのレプリケーションを設定します。

RDS は、ブルー環境のトポロジと設定をステージングエリアにコピーします。ブルー DB インスタンスにリードレプリカがある場合、グリーンインスタンスのレプリカとしてコピーされます。すべてのグリーンレプリカに割り当てられたストレージはグリーンプライマリインスタンスと一致し、他のストレージパラメータはブルーレプリカから継承されます。

ブルー DB インスタンスがマルチ AZ DB インスタンスデプロイの場合、グリーン DB インスタンスは、マルチ AZ DB インスタンスデプロイとして作成されます。

トピック

- ブルー/グリーンデプロイの準備
- ブルー/グリーンデプロイを作成するときの変更を指定
- ブルー/グリーンデプロイの遅延読み込みとストレージの初期化
- ブルー/グリーンデプロイの作成
- ブルー/グリーンデプロイを作成するための設定

ブルー/グリーンデプロイの準備

DB インスタンスが実行しているエンジンに応じて、ブルー/グリーンデプロイを作成する前に実行する必要がある特定の手順があります。

トピック

- RDS for MySQL または RDS for MariaDB DB インスタンスをブルー/グリーンデプロイ用に準備する
- 物理レプリケーションを使用する RDS for PostgreSQL DB インスタンスをブルー/グリーンデプロイ用に準備する
- 論理レプリケーションを使用する RDS for PostgreSQL DB インスタンスをブルー/グリーンデプロイ用に準備する

RDS for MySQL または RDS for MariaDB DB インスタンスをブルー/グリーンデプロイ用に準備する

RDS for MySQL または RDS for MariaDB DB インスタンスのブルー/グリーンデプロイを作成する前に、自動バックアップを有効にする必要があります。手順については、「the section called “自動バックアップの有効化”」を参照してください。

物理レプリケーションを使用する RDS for PostgreSQL DB インスタンスをブルー/グリーンデプロイ用に準備する

物理レプリケーションを使用する RDS for PostgreSQL のブルー/グリーンデプロイを作成する前に、必ず以下を実行してください。物理レプリケーションまたは論理レプリケーションを使用するバージョンのリストについては、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

- DB インスタンスの自動バックアップを有効にします。手順については、「the section called “自動バックアップの有効化”」を参照してください。
- DB インスタンスが外部レプリケーションのソースでもターゲットでもないことを確認します。詳細については、「the section called “一般的な制限事項”」を参照してください。

論理レプリケーションを使用する RDS for PostgreSQL DB インスタンスをブルー/グリーンデプロイ用に準備する

論理レプリケーションを使用する RDS for PostgreSQL のブルー/グリーンデプロイを作成する前に、必ず以下を実行してください。論理レプリケーションまたは物理レプリケーションを使用するバージョンのリストについては、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

- 論理レプリケーション (`rds.logical_replication`) をオンにしたカスタム DB パラメータグループにインスタンスを関連付けます。ブルー環境からグリーン環境へのレプリケーションには、論理レプリケーションが必要です。手順については、「the section called “DB パラメータグループのパラメータの変更”」を参照してください。

  ブルー/グリーンデプロイではデータベースごとに少なくとも 1 つのバックグラウンドワーカーが必要なため、ワークロードに応じて次の構成設定を調整してください。各設定を調整する手順については、PostgreSQL ドキュメントの「構成設定」を参照してください。

  - `max_replication_slots`
  - `max_wal_senders`
  - `max_logical_replication_workers`
  - `max_worker_processes`

  論理レプリケーションを有効にしてすべての設定オプションを設定したら、DB インスタンスを再起動して変更を有効にします。ブルー/グリーンデプロイでは、DB インスタンスが DB パラメータ

グループと同期している必要があり、同期していない場合は作成に失敗します。詳細については、「the section called “DB インスタンスを再起動する”」を参照してください。

- DB インスタンスが外部レプリケーションのソースでもターゲットでもないことを確認します。詳細については、「the section called “一般的な制限事項”」を参照してください。
- DB インスタンスのすべてのテーブルにプライマリキーがあることを確認してください。PostgreSQL の論理レプリケーションでは、プライマリキーのないテーブルに対する UPDATE または DELETE オペレーションは許可されません。
- RDS for PostgreSQL は PostgreSQL のネイティブ論理レプリケーションを使用し、先行書き込みログ (WAL) セグメントを、グリーン環境で再生するまでブルーインスタンスに保存します。ブルー/グリーンデプロイを作成する前に、以下のメトリクスをチェックして、ブルーインスタンスに十分な容量があることを確認します。
  - `FreeStorageSpace`
  - `TransactionLogsGeneration`
  - `TransactionLogsDiskUsage`
  - `OldestReplicationSlotLag`

  ブルーインスタンスに必要な追加のストレージを見積もるには、ワークロードのピーク期間中の `TransactionLogsGeneration` CloudWatch メトリクスをモニタリングします。例えば、ワークロードが 24 時間で 100 GB の WAL データを生成する場合は、1 日分の WAL セグメントを収容するために、少なくとも 100 GB の追加ストレージがあることを確認してください。詳細については、「DB インスタンスでのメトリクスのモニタリング」を参照してください。

ブルー/グリーンデプロイを作成するときの変更を指定

ブルー/グリーンデプロイを作成するときに、グリーン環境の DB インスタンスに次の変更を加えることができます。

デプロイ後に、グリーン環境の DB インスタンスに他の変更を加えることができます。例えば、より高いエンジンバージョンまたは別のパラメータグループを指定できます。

DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

トピック

- より高いエンジンバージョンを指定する
- 別の DB パラメータグループを指定する

- ストレージとパフォーマンスの設定を変更する
- RDS Optimized Writes の有効化
- ストレージ設定のアップグレード

より高いエンジンバージョンを指定する

DB エンジンのアップグレードをテストする場合は、上位のエンジンバージョンを指定できます。スイッチオーバー時に、データベースは指定したメジャーまたはマイナー DB エンジンバージョンにアップグレードされます。

別の DB パラメータグループを指定する

パラメータの変更がグリーン環境の DB インスタンスにどのように影響するかをテストしたり、アップグレードの場合に新しいメジャー DB エンジンバージョンのパラメータグループを指定したりできます。

別の DB パラメータグループを指定した場合、指定した DB パラメータグループはグリーン環境内のすべての DB インスタンスに関連付けられます。別のパラメータグループを指定しなかった場合、グリーン環境の各 DB インスタンスは、対応するブルー DB インスタンスのパラメータグループに関連付けられます。

ストレージとパフォーマンスの設定を変更する

グリーン環境でストレージとパフォーマンスの設定を調整して、リソースの割り当てを最適化します。これらの設定には、割り当てストレージ、プロビジョンド IOPS、ストレージタイプ、ストレージスループット (gp3 ストレージ用) が含まれます。

グリーン DB インスタンスのストレージタイプは、gp2、gp3、io1、または io2 に変更できます。gp3 ストレージでは、ストレージスループットを調整して、需要の高いワークロードでデータ転送パフォーマンスを向上させたり、負荷の低いアプリケーションのコストを削減したりすることもできます。詳しくは、「Amazon RDS DB インスタンスストレージ」を参照してください。

グリーン環境で割り当てストレージを増減することもできます。ただし、ストレージの削減は、ターゲットの割り当てストレージが現在のストレージ使用量よりも 20% 以上多い場合にのみ発生します。割り当てストレージを減らすと、Amazon RDS はストレージ設定のアップグレードを開始します。詳細については、「the section called “ストレージ設定のアップグレード”」を参照してください。最小ターゲットストレージは次のように計算されます。

```
Minimum Target Storage = Total Allocated Storage × Current Apparent Utilization × 1.2
```

1. 合計ストレージ割り当て: RDS コンソールに表示される、DB インスタンスにプロビジョニングされたストレージ容量。
2. 現在の表面上の使用率: 使用中のストレージ割り当ての割合。これを見積るには、パフォーマンスインサイト (Database Insights) または拡張モニタリングの `os.fileSys.usedPercent` メトリクスを使用します。

> **Note**
>
> ストレージ使用率は時間の経過に伴って変動するため、削減プロセス中の潜在的な増加を考慮して、計算された最小値をわずかに上回るようにターゲットストレージを設定することをお勧めします。

ブルー DB インスタンスがマグネティックストレージを使用している場合は、割り当てストレージを増減するために、グリーン DB インスタンスを汎用またはプロビジョンド IOPS ストレージタイプに変更する必要があります。

RDS Optimized Writes の有効化

ブルー/グリーンデプロイを使用して RDS Optimized Writes をサポートする DB インスタンスクラスにアップグレードできます。RDS Optimized Writes は、サポートされている DB インスタンスクラスで作成されたデータベースでのみ有効にできます。したがって、このオプションでは、サポートされている DB インスタンスクラスを使用するグリーンデータベースが作成され、グリーン DB インスタンスで RDS Optimized Writes を有効にできます。

RDS Optimized Writes をサポートしていない DB インスタンスクラスからサポートする DB インスタンスクラスにアップグレードする場合は、グリーン DB インスタンスのストレージ設定もアップグレードする必要があります。詳細については、「the section called “ストレージ設定のアップグレード”」を参照してください。

アップグレードできるのは、プライマリグリーン DB インスタンスの DB インスタンスクラスだけです。デフォルトでは、グリーン環境のリードレプリカはブルー環境の DB インスタンス設定を継承します。グリーン環境が正常に作成されたら、グリーン環境のリードレプリカの DB インスタンスクラスを手動で変更する必要があります。

ブルー DB インスタンスのエンジンバージョンとインスタンスクラスによっては、インスタンスクラスのアップグレードがサポートされていない場合があります。DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

ストレージ設定のアップグレード

ブルーデータベースが最新のストレージ設定になっていない場合、RDS はグリーン DB インスタンスを古いストレージ設定 (32 ビットファイルシステム) から希望の設定に移行できます。RDS ブルー/グリーンデプロイを使用すると、古い 32 ビットファイルシステムのストレージとファイルサイズのスケーリング制限を克服できます。さらに、指定した DB インスタンスクラスが Optimized Writes をサポートしている場合、この設定により Optimized Writes に対応するようにストレージ設定が変更されます。

> **Note**
>
> ストレージ設定のアップグレードは I/O の負荷が高い操作であり、ブルー/グリーンデプロイでは作成時間が長くなります。ブルー DB インスタンスがプロビジョンド IOPS SSD (io1 または io2 Block Express) ストレージを使用し、インスタンスサイズが 4xlarge 以上のグリーン環境をプロビジョニングした場合、ストレージのアップグレードプロセスは速くなります。汎用 SSD (gp2) ストレージを含むストレージをアップグレードすると、I/O クレジットバランスを使い切る可能性があり、よってよりアップグレード時間が長くなります。詳細については、「the section called “DB インスタンスストレージ”」を参照してください。
> ストレージのアップグレード中、グリーン DB インスタンスは一時的に使用できなくなりますが、ブルー DB インスタンスは引き続き使用できます。この間、レプリケーションは一時停止します。ブルーインスタンスでストレージをモニタリングし、ストレージが 90% に達した場合はスケールすることを検討してください。グリーンインスタンスはアップグレード後に自動的に 10% スケールします。

このオプションは、ブルーデータベースが最新のストレージ設定になっていない場合、または同じリクエスト内で DB インスタンスクラスを変更する場合にのみ使用できます。ストレージ設定は、ブルー/グリーンデプロイを最初に作成するときにのみアップグレードできます。

ブルー/グリーンデプロイの遅延読み込みとストレージの初期化

ブルー/グリーンデプロイを作成すると、Amazon RDS は DB スナップショットから復元することにより、グリーン環境にプライマリ DB インスタンスを作成します。作成後、グリーン DB インスタンスとそのリードレプリカは引き続きバックグラウンドでデータをロードします。このプロセスは遅延読み込みと呼ばれます。

遅延読み込みは、アプリケーションがリクエストするデータブロックのみをロードします。まだロードされていないデータにアクセスしようとすると、Amazon EBS は Amazon S3 からすぐにデータを

取得し、残りのデータはバックグラウンドでロードし続けます。詳細については、「Amazon EBS スナップショット」を参照してください。

ボリューム全体のパフォーマンスを加速するため、Amazon RDS はグリーン環境ボリューム内のすべてのブロックを読み取るストレージ初期化を提供します。Amazon EBS は Amazon S3 からブロックをプロアクティブにダウンロードし、最初の使用時から最大ボリュームパフォーマンスを実現します。ストレージの初期化は完全にバックグラウンドで行われるため、DB インスタンスの可用性、パッチ適用やアップグレードなどの進行中のアクティビティには影響しません。

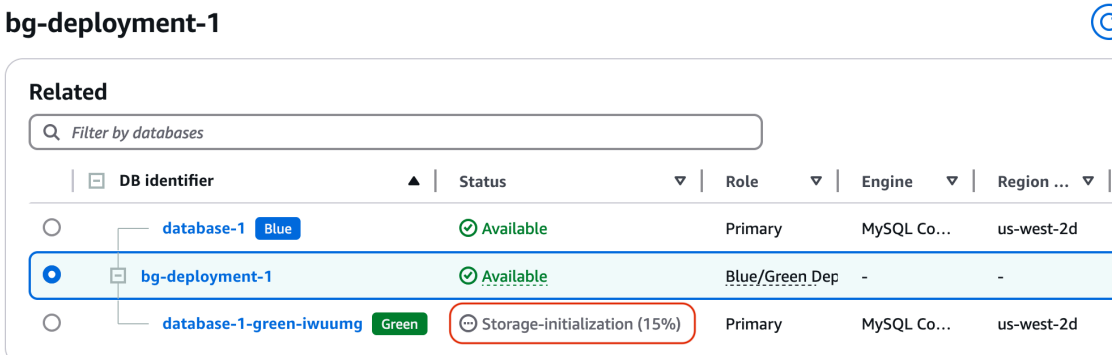
ストレージの初期化は、ボリュームタイプが gp2、gp3、io1、io2 のブルー/グリーンデプロイのインスタンスでのみ使用できます。t3 および t4 ファミリーを除くすべてのインスタンスクラスをサポートします。シングル AZ デプロイのグリーン DB インスタンスをマルチ AZ DB インスタンスデプロイに変更すると、ストレージの初期化にはマルチ AZ 設定のセカンダリノードが含まれます。

ストレージの初期化中、インスタンスはデータベースオペレーションで完全に利用可能ですが、初期化が完了するまでストレージのフルパフォーマンスに達しない場合があります。ストレージの初期化中、インスタンス全体のステータスは「ストレージの初期化」に変わり、進行状況インジケータには DB インスタンスのすべてのボリュームの最小初期化レベルが反映されます。

コンソール、AWS CLI、または Amazon RDS API を使用して、ストレージの初期化をモニタリングします。ブルー/グリーンデプロイは、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用してボリュームごとの進行状況を観察できるいくつかのワークフローの 1 つです。対象となるすべてのワークフローとフィールドセマンティクスの詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

Console

AWS マネジメントコンソールに、ストレージの初期化の進行状況と DB インスタンスのステータスが表示されます。



AWS CLI

AWS CLI では、describe-db-instances コマンドを使用してストレージの初期化をモニタリングできます。StorageOperationStatus および StorageOperationPercentProgress フィールドは、ボリュームごとの状態と進行状況を報告します。レガシーの PercentProgress フィールドは、ストレージの初期化中にインスタンスレベルの進捗状況を引き続き報告します。

```
aws rds describe-db-instances --db-instance-identifier my-db-instance

{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "my-db-instance",
            "DBInstanceClass": "db.m5.2xlarge",
            "Engine": "postgres",
            "DBInstanceStatus": "storage-initialization",
            ...
            "StorageOperationStatus": "Initializing",
            "StorageOperationPercentProgress": 34,
            "PercentProgress": "34"
        }
    ]
}
```

Amazon RDS API

Amazon RDS API では、DescribeDBInstances アクションを呼び出してストレージ初期化のステータスを取得します。レスポンスの StorageOperationStatus フィールドと StorageOperationPercentProgress フィールドを確認します。

バックグラウンド初期化ジョブの進行に伴って進行状況インジケータが更新され、フルストレージ初期化が完了する前にストレージの準備状況を追跡できます。ストレージの初期化により、グリーン DB インスタンスが完全に動作すると最適化されたパフォーマンスが実現されます。

ブルー/グリーンデプロイの作成

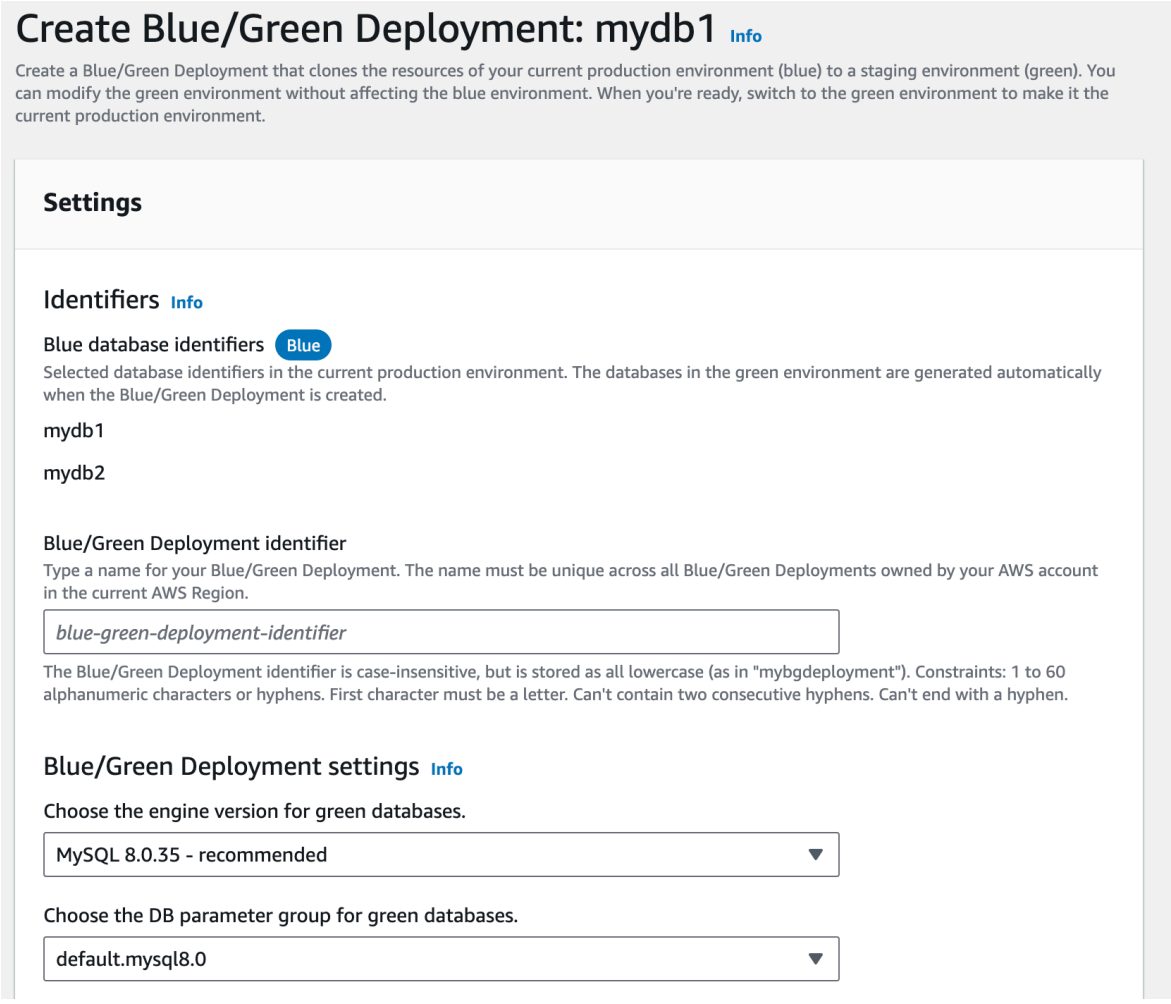
ブルー/グリーンデプロイは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して作成できます。

コンソール

ブルー/グリーンデプロイを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、グリーン環境にコピーする DB インスタンスを選択します。
3. [アクション] で [ブルー/グリーンデプロイを作成] を選択します。

   [ブルー/グリーンデプロイを作成] ページが表示されます。



4. ブルーデータベース識別子を確認します。ブルー環境で予期される DB インスタンスに一致することを確認します。一致しない場合は、[Cancel] (キャンセル) を選択します。
5. ブルー/グリーンデプロイ名として、ブルー/グリーンデプロイの名前を入力します。

6. 残りのセクションで、グリーン環境の設定を指定します。各設定の詳細については、「the section called “使用できる設定”」を参照してください。

   グリーン環境のデータベースには、デプロイ後に他の変更を加えることができます。

7. [作成] を選択します。

AWS CLI

AWS CLI を使用してブルー/グリーンデプロイを作成するには、create-blue-green-deployment コマンドを使用します。利用できるすべてのオプションについては、「the section called “使用できる設定”」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name my-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:db:mydb1 \
    --target-engine-version 8.0.31 \
    --target-db-parameter-group-name mydbparametergroup
```

Windows の場合:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name my-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:db:mydb1 ^
    --target-engine-version 8.0.31 ^
    --target-db-parameter-group-name mydbparametergroup
```

RDS API

Amazon RDS API を使用してブルー/グリーンデプロイを作成するには、`CreateBlueGreenDeployment` 操作を使用します。各オプションの詳細については、「」を参照してください。the section called “使用できる設定”

ブルー/グリーンデプロイを作成するための設定

次の表は、ブルー/グリーンデプロイを作成するときに選択できる設定を示しています。AWS CLI オプションの詳細については、「create-blue-green-deployment」を参照してください。RDS API パラメータの詳細については、「CreateBlueGreenDeployment」を参照してください。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

ストレージ割り当て	グリーン DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。割り当てストレージを増減することを選択できます。
	
	ブルー DB インスタンスがマグネティック (standard) ストレージを使用している場合は、グリーン環境で割り当てストレージを変更するために、グリーン DB インスタンスを汎用またはプロビジョンド IOPS ストレージタイプに変更する必要があります。
	
	詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。
		CLI オプション:
		
		`--target-allocated-storage`
		
		API パラメータ:
		
		`TargetAllocatedStorage`

ブルー/グリーンデプロイ識別子	ブルー/グリーンデプロイの名前。	CLI オプション:
		
		`--blue-green-deployment-name`
		
		API パラメータ:
		
		`BlueGreenDeploymentName`

ブルーデータベース識別子	グリーン環境にコピーするインスタンスの識別子。CLI または API を使用するときには、インスタンスの Amazon リソースネーム (ARN) を指定します。	CLI オプション:
		
		`--source`
		
		API パラメータ:
		
		`Source`

グリーンデータベースの DB パ	グリーン環境内のデータベースに関連付けるパラメータグループ。	CLI オプション:

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

ラメータグループ		`--target-db-parameter-group-name`
		
		`--target-db-cluster-parameter-group-name`
		
		API パラメータ:
		
		`TargetDBParameterGroupName`
		
		`TargetDBClusterParameterGroupName`

グリーンデータベースの最適化書き込みを有効にする	グリーンプライマリ DB インスタンスで RDS 最適化書き込みを有効にします。詳細については、「the section called “RDS Optimized Writes の有効化”」を参照してください。 Optimized Writes をサポートしない DB インスタンスクラスからサポートする DB インスタンスクラスに変更する場合は、ストレージ設定のアップグレードも実行する必要があります。詳細については、「the section called “ストレージ設定のアップグレード”」を参照してください。	CLI と API では、RDS 最適化書き込みをサポートするターゲット DB インスタンスクラスを指定すると、グリーンプライマリ DB インスタンスで自動的に有効になります。

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

グリーンデータベースのエンジンバージョン	グリーン環境のデータベースを指定された DB エンジンバージョンにアップグレードします。
	
	指定されなかった場合、グリーン環境の各データベースは、ブルー環境の対応する DB インスタンスと同じエンジンバージョンで作成されます。
	
	論理レプリケーションを使用する RDS for PostgreSQL DB インスタンスを選択する場合は、論理レプリケーションの制限を確認して承認してください。詳細については、「the section called “ブルー/グリーンデプロイの論理レプリケーション固有の制限事項”」を参照してください。
		CLI オプション:
		
		`--target-engine-version`
		
		RDS API パラメータ:
		
		`TargetEngineVersion`

グリーン DB インスタンスクラス	グリーン環境の各 DB インスタンスのコンピューティング容量とメモリ容量 (例: `db.m5d.xlarge`)。
	
	このオプションは、グリーンデータベースの RDS 最適化書き込みを有効にした場合にのみ表示されます。
		CLI オプション:
		
		`--target-db-instance-class`
		
		RDS API パラメータ:
		
		`TargetDBInstanceClass`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

プロビジョンド IOPS	初めにグリーンデータベースに割り当てられるプロビジョンド IOPS (1 秒あたりの入出力オペレーション) の量です。
	
	この値は、グリーンプライマリ DB インスタンスにのみ適用され、グリーンレプリカには適用されません。
		CLI オプション:
		
		`--target-iops`
		
		RDS API パラメータ:
		
		`TargetIops`

ストレージ設定のアップグレード	ストレージファイルシステム設定をアップグレードするかどうかを選択します。この設定を有効にすると、RDS はグリーンデータベースを古いストレージファイルシステムから優先設定に移行します。
	
	このオプションを使用できるのは、ブルーデータベースが最新のストレージ設定になっていない場合、または同じリクエスト内で RDS Optimized Writes を有効にする場合にのみです。ストレージ設定は、ブルー/グリーンデプロイを最初に作成するときにのみアップグレードできます。
	
	詳細については、「the section called “ストレージファイルシステムのアップグレード”」を参照してください。
		CLI オプション:
		
		`--upgrade-target-storage-config`
		
		RDS API パラメータ:
		
		`UpgradeTargetStorageConfig`

コンソール設定	設定の説明	CLI オプションと RDS API パラメータ

ストレージスループット	グリーンデータベースのストレージスループット値。この設定は、ストレージタイプに汎用 SSD (gp3) を選択した場合にのみ表示されます。
	
	この値は、グリーンプライマリ DB インスタンスにのみ適用され、グリーンレプリカには適用されません。
	
	詳細については、「gp3 ストレージ (推奨)」を参照してください。
		CLI オプション:
		
		`--target-storage-throughput`
		
		RDS API パラメータ:
		
		`TargetStorageThroughput`

ストレージタイプ	グリーンデータベースのストレージタイプ。以下のストレージタイプがサポートされています。
	
	• 汎用 SSD gp2
	• 汎用 SSD (gp3)
	• プロビジョンド IOPS (io1)
	• プロビジョンド IOPS SSD (io2)
	
	この値は、グリーンプライマリ DB インスタンスにのみ適用され、グリーンレプリカには適用されません。
	
	詳細については、「Amazon RDS ストレージタイプ」を参照してください。
		CLI オプション:
		
		`--target-storage-type`
		
		RDS API パラメータ:
		
		`TargetStorageType`

Amazon RDS でのブルー/グリーンデプロイの表示

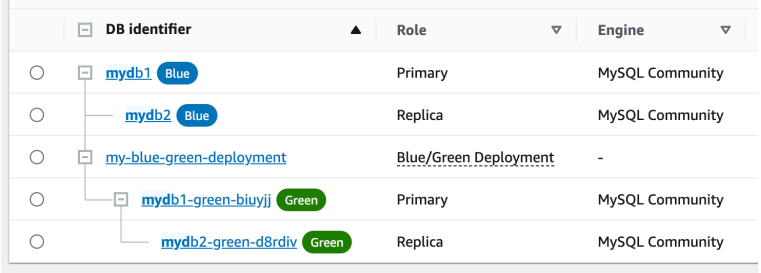
ブルー/グリーンデプロイの詳細は、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して表示できます。

ブルー/グリーンデプロイについての情報を表示して、イベントをサブスクライブすることもできます。詳細については、「ブルー/グリーンデプロイイベント」を参照してください。

コンソール

ブルー/グリーンデプロイの詳細を表示するには

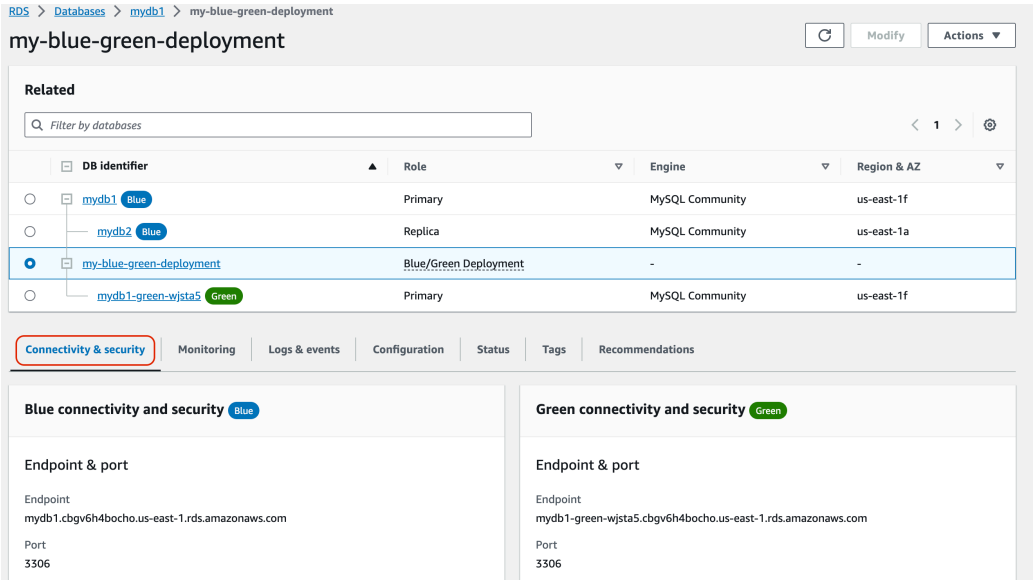
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [Databases] (データベース) を選択し、一覧からブルー/グリーンデプロイを見つけます。



   ブルー/グリーンデプロイの [Role] (ロール) 値は、[Blue/Green Deployment] (ブルー/グリーンデプロイ) です。
3. 表示するブルー/グリーンデプロイの名前を選択すると、詳細が表示されます。

   各タブには、ブルーデプロイ用のセクションとグリーンデプロイ用のセクションがあります。例えば、[設定] タブでは、グリーン環境で DB エンジンのバージョンをアップグレードしている場合、ブルー環境とグリーン環境で DB エンジンのバージョンが異なる場合があります。

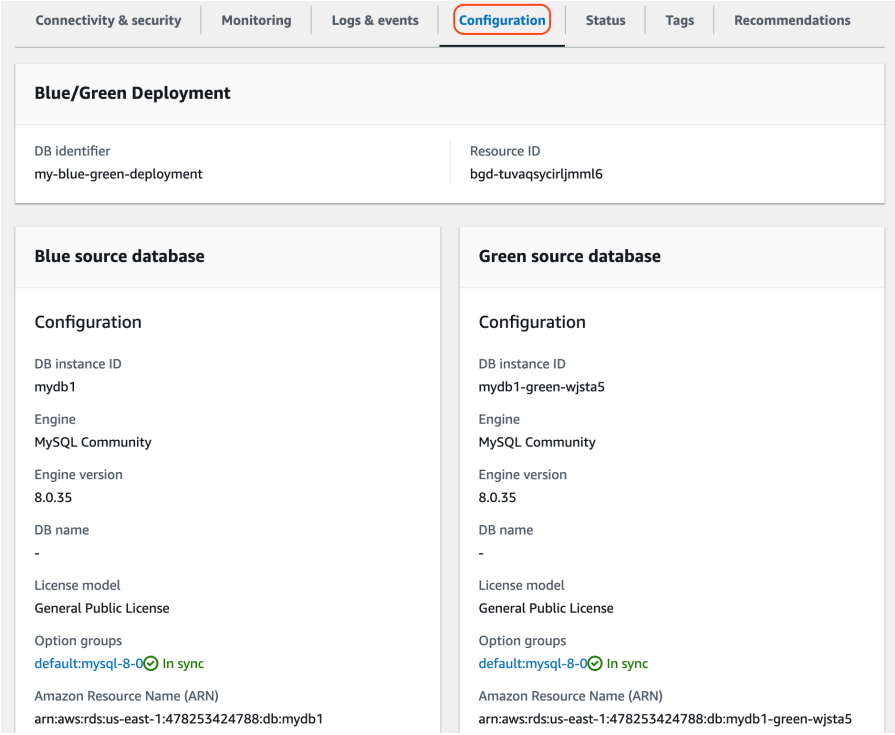
   次の画像は、[接続とセキュリティ] タブの例を示しています。



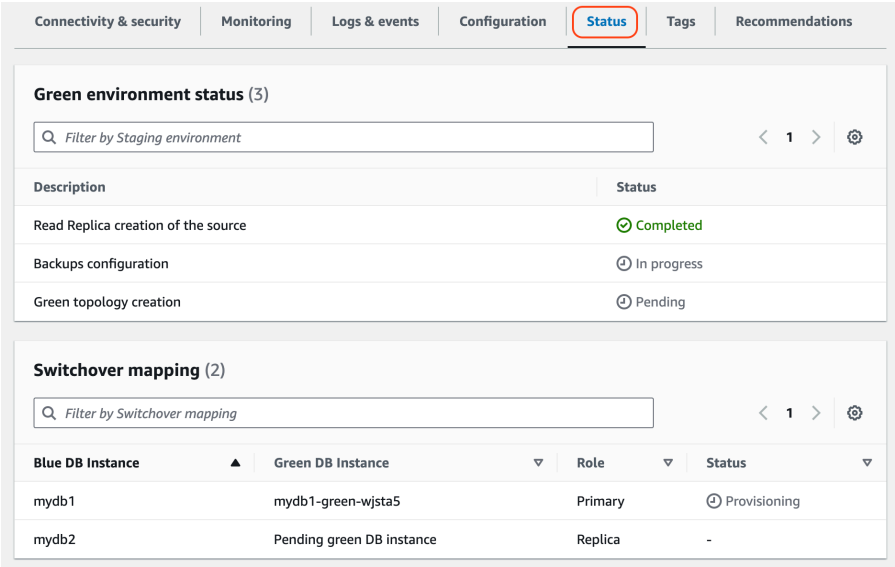
[接続とセキュリティ] タブには、[レプリケーション] というセクションもあります。このセクションには、レプリケーションの現在の状態と、ブルー環境とグリーン環境間のレプリカラグが表示されます。レプリケーションの状態が `Replicating` の場合、ブルー/グリーンデプロイは正常にレプリケートされています。

論理レプリケーションを使用する RDS for PostgreSQL ブルー/グリーンデプロイでは、ブルー環境でサポートされていない DDL または大きなオブジェクトを変更すると、レプリケーションの状態が `Replication degraded` に変わることがあります。詳細については、「the section called “ブルー/グリーンデプロイの論理レプリケーション固有の制限事項”」を参照してください。

次の画像は、[設定] タブの例を示しています。



次の画像は、[ステータス] タブの例を示しています。



AWS CLI

AWS CLI を使用してブルー/グリーンデプロイの詳細を表示するには、describe-blue-green-deployments コマンドを使用します。

Exampleブルー/グリーンデプロイの詳細を名前で絞り込んで表示する

describe-blue-green-deployments コマンドを使用すると、--blue-green-deployment-name でフィルタリングできます。

次の例は、*my-blue-green-deployment* という名前のブルー/グリーンデプロイの詳細を示しています。

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example識別子を指定して、ブルー/グリーンデプロイの詳細を表示する

describe-blue-green-deployments コマンドを使用すると、--blue-green-deployment-identifier オプションを指定できます。

次の例は、識別子 *bgd-1234567890abcdef* を持つブルー/グリーンデプロイの詳細を示しています。

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

RDS API

Amazon RDS API を使用してブルー/グリーンデプロイの詳細を表示するには、DescribeBlueGreenDeployments オペレーションを使用して BlueGreenDeploymentIdentifier を指定します。

Amazon RDS でのブルー/グリーンデプロイの切り替え

スイッチオーバーは、グリーン環境を新しい本番稼働環境に移行します。グリーン DB インスタンス内にリードレプリカがある場合、これらも移行されます。切り替え前は、本稼働環境のトラフィックはブルー環境の DB インスタンスとリードレプリカにルーティングされます。切り替え後は、本稼働環境のトラフィックはグリーン環境の DB インスタンスとリードレプリカにルーティングされます。

ブルー/グリーンデプロイの切り替えは、ブルー/グリーンデプロイ内のグリーン DB インスタンス の昇格とは異なります。[アクション] メニューで [昇格] を選択してグリーン DB インスタンス を手動で昇格させると、ブルー環境とグリーン環境間のレプリケーションが中断され、ブルー/グリーンデプロイは [無効な設定] の状態になります。

トピック

- 切り替えタイムアウト
- 切り替えガードレール
- 切り替えアクション
- Amazon RDS Proxy を使用する場合のスイッチオーバー
- 切り替えのベストプラクティス
- 切り替え前に CloudWatch メトリクスを確認する
- スイッチオーバー前のレプリカラグのモニタリング
- ブルー/グリーンデプロイの切り替え
- 切り替え後

切り替えタイムアウト

切り替えのタイムアウト期間は、30 秒から 3,600 秒 (1 時間) まで指定できます。切り替えに指定された期間より長くかかる場合、変更はすべてロールバックされ、どちらの環境にも変更は加えられません。デフォルトのタイムアウト期間は 300 秒 (5 分) です。

切り替えガードレール

切り替えを開始すると、Amazon RDS はいくつかの基本的なチェックを実行して、ブルー環境とグリーン環境が切り替えの準備が整っているかテストします。これらのチェックは切り替えガードレールと呼ばれます。これらの切り替えガードレールは、準備が整っていない環境の切り替えを防ぎます。そのため、予想以上に長いダウンタイムが回避され、切り替えが開始された場合に発生する可能性のあるブルー環境とグリーン環境間のデータ損失を防ぐことができます。

Amazon RDS は、グリーン環境で以下のガードレールチェックを実行します。

- レプリケーションの状態 – グリーンプライマリ DB インスタンス のレプリケーションステータスが正常かどうかをチェックします。グリーンプライマリ DB インスタンス は、ブループライマリ DB インスタンス のレプリカです。
- レプリケーションラグ – グリーンプライマリ DB インスタンス のレプリカラグがスイッチオーバーの許容範囲内にあるかどうかをチェックします。許容限度は、指定されたタイムアウト期間に基づきます。レプリカラグは、グリーンプライマリ DB インスタンス がブループライマリ DB インスタンス よりどれだけ遅れているかを示します。詳細については、「the section called “スイッチオーバー前のレプリカラグのモニタリング”」を参照してください。

- アクティブな書き込み – グリーンプライマリ DB インスタンスにアクティブな書き込みがないことを確認します。

Amazon RDS は、ブルー環境で以下のガードレールチェックを実行します。

- 外部レプリケーション — RDS for PostgreSQL では、ブルー環境がセルフマネージド論理ソース (パブリッシャー) でもレプリカ (サブスクライバー) でもないことを確認します。その場合は、ブルー環境のすべてのデータベースでセルフマネージドレプリケーションスロットとサブスクリプションを削除し、スイッチオーバーを続行してからそれらを再作成してレプリケーションを再開することをお勧めします。RDS for MySQL および RDS for MariaDB の場合は、ブルーデータベースが外部のバイナリログレプリカではないことを確認してください。その場合、アクティブにレプリケートされていないことを確認してください。
- 実行時間の長いアクティブな書き込み — レプリカラグが増える可能性があるため、ブループライマリ DB インスタンス に実行時間の長いアクティブな書き込みがないことを確認します。
- 実行時間が長い DDL ステートメント – レプリカラグを増加させる可能性があるため、ブループライマリ DB インスタンス に実行時間が長い DDL ステートメントがないことを確認します。
- サポートされていない PostgreSQL の変更 – 論理レプリケーションを使用する RDS for PostgreSQL ブルー/グリーンデプロイでは、ブルー環境で DDL の変更や大きなオブジェクトの追加や変更が行われていないことを確認します。詳細については、「the section called “ブルー/グリーンデプロイの論理レプリケーション固有の制限事項”」を参照してください。

  Amazon RDS がサポートされていない PostgreSQL の変更を検出すると、レプリケーションの状態が `Replication degraded` に変更され、ブルー/グリーンデプロイではスイッチオーバーができないことが通知されます。スイッチオーバーを続行するには、ブルー/グリーンデプロイとすべてのグリーンデータベースを削除して再作成することをお勧めします。そのためには、[アクション]、[グリーンデータベースで削除] を選択します。

切り替えアクション

ブルー/グリーンデプロイを切り替えると、RDS は次のアクションを実行します。

1. ガードレールチェックを実行して、ブルー環境とグリーン環境を切り替える準備ができているかどうかを確認します。
2. 両方の環境でプライマリ DB インスタンス での新しい書き込みオペレーションを停止します。
3. 両方の環境で DB インスタンスへの接続を切断し、新しい接続を許可しません。

4. グリーン環境がブルー環境と同期するように、レプリケーションがグリーン環境で追いつくのを待ちます。
5. 両方の環境の DB インスタンスの名前を変更します。

   RDS は、グリーン環境の DB インスタンスが、ブルー環境の対応する DB インスタンスに一致するように名前を変更します。例えば、ブルー環境の DB インスタンスの名前が `mydb` であるとします。また、グリーン環境の対応する DB インスタンスの名前が `mydb-green-abc123` であると仮定します。切り替え時、グリーン環境の DB インスタンスの名前は `mydb` に変更されます。

   RDS は、現在の名前に `-old`*n* を追加して、ブルー環境の DB インスタンスの名前を変更します。ここで、*n* は数字です。例えば、ブルー環境の DB インスタンスの名前が `mydb` であるとします。切り替え後、DB インスタンス名は `mydb-old1` になります。

   また、RDS はグリーン環境のエンドポイントの名前を、ブルー環境の対応するエンドポイントと一致するように変更するため、アプリケーションを変更する必要はありません。
6. 両方の環境でデータベースへの接続を許可します。
7. 新しい本稼働環境のプライマリ DB インスタンス への書き込みオペレーションを許可します。

   スイッチオーバー後、以前の本番プライマリ DB インスタンス は、`read_only` パラメータ (RDS for MySQL の場合) または `default_transaction_read_only` パラメータ (RDS for PostgreSQL の場合) を `0` に設定し、DB インスタンスが再起動されるまで読み取りオペレーションのみを許可します。

Amazon EventBridge を使用してスイッチオーバーのステータスをモニタリングできます。詳細については、「the section called “ブルー/グリーンデプロイイベント”」を参照してください。

スイッチオーバー中、ブルー環境のタグは、グリーン環境のリソースのすべてのタグを置き換えます。グリーン環境リソースに直接追加したタグは、このプロセス中に上書きされます。タグの詳細については、 Amazon RDS リソースのタグ付けを参照してください。

切り替えが開始され、終了する前に何らかの理由で停止した場合、変更はすべてロールバックされ、どちらの環境にも変更は加えられません。

Amazon RDS Proxy を使用する場合のスイッチオーバー

上記のスイッチオーバーアクションに加えて、ブルークラスターが Amazon RDS Proxy のターゲットである場合、スイッチオーバー中に次の動作が適用されます。

- スイッチオーバーガードレール

  - Amazon RDS は追加のガードレールチェックを実行して、プロキシがブルー環境とグリーン環境の両方に正常に到達でき、スイッチオーバーの準備が整っていることを検証します。
- スイッチオーバー中のアプリケーショントラフィックルーティング
  - スイッチオーバー中、ブルーデータベースは、グリーン環境が昇格する前に読み取り専用モードになります。Amazon RDS Proxy は、この移行期間中もブルーデータベースへの接続のルーティングを続行します。
  - この期間中の RDS for MySQL および RDS for MariaDB での書き込みオペレーションは、読み取り専用エラーを返す可能性があります。例: `1290 (HY000): The MySQL server is running with the --read-only option` このステートメントを実行することはできません。RDS for PostgreSQL では、読み取りクエリと書き込みクエリは `AdminShutdown: terminating connection due to administrator command` を返します。
  - スイッチオーバーが検出されると、プロキシは新しく昇格されたグリーン環境にトラフィックをルーティングします。
  - グリーン環境が新しいライターとして昇格されると、プロキシへの既存の接続は削除されます。アプリケーションは、昇格の完了後に接続を再確立する必要があります。
  - Amazon RDS Proxy の CloudWatch ログを確認して、この移行動作がいつ発生したかを確認できます。
- RDS Proxy API の動作
  - `describe-db-proxy-targets` などのプロキシ API は、トラフィックルーティングが以前に発生した場合でも、スイッチオーバーが完全に完了した後にのみ、更新されたターゲットを反映します。

切り替えのベストプラクティス

スイッチオーバーの前に、次のタスクを実行してベストプラクティスに従うことを強くお勧めします。

- グリーン環境でリソースを徹底的にテストします。適切かつ効率的に機能することを確認してください。
- 関連する Amazon CloudWatch メトリクスをモニタリングします。詳細については、「the section called “切り替え前に CloudWatch メトリクスを確認する”」を参照してください。
- 切り替えに最適なタイミングを特定します。

  切り替え中は、両方の環境でデータベースからの書き込みが遮断されます。本稼働環境でトラフィックが最も少ない時間を特定します。アクティブな DDL など、トランザクションの実行時間

が長い場合、切り替え時間が長くなり、本稼働環境のワークロードのダウンタイムが長くなる可能性があります。

DB インスタンス に多数の接続がある場合は、ブルー/グリーンデプロイを切り替える前に、アプリケーションに必要な最小限の接続数に手動で減らすことを検討してください。これを実現する 1 つの方法は、ブルー/グリーンデプロイのステータスを監視し、ステータスが SWITCHOVER_IN_PROGRESS に変わったことを検出すると接続のクリーンアップを開始するスクリプトを作成することです。

- 両方の環境の DB インスタンスが Available 状態にあることを確認します。
- グリーン環境のプライマリ DB インスタンス が正常でレプリケートしていることを確認します。
- ネットワークとクライアントの設定で、DNS キャッシュの存続可能時間 (TTL) が 5 秒を超えないようにしてください。これは RDS DNS ゾーンのデフォルトです。そうしないと、アプリケーションはスイッチオーバー後に書き込みトラフィックをブルー環境に送信し続けます。
- 切り替える前に、データの読み込みが完了していることを確認してください。詳細については、「the section called “遅延読み込みとストレージの初期化”」を参照してください。
- 論理レプリケーションを使用する RDS for PostgreSQL ブルー/グリーンデプロイの場合は、以下を実行します。
  - スイッチオーバーの前に論理レプリケーションの制約事項を確認し、必要なアクションをすべて実行します。詳細については、「the section called “ブルー/グリーンデプロイの論理レプリケーション固有の制限事項”」を参照してください。
  - ANALYZE 操作を実行して pg_statistics テーブルを更新します。これにより、スイッチオーバー後のパフォーマンス上の問題のリスクが軽減されます。
  - ブルー/グリーンデプロイのスイッチオーバーを開始する前に、アプリケーションがセッションレベルで default_transaction_read_only パラメータを上書きしないことを確認します。スイッチオーバー中、昇格が完了するまで書き込みが行われないように、このパラメータはグリーン環境ライターで on に設定されます。アプリケーションまたはトランザクションがこの設定を off に上書きする場合、アプリケーションはスイッチオーバープロセス中にグリーン環境にデータを書き込むことがあります。スイッチオーバーをロールバックする必要がある場合、これらの書き込みはブルー環境で使用できないため、データの不整合を手動で解決する必要があります。スイッチオーバーに進む前に、アプリケーションクエリを監査して default_transaction_read_only 設定を優先することを強くお勧めします。

> **Note**
>
> 切り替え中は、切り替えに含まれる DB インスタンスを変更することはできません。

切り替え前に CloudWatch メトリクスを確認する

ブルー/グリーンデプロイを切り替える前に、Amazon CloudWatch で次ののを確認することをお勧めします。

- `DatabaseConnections` — このメトリクスを使用して、ブルー/グリーンデプロイのアクティビティレベルを推定し、スイッチオーバー前に、その値がデプロイにとって許容可能なレベルであることを確認します。Performance Insights がオンになっている場合、`DBLoad` は、より正確なメトリクスになります。

詳細については、「the section called “RDS の CloudWatch メトリクス”」を参照してください。

スイッチオーバー前のレプリカラグのモニタリング

ブルー/グリーンデプロイを切り替える前に、ダウンタイムを減らすために、レプリカラグがゼロに近いことを確認します。

RDS for MySQL と RDS for MariaDB

MySQL と MariaDB ブルー/グリーンデプロイでは、グリーン環境の `ReplicaLag` CloudWatch メトリクスをチェックして、現在のレプリカラグを特定します。詳細については、「the section called “リードレプリカ間の遅延の診断と解決”」を参照してください。

RDS for PostgreSQL

物理レプリケーションを使用する PostgreSQL ブルー/グリーンデプロイについては、「the section called “レプリケーションプロセスのモニタリングとチューニング”」で現在のレプリカラグを特定する手順を確認してください。

論理レプリケーションを使用する PostgreSQL ブルー/グリーンデプロイの場合は、ブルー環境の `OldestReplicationSlotLag` CloudWatch メトリクスをチェックして、現在のレプリカラグを特定します。詳細については、「the section called “Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス”」を参照してください。

さらに、ブルー環境で次の SQL クエリを実行できます。

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn` は、レプリカに送信されたログシーケンス番号 (LSN) の最大値を表します。`pg_current_wal_lsn` はデータベースの現在の位置を表します。`lsn_distance` が 0 のとき、レプリカが追いついたことを意味します。

ブルー/グリーンデプロイが物理レプリケーションを使用する状況と論理レプリケーションを使用する状況については、「the section called “PostgreSQL レプリケーション方法”」を参照してください。

ブルー/グリーンデプロイの切り替え

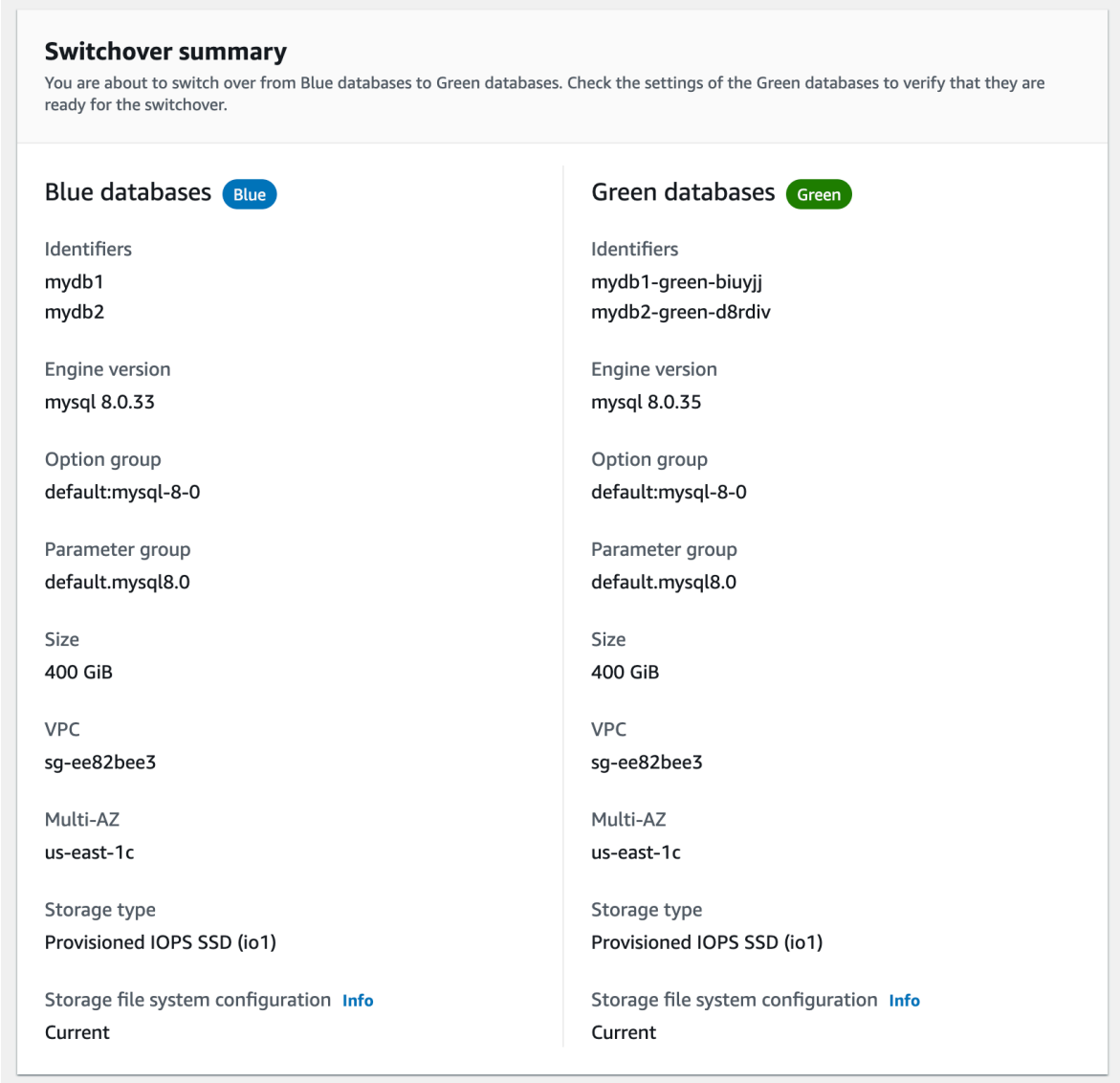
ブルー/グリーンデプロイは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して切り替えることができます。

コンソール

ブルー/グリーンデプロイを切り替えるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、切り替えるブルー/グリーンデプロイを選択します。
3. [Actions] (アクション) で、[Switch over] (切り替え) を選択します。

   [Switch over] (切り替え) ページが表示されます。



4. [Switch over] (切り替え) ページで、切り替えの概要を確認します。両方の環境のリソースが期待どおりであることを確認します。一致しない場合は、[Cancel] (キャンセル) を選択します。
5. [タイムアウトの設定] に、スイッチオーバーの制限時間を入力します。
6. インスタンスが RDS for PostgreSQL を実行している場合は、スイッチオーバーの前の推奨事項を確認し、承認してください。詳細については、「the section called “ブルー/グリーンデプロイの論理レプリケーション固有の制限事項”」を参照してください。
7. [Switch over] (切り替え) を選択します。

AWS CLI

AWS CLI を使用してブルー/グリーンデプロイを切り替えるには、switchover-blue-green-deployment コマンドを次のオプションを指定して使用します。

- `--blue-green-deployment-identifier` — 削除するブルー/グリーンデプロイのリソース ID を指定します。
- `--switchover-timeout` — 切り替えの制限時間を秒単位で指定します。デフォルトは 300 です。

Exampleブルー/グリーンデプロイを切り替える

Linux、macOS、Unix の場合:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Windows の場合:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

RDS API

Amazon RDS API を使用してブルー/グリーンデプロイを切り替えるには、SwitchoverBlueGreenDeployment オペレーションを以下のパラメータを指定して使用します。

- `BlueGreenDeploymentIdentifier` — 削除するブルー/グリーンデプロイのリソース ID を指定します。
- `SwitchoverTimeout` — 切り替えの制限時間を秒単位で指定します。デフォルトは 300 です。

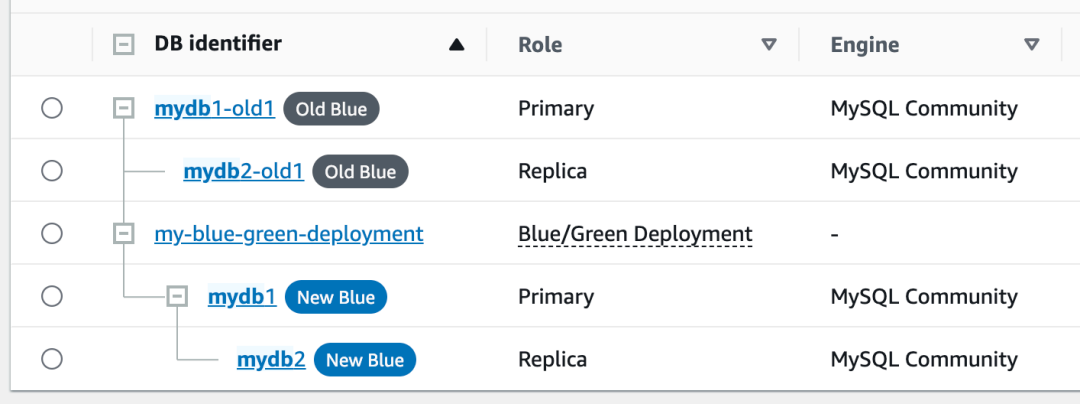
切り替え後

切り替え後、以前のブルー環境の DB インスタンスは保持されます。これらのリソースには標準費用が適用されます。ブルーとグリーンの環境間のレプリケーションとは停止します。

RDS は、現在のリソース名に `-old`*n* を付加することによって、ブルー環境の DB インスタンスの名前を変更します。ここで、*n* は数字です。古いブルー環境の DB インスタンスは、`read_only` パラメータ (RDS for MySQL の場合) または `default_transaction_read_only` パラメータ (RDS for

PostgreSQL の場合) を 0 に設定するまで読み取り専用になります。RDS は、グリーン環境の DB インスタンスを -new*n* と名付けます。

ブルー/グリーンデプロイリソースを削除すると、RDS は -old*n* および -new*n* リソースを保持します。



コンシューマーの親ノードの更新

RDS はフルマネージドリードレプリカを提供します。ただし、外部レプリカとも呼ばれるセルフマネージドレプリカを設定するオプションも用意されています。外部レプリカを使用すると、レプリケーションターゲットとしてサードパーティのリソースを使用できます。

RDS for MariaDB または RDS for MySQL ブルー/グリーンデプロイを切り替えた後、スイッチオーバー前にブルー DB インスタンス に外部レプリカまたはバイナリログコンシューマーがあった場合は、レプリケーションの継続性を維持するために、スイッチオーバー後に親ノードを更新する必要があります。

親ノードを更新するには

1. スイッチオーバー後、グリーン環境に以前存在していた DB インスタンスは、マスターログファイル名とマスターログの位置を含むイベントを発行します。イベントを見つけるには、RDS コンソールに移動し、左側のナビゲーションペインから [イベント] を選択します。
2. スイッチオーバー前の古いグリーン DB インスタンスの名前をソースとするイベントでフィルタリングします。
3. バイナリログ座標を含むイベントを見つけます。イベントメッセージは次のようになります: `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.000003 and position 40134574`
4. コンシューマーまたはレプリカが古いブルー環境からすべてのバイナリログを適用していることを確認します。次に、提供されたバイナリログ座標を使用して、コンシューマーでのレプリケー

ションを再開します。例えば、EC2 で MySQL レプリカを実行している場合は、以下のコマンドを使用できます。

MySQL 8.0.22 以前のメジャーバージョンとマイナーバージョン

```
CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
```

MySQL 8.0.23 以降のメジャーバージョンとマイナーバージョン

```
CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
 SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
```

コンシューマーが別の RDS for MariaDB または RDS for MariaDB の DB インスタンスである場合は、以下のストアドプロシージャを順番に実行します。

1. the section called “mysql.rds_stop_replication”
2. mysql.rds_reset_external_master (バージョン 8.0 以前) または mysql_rds_reset_external_source (バージョン 8.4 以降)
3. mysql.rds_set_external_master (バージョン 8.0 以前) または mysql_rds_set_external_source (バージョン 8.4 以降)
4. the section called “mysql.rds_start_replication”

Amazon RDS でのブルー/グリーンデプロイの削除

ブルー/グリーンデプロイは、切り替え前または切り替え後に削除できます。

切り替える前にブルー/グリーンデプロイを削除すると、Amazon RDS はグリーン環境の DB インスタンスをオプションで削除します。

- グリーン環境の DB インスタンスを削除する場合 (`--delete-target`)、そのインスタンスの削除保護がオフになっていることを確認してください。
- グリーン環境の DB インスタンスを削除しなかった場合 (`--no-delete-target`)、そのインスタンスは保持されますが、そのインスタンスはブルー/グリーンデプロイの一部ではなくなります。RDS for MySQL の場合、レプリケーションは環境間で続行されます。RDS for PostgreSQL の場合、グリーン環境はスタンドアロン環境に昇格されるため、レプリケーションは停止します。

グリーンデータベースを削除するオプションは、切り替え後はコンソールで使用できなくなります。AWS CLI を使用してブルー/グリーンデプロイを削除するときには、デプロイのステータスが SWITCHOVER_COMPLETED の場合、--delete-target パラメータを指定できません。

> **Important**
>
> ブルー/グリーンデプロイを削除しても、ブルー環境に影響はありません。

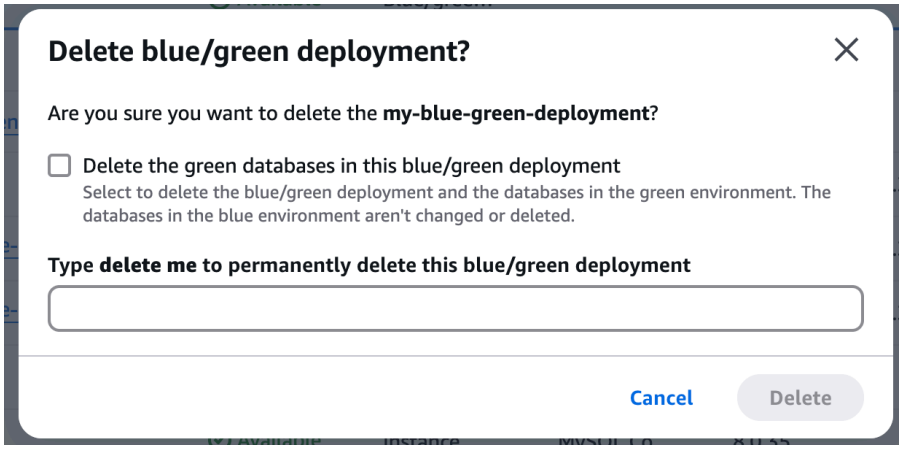
ブルー/グリーンデプロイは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して削除できます。

コンソール

ブルー/グリーンデプロイを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択し、削除するブルー/グリーンデプロイを選択します。
3. [Actions] (アクション) として、[Delete] (削除) を選択します。

   [ブルー/グリーンデプロイを削除しますか?] ウィンドウが表示されます。

   グリーンデータベースを削除するには、[このブルー/グリーンデプロイのグリーンデータベースを削除] を選択します。
4. ボックスに「**`delete me`**」と入力します。
5. [削除] を選択します。

AWS CLI

AWS CLI を使用してブルー/グリーンデプロイを削除するには、delete-blue-green-deployment コマンドを次のオプションを指定して使用します。

- `--blue-green-deployment-identifier` — 削除するブルー/グリーンデプロイのリソース ID。
- `--delete-target` — グリーン環境の DB インスタンスを削除するよう指定します。ブルー/グリーンデプロイのステータスが `SWITCHOVER_COMPLETED` の場合、このオプションは指定できません。
- `--no-delete-target` — グリーン環境の DB インスタンスを保持するよう指定します。

Exampleブルー/グリーンデプロイとグリーン環境の DB インスタンスを削除する

Linux、macOS、Unix の場合:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Windows の場合:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Exampleブルー/グリーンデプロイを削除し、グリーン環境の DB インスタンスを保持する

Linux、macOS、Unix の場合:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Windows の場合:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

RDS API

Amazon RDS API を使用してブルー/グリーンデプロイを削除するには、以下のパラメータを指定して DeleteBlueGreenDeployment オペレーションを使用します。

- BlueGreenDeploymentIdentifier — 削除するブルー/グリーンデプロイのリソース ID。
- DeleteTarget — TRUE によりグリーン環境の DB クラスターを削除するか、FALSE によりインスタンスを保持するかを指定します。ブルー/グリーンデプロイのステータスが SWITCHOVER_COMPLETED の場合、TRUE にはできません。

データのバックアップ、復元、エクスポート

このセクションでは、Amazon RDS DB インスタンスまたはマルチ AZ DB クラスターをバックアップする方法、復元する方法、エクスポートする方法について説明します。

Amazon RDS DB インスタンスまたはマルチ AZ DB クラスターのバックアップの詳細については、以下のトピックを参照してください。

- バックアップの概要
- 自動バックアップの管理
- 手動バックアップの管理

Amazon RDS DB インスタンスまたはマルチ AZ DB クラスターの復元の詳細については、「DB インスタンスへの復元」を参照してください。

DB スナップショットのコピー、共有、エクスポートについては、以下のトピックを参照してください。

- Amazon RDS の DB スナップショットのコピー
- Amazon RDS の DB スナップショットを共有する
- Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート

AWS Backup が管理する自動バックアップの詳細については、「Amazon RDS の自動バックアップの管理に AWS Backup を使用する」を参照してください。

バックアップの概要

Amazon RDS は、DB インスタンスのバックアップ期間中に、DB インスタンスまたはマルチ AZ DB クラスターの自動バックアップを作成して保存します。RDS は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、DB インスタンス全体をバックアップします。RDS は、指定したバックアップ保持期間に従って DB インスタンスの自動バックアップを保存します。必要に応じて、バックアップ保持期間内の任意の時点に DB インスタンスを回復できます。

自動バックアップは次のルールに従います。

- 自動バックアップを行うには、DB インスタンスが `available` 状態になっている必要があります。DB インスタンスが `storage_full` など、`available` 以外の状態にある間は、自動バックアップは行われません。
- DB スナップショットのコピーが同じデータベースの同じ AWS リージョン で実行している間は、自動バックアップは行われません。

DB スナップショットを作成することにより、手動で DB インスタンスをバックアップすることもできます。DB スナップショットの手動作成について詳しくは、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

スナップショットおよびバックアップ機能は、マルチボリューム設定をサポートします。すべてのバックアップオペレーションには、プライマリボリュームと追加のストレージボリュームの両方が含まれます。スナップショットは、データベースストレージ設定全体をキャプチャします。ポイントインタイムリカバリ (PITR) は、すべてのストレージボリュームで機能します。

DB インスタンスの最初のスナップショットには、データベース全体のデータが含まれます。同じ DB インスタンスの後続のスナップショットは増分です。つまり、直近のスナップショットの保存後に変更されたデータのみが含まれます。

自動および手動 DB スナップショットのコピー、および手動 DB スナップショットの共有を行うことができます。DB スナップショットのコピーに関する詳細は、「Amazon RDS の DB スナップショットのコピー」を参照してください。DB スナップショットの共有に関する詳細は、「Amazon RDS の DB スナップショットを共有する」を参照してください。

バックアップストレージ

各 AWS リージョン の Amazon RDS バックアップストレージは、そのリージョンの自動バックアップと手動 DB スナップショットで構成されています。バックアップストレージの合計容量は、そ

のリージョンのすべてのバックアップに対するストレージの合計と等しくなります。DB スナップショットを他のリージョンに移動すると、移動先リージョンのバックアップストレージは増加します。バックアップは Amazon S3 に保存されます。

バックアップストレージコストの詳細については、「Amazon RDS の料金」を参照してください。

DB インスタンスの削除時に自動バックアップを保持することを選択した場合、自動バックアップは全保持期間にわたって保存されます。DB インスタンスの削除時に、[Retain automated backups (自動バックアップの保持)] を選択しない場合、すべての自動バックアップは DB インスタンスと一緒に削除されます。ただし、DB スナップショット (最終バックアップと手動スナップショットの両方) は自動バックアップとは独立しており、この削除の影響を受けません。Amazon RDS によって DB インスタンスの削除前に最終的な DB スナップショットが作成されるように選択した場合、そのスナップショットを使用して DB インスタンスを復元できます。オプションで、以前に作成した手動スナップショットを使用できます。インスタンスが削除されても、手動スナップショットは削除されません。リージョンあたり最大 100 個の手動スナップショットを作成することができます。

自動バックアップの管理

このセクションでは、DB インスタンスとマルチ AZ DB クラスターの自動バックアップを管理する方法について説明します。

トピック

- バックアップウィンドウ
- バックアップの保存期間
- 自動バックアップの有効化
- 自動バックアップの保持
- 保持している自動バックアップの削除
- サポートされない MySQL ストレージエンジンを使用した自動バックアップ
- サポートされない MariaDB ストレージエンジンを使用した自動バックアップ
- 別の AWS リージョン への自動バックアップのレプリケーション

バックアップウィンドウ

自動バックアップは、優先されるバックアップウィンドウ中に毎日行われます。バックアップ期間に割り当てられた時間より長い時間がバックアップに必要な場合、期間が終了した後もバックアップが完了するまでバックアップが継続します。バックアップ期間は、DB インスタンスまたはマルチ AZ DB クラスターの週 1 回のメンテナンス期間と重複させることはできません。

自動バックアップウィンドウ中、バックアッププロセスのスタート時にストレージ I/O が一時中断することがあります (通常は数秒間)。マルチ AZ 配置のバックアップ中は、レイテンシーが数分間高くなることがあります。MariaDB、MySQL、Oracle、PostgreSQL の場合、バックアップはスタンバイから取られるため、マルチ AZ 配置のバックアップ中、プライマリの I/O アクティビティは中断しません。SQL Server の場合、バックアップはプライマリから取られるため、シングル AZ 配置とマルチ AZ 配置のバックアップ中に I/O アクティビティが短時間だけ中断します。Db2 では、バックアップがスタンバイから取得されている場合でも、バックアップ中に I/O アクティビティが短時間、中断されます。

バックアップが開始するときに、DB インスタンスまたはクラスターの負荷が大きい場合、自動バックアップがスキップされることがあります。バックアップがスキップされた場合でも、ポイントインタイムリカバリ (PITR) を実行し、次のバックアップウィンドウ中にバックアップを試行できます。PITR の詳細については「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

DB インスタンスまたはマルチ AZ DB クラスターの作成時に優先バックアップ期間を指定しなかった場合、Amazon RDS によってデフォルトの 30 分のバックアップ期間が割り当てられます。この期間は、各 AWS リージョン の 8 時間の時間ブロックからランダムに選択されます。次の表は、デフォルトのバックアップ期間が割り当てられる各 AWS リージョン の時間ブロックを示しています。

リージョン名	リージョン	時間ブロック
米国東部 (バージニア北部)	us-east-1	03:00 ~ 11:00 UTC
米国東部 (オハイオ)	us-east-2	03:00 ~ 11:00 UTC
米国西部 (北カリフォルニア)	us-west-1	06:00 ~ 14:00 UTC
米国西部 (オレゴン)	us-west-2	06:00 ~ 14:00 UTC
アフリカ (ケープタウン)	af-south-1	03:00 ~ 11:00 UTC
アジアパシフィック (香港)	ap-east-1	06:00 ~ 14:00 UTC
アジアパシフィック (ハイデラバード)	ap-south-2	06:30 ~ 14:30 UTC
アジアパシフィック (ジャカルタ)	ap-southeast-3	08:00 ~ 16:00 UTC
アジアパシフィック (マレーシア)	ap-southeast-5	09:00 ~ 17:00 UTC
アジアパシフィック (メルボルン)	ap-southeast-4	11:00 ~ 19:00 UTC
アジアパシフィック (ムンバイ)	ap-south-1	16:30 ~ 00:30 UTC

リージョン名	リージョン	時間ブロック
アジアパシフィック (ニュージーランド)	ap-southeast-6	13:00 ~ 21:00 UTC
アジアパシフィック (大阪)	ap-northeast-3	00:00 ~ 08:00 UTC
アジアパシフィック (ソウル)	ap-northeast-2	13:00 ~ 21:00 UTC
アジアパシフィック (シンガポール)	ap-southeast-1	14:00 ~ 22:00 UTC
アジアパシフィック (シドニー)	ap-southeast-2	12:00 ~ 20:00 UTC
アジアパシフィック (台北)	ap-east-2	9:00 ~ 17:00 UTC
アジアパシフィック (タイ)	ap-southeast-7	8:00 ~ 16:00 UTC
アジアパシフィック (東京)	ap-northeast-1	13:00 ~ 21:00 UTC
カナダ (中部)	ca-central-1	03:00 ~ 11:00 UTC
カナダ西部 (カルガリー)	ca-west-1	18:00 ~ 02:00 UTC
中国 (北京)	cn-north-1	06:00 ~ 14:00 UTC
中国 (寧夏)	cn-northwest-1	06:00 ~ 14:00 UTC
欧州 (フランクフルト)	eu-central-1	20:00 ~ 04:00 UTC
欧州 (アイルランド)	eu-west-1	22:00 ~ 06:00 UTC

リージョン名	リージョン	時間ブロック
欧州 (ロンドン)	eu-west-2	22:00 ~ 06:00 UTC
欧州 (ミラノ)	eu-south-1	02:00 ~ 10:00 UTC
欧州 (パリ)	eu-west-3	07:29 ~ 14:29 UTC
欧州 (スペイン)	eu-south-2	13:00 ~ 21:00 UTC
欧州 (ストックホルム)	eu-north-1	23:00 ~ 07:00 UTC
欧州 (チューリッヒ)	eu-central-2	02:00 ~ 10:00 UTC
イスラエル (テルアビブ)	il-central-1	03:00 ~ 11:00 UTC
メキシコ (中部)	mx-central-1	19:00 ~ 03:00 UTC
中東 (バーレーン)	me-south-1	06:00 ~ 14:00 UTC
中東 (アラブ首長国連邦)	me-central-1	05:00 ~ 13:00 UTC
南米 (サンパウロ)	sa-east-1	23:00 ~ 07:00 UTC
AWS GovCloud (米国東部)	us-gov-east-1	17:00 ~ 01:00 UTC
AWS GovCloud (米国西部)	us-gov-west-1	06:00 ~ 14:00 UTC

バックアップの保存期間

DB インスタンスまたはマルチ AZ DB クラスターを作成または復元するときに、バックアップ保持期間を設定できます。Amazon RDS API または AWS CLI を使用して DB インスタンスを作成し、バックアップ保持期間を設定しない場合、デフォルトのバックアップ保持期間は 1 日です。コンソールを使用して DB インスタンスを作成する場合、デフォルトのバックアップ保持期間は 7 日です。

DB インスタンスまたはクラスターを作成した後、バックアップ保持期間を変更できます。DB インスタンスのバックアップ保持期間は、0 ~ 35 日間で設定できます。バックアップ保持期間を 0 に設定すると、自動バックアップが無効になります。マルチ AZ DB クラスターのバックアップ保持期間は、1 ~ 35 日の間で設定できます。手動スナップショットの制限 (リージョンごとに 100) は、自動バックアップには適用されません。

復元オペレーション中に、DB インスタンスまたはマルチ AZ DB クラスターのバックアップ保持期間を指定できます。この値を明示的に設定しない場合、復元されたデータベースはソーススナップショットまたはインスタンスからバックアップ保持期間を継承します。この継承動作は復元オペレーションに固有であることに注意してください。新しいデータベースを作成すると、代わりにデフォルトの保持期間が適用されます。

> **Important**
>
> DB インスタンスのバックアップ保持期間を 0 から 0 以外の値、または 0 以外の値から 0 に変更すると、停止します。

RDS では、バックアップ保存期間の計算時に `stopped` 状態にあった時間は含まれません。DB インスタンスまたはクラスターの停止中は、自動バックアップは作成されません。DB インスタンスが停止している場合、バックアップ保持期間より長くバックアップを保持できます。

自動バックアップの有効化

DB インスタンスで自動バックアップが有効になっていない場合、いつでも有効にすることができます。バックアップ保持期間を 0 以外の正の値に設定することで自動バックアップを有効にします。自動バックアップが有効なとき、DB インスタンスがオフラインになり、ただちにバックアップが作成されます。

> **Note**
>
> AWS Backup でバックアップを管理している場合、自動バックアップを有効にすることはできません。詳細については、「Amazon RDS の自動バックアップの管理に AWS Backup を使用する」を参照してください。

コンソール

自動バックアップをすぐに有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスまたはマルチ AZ DB クラスターを選択します。
3. [Modify] (変更) を選択します。
4. [バックアップ保持期間] で、ゼロ以外の正の値 (3 日など) を選択します。
5. [続行] を選択します。
6. [すぐに適用] を選択します。
7. [DB インスタンスの変更] または [クラスターの変更] を選択して変更を保存し、自動バックアップを有効にします。

AWS CLI

自動バックアップを有効にするには、AWS CLI の modify-db-instance コマンドまたは modify-db-cluster コマンドを使用します。

以下のパラメータを含めます。

- `--db-instance-identifier` (またはマルチ AZ DB クラスターの場合は `--db-cluster-identifier`)
- `--backup-retention-period`
- `--apply-immediately` または `--no-apply-immediately`

次の例では、バックアップ保持期間を 3 日に設定して、自動バックアップを有効にします。変更はすぐに適用されます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
```

```
    --backup-retention-period 3 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

自動バックアップを有効にするには、以下の必須パラメータを指定して RDS API の ModifyDBInstance オペレーションまたは ModifyDBCluster オペレーションを使用します。

- DBInstanceIdentifier-または-DBClusterIdentifier
- BackupRetentionPeriod

自動バックアップの表示

自動バックアップを表示するには、ナビゲーションペインで「自動バックアップ」を選択します。自動バックアップに関連付けられている個々のスナップショットを表示するには、ナビゲーションペインで 「Snapshots」(スナップショット) を選択します。または、自動バックアップに関連付けられた個別のスナップショットを説明できます。そのページで、スナップショットのいずれかから DB インスタンスを直接復元できます。

自動スナップショットの名前はパターン rds:<database-name>-yyyy-mm-dd-hh-mm に従い、yyyy-mm-dd-hh-mm はスナップショットが作成された日時を表します。

AWS CLI を使用してDB インスタンスに既存の自動バックアップを説明するには、次のいずれかのコマンドを使用します。

```
aws rds describe-db-instance-automated-backups --db-instance-
identifier DBInstanceIdentifier
```

または

```
aws rds describe-db-instance-automated-backups --dbi-resource-id DbiResourceId
```

RDS API を使用して保持されている自動バックアップを説明するには、以下のパラメータのいずれかを指定して DescribeDBInstanceAutomatedBackups アクションを呼び出します。

- DBInstanceIdentifier
- DbiResourceId

自動バックアップの保持

> **Note**
>
> DB インスタンスの自動バックアップのみを保持でき、マルチ AZ DB クラスターは保持できません。

DB インスタンスを削除するときに、自動バックアップを保持することを選択できます。自動バックアップは、DB インスタンスの削除時に設定したバックアップ保持期間と同じ日数だけ保持されます。

保持されている自動バックアップには、DB インスタンスからのシステムスナップショットとトランザクションログが含まれます。割り当て済みストレージや DB インスタンスクラスなど、アクティブなインスタンスに復元するために必要な DB インスタンスプロパティも含まれます。

自動バックアップと手動スナップショットを保持すると、削除されるまで請求が発生します。詳細については、「保持コスト」を参照してください。

Db2、MariaDB、MySQL、PostgreSQL、Oracle、および Microsoft SQL Server の各エンジンを実行している RDS インスタンスの自動バックアップを保持できます。

AWS マネジメントコンソール、RDS API、および AWS CLI を使用すると、保持されている自動バックアップを復元または削除できます。

トピック

- 保持期間
- 保持されたバックアップの表示
- 復元
- 保持コスト
- 制約事項

保持期間

保持されている自動バックアップのシステムスナップショットやトランザクションログは、ソース DB インスタンスの期限切れと同じ方法で期限切れになります。このインスタンス用に作成された新しいスナップショットやログがないため、保持されている自動バックアップは最終的には完全に期限切れになります。実際には、ソースインスタンスを削除したときに設定されていた保持期間の設定に基づいて、最後のシステムスナップショットが実行されている間は存続します。保持されている自動バックアップは、最後のシステムスナップショットの有効期限が切れると、システムによって削除されます。

保持されている自動バックアップは、DB インスタンスを削除するのと同じ方法で削除できます。コンソールまたは RDS API オペレーション `DeleteDBInstanceAutomatedBackup` を使用すると、保持されている自動バックアップを削除できます。

最終スナップショットは、保持されている自動バックアップから独立しています。自動バックアップを保持している場合でも、保持されている自動バックアップは最終的に期限切れになるため、最終スナップショットを作成しておくことを強くお勧めします。最終スナップショットに有効期限はありません。

保持されたバックアップの表示

保持されている自動バックアップを表示するには、ナビゲーションペインで [Automated backups] (自動バックアップ) を選択し、[Retained] (保持) を選択します。保持された自動バックアップに関連付けられている個々のスナップショットを表示するには、ナビゲーションペインで [Snapshots] (スナップショット) を選択します。または、保持されている自動バックアップに関連付けられた個別のスナップショットを記述できます。そのページで、スナップショットのいずれかから DB インスタンスを直接復元できます。

AWS CLI を使用して保持されている自動バックアップを説明するには、次のいずれかのコマンドを使用します。

```
aws rds describe-db-instance-automated-backups --dbi-resource-id DbiResourceId
```

RDS API を使用して保持されている自動バックアップを説明するには、以下のパラメータのいずれかを指定して[DescribeDBInstanceAutomatedBackups]アクション`DbiResourceId`を呼び出します。

復元

自動バックアップから DB インスタンスを復元する方法については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

保持コスト

保持されている自動バックアップのコストは、関連付けられているシステムスナップショットの合計ストレージ容量のコストです。トランザクションログまたはインスタンスメタデータには追加料金はかかりません。バックアップのその他の料金ルールはすべて復元可能なインスタンスに適用されます。

例えば、実行中のインスタンスの割り当て済みストレージの合計が 100 GB であるとします。また、保持されている自動バックアップに関連付けられた手動スナップショットが 50 GB、システムスナップショットが 75 GB であるとします。この場合、25 GB の追加バックアップストレージについてのみ請求されます。(50 GB + 75 GB) - 100 GB = 25 GB。

制約事項

保持されている自動バックアップには、次の制限が適用されます。

- 1 つの AWS リージョンで保持できる自動バックアップの最大数は 40 個です。これは、DB インスタンスのクォータには含まれません。40 の実行 DB インスタンスと追加の 40 の自動バックアップを同時に保持できます。
- 保持されている自動バックアップには、パラメータグループまたはオプショングループについての情報は含まれません。
- 削除したインスタンスは、削除時の保持期間内の時点に復元できます。
- 保持された自動バックアップは変更できません。これは、システムバックアップ、トランザクションログ、およびソースインスタンスを削除したときに存在した DB インスタンスプロパティで構成されているためです。

保持している自動バックアップの削除

保持された自動バックアップは、不要になったら削除できます。

コンソール

保持されている自動バックアップを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups (自動バックアップ)] を選択します。
3. Retainedタブで、削除する保持された自動バックアップを選択します。
4. [アクション] で、[削除] を選択します。
5. 確認ページで、「**`delete me`**」を入力し、[Delete (削除)] を選択します。

AWS CLI

次のオプションを指定して AWS CLI コマンド delete-db-instance-automated-backup を使用すると、保持されている自動バックアップを削除できます。

- `--dbi-resource-id` - ソース DB インスタンスのリソース識別子です。

  AWS CLI コマンド describe-db-instance-automated-backups を実行すると、保持された自動バックアップのソース DB インスタンスのリソース識別子を見つけることができます。

Example

次の例では、ソース DB インスタンスのリソース識別子 `db-123ABCEXAMPLE` を持つ保持された自動バックアップを削除します。

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id db-123ABCEXAMPLE
```

Windows の場合:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id db-123ABCEXAMPLE
```

RDS API

次のパラメータを指定して Amazon RDS API オペレーション DeleteDBInstanceAutomatedBackup を使用すると、保持されている自動バックアップを削除できます。

- `DbiResourceId` - ソース DB インスタンスのリソース識別子です。

  Amazon RDS API オペレーション DescribeDBInstanceAutomatedBackups を使用して、保持された自動バックアップのソース DB インスタンスのリソース識別子を見つけることができます。

自動バックアップの無効化

大量のデータをロードする場合など、特定の状況では、自動バックアップを一時的に無効にすることができます。

> **Important**
>
> 自動バックアップは、無効にするとポイントインタイムリカバリも無効になるため、無効にしないことを強くお勧めします。DB インスタンスまたはマルチ AZ DB クラスターの自動バックアップを無効にすると、データベースの既存の自動バックアップがすべて削除されます。自動バックアップを無効にしてから再度有効にすると、自動バックアップを再度有効にした時点からリストアをスタートできます。

コンソール

自動バックアップをすぐに無効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスまたはマルチ AZ DB クラスターを選択します。
3. [変更] を選択します。
4. [バックアップ保持期間] で [0 日] を選択します。
5. [Continue] を選択します。
6. [すぐに適用] を選択します。
7. [DB インスタンスの変更] または [クラスターの変更] を選択して変更を保存し、自動バックアップを無効にします。

AWS CLI

自動バックアップをすぐに無効にするには、modify-db-instance または modify-db-cluster コマンドを使用して、バックアップ保持期間を 0 に設定し、`--apply-immediately` を指定します。

Example

次の例では、マルチ AZ DB クラスターの自動バックアップをただちに無効にします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 0 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 0 ^
    --apply-immediately
```

変更が有効になるタイミングを知るには、バックアップ保持期間の値が 0 になり、mydbcluster ステータスが available になるまで、DB インスタンスに対して `describe-db-instances` (マルチ AZ DB クラスターの場合は `describe-db-clusters`) を呼び出します。

```
aws rds describe-db-clusters --db-cluster-identifier mydcluster
```

RDS API

自動バックアップをすぐに無効にするには、以下のパラメータを指定して、ModifyDBInstance または ModifyDBCluster オペレーションを呼び出します。

- `DBInstanceIdentifier = mydbinstance+` または `DBClusterIdentifier = mydbcluster-`
- `BackupRetentionPeriod = 0`

Example

```
https://rds.amazonaws.com/
```

```
?Action=ModifyDBInstance
&DBInstanceIdentifier=mydbinstance
&BackupRetentionPeriod=0
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2009-10-14T17%3A48%3A21.746Z
&AWSAccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

サポートされない MySQL ストレージエンジンを使用した自動バックアップ

MySQL DB エンジン、自動バックアップは InnoDB ストレージエンジンでのみサポートされています。これらの機能を他の MySQL ストレージエンジン (MyISAM など) で使用すると、バックアップからの復元中に動作の信頼性が失われる場合があります。具体的には、MyISAM などのストレージエンジンは信頼性の高いクラッシュリカバリをサポートしていないため、クラッシュ時にテーブルが破損する可能性があります。このため、InnoDB ストレージエンジンを使用することをお勧めします。

- 既存の MyISAM テーブルを InnoDB テーブルに変換する場合は、`ALTER TABLE` コマンド (`ALTER TABLE *table_name* ENGINE=innodb, ALGORITHM=COPY;` など) を使用できます。
- MyISAM の使用を選択した場合、`REPAIR` コマンドを使用して、クラッシュ後に破損したテーブルの修復を手動で試みることができます。詳細については、MySQL ドキュメントの「REPAIR TABLE 構文」を参照してください。ただし、MySQL ドキュメントで説明されているとおり、すべてのデータを復旧できない可能性が高くなります。
- 復元前に MyISAM テーブルのスナップショットを作成する場合は、次のステップに従います。

  1. MyISAM テーブルに対するすべてのアクティビティを停止します (つまり、すべてのセッションを閉じます)。

     `SHOW FULL PROCESSLIST` コマンドから返される各プロセスの mysql.rds_kill コマンドを呼び出すことによって、すべてのセッションを閉じます。

  2. 各 MyISAM テーブルをロックしてフラッシュします。例えば、次のコマンドは、`myisam_table1` および `myisam_table2` という名前の 2 つのテーブルをロックしてフラッシュします。

     ```
     mysql> FLUSH TABLES myisam_table, myisam_table2 WITH READ LOCK;
     ```

3. DB インスタンスまたはマルチ AZ DB クラスターのスナップショットを作成します。スナップショットが完了したら、ロックを解除し、MyISAM テーブルでのアクティビティを再開します。次のコマンドを使用して、テーブルのロックを解除できます。

```
mysql> UNLOCK TABLES;
```

これらのステップでは、MyISAM によってメモリに保存されたデータがディスクに強制的にフラッシュされるため、DB スナップショットから復元する際のクリーンスタートを確実にできます。DB スナップショットの作成の詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

サポートされない MariaDB ストレージエンジンを使用した自動バックアップ

MariaDB DB エンジン、自動バックアップは InnoDB ストレージエンジンでのみサポートされています。これらの機能を他の MariaDB ストレージエンジン (Aria など) で使用すると、バックアップからの復元中に動作の信頼性が失われる場合があります。Aria が MyISAM の代替となる耐クラッシュ性を備えたものであっても、テーブルはクラッシュ時に破損する可能性があります。このため、InnoDB ストレージエンジンを使用することをお勧めします。

- 既存の Aria テーブルを InnoDB テーブルに変換するには、`ALTER TABLE` コマンドを使用できます。例: `ALTER TABLE table_name ENGINE=innodb, ALGORITHM=COPY;`
- Aria の使用を選択した場合、`REPAIR TABLE` コマンドを使用して、クラッシュ後に破損したテーブルの修復を手動で試みることができます。詳細については、http://mariadb.com/kb/en/mariadb/repair-table/ を参照してください。
- 復元前に Aria テーブルのスナップショットを作成する場合は、次のステップに従います。
  1. Aria テーブルに対するすべてのアクティビティを停止します (つまり、すべてのセッションを閉じます)。
  2. 各 Aria テーブルをロックしてフラッシュします。
  3. DB インスタンスまたはマルチ AZ DB クラスターのスナップショットを作成します。スナップショットが完了したら、ロックを解除し、Aria テーブルでのアクティビティを再開します。これらのステップでは、Aria のメモリに保存されたデータがディスクに強制的にフラッシュされるため、DB スナップショットから復元する際に確実にクリーンスタートできます。

別の AWS リージョン への自動バックアップのレプリケーション

ディザスタリカバリ機能を向上させるため、任意の送信先 AWS リージョン にスナップショットとトランザクションログをレプリケーションするよう Amazon RDS データベースインスタンスを設定できます。DB インスタンスにバックアップレプリケーションを設定すると、RDS は DB インスタンスで準備ができるとすぐに、すべてのスナップショットとトランザクションログのクロスリージョンコピーをスタートします。

データ転送には、DB スナップショットコピー料金が適用されます。DB スナップショットがコピーされると、コピー先リージョンのストレージにスタンダード料金が適用されます。詳細については、RDS の料金を参照してください。

バックアップレプリケーションの使用例については、AWS オンラインテクニカルトークの「Managed Disaster Recovery with Amazon RDS for Oracle Cross-Region Automated Backups」を参照してください。

> **Note**
>
> 自動バックアップレプリケーションは、マルチ AZ DB クラスターではサポートされていません。ただし、マルチ AZ DB インスタンスのデプロイではサポートされています。自動バックアップの制限の詳細については、「Amazon RDS のクォータと制約」を参照してください。

Amazon RDS の自動バックアップの設定と管理の詳細については、以下のトピックを参照してください。

トピック

- Amazon RDS のクロスリージョン自動バックアップの有効化
- Amazon RDS のレプリケートされたバックアップに関する情報の検索
- Amazon RDS のレプリケートされたバックアップから指定された時刻への復元
- Amazon RDS の自動バックアップレプリケーションの停止
- Amazon RDS のレプリケートされたバックアップの削除
- 停止したクロスリージョン自動バックアップのトラブルシューティング

マルチ AZ 配置のサポート

クロスリージョンの自動バックアップレプリケーションは、以下のエンジンのマルチ AZ DB インスタンスデプロイでサポートされています。

- RDS for Db2
- RDS for MariaDB
- RDS for MySQL
- RDS for Oracle
- RDS for PostgreSQL
- RDS for SQL Server

クロスリージョンの自動バックアップレプリケーションは、マルチ AZ DB クラスターではサポートされていません。

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。クロスリージョン自動バックアップによるバージョンとリージョンの可用性の詳細については、「Amazon RDS でのクロスリージョン自動バックアップでサポートされているリージョンと DB エンジン」を参照してください。

送信元と送信先 AWS リージョン サポート

バックアップレプリケーションは、次の AWS リージョン 間でサポートされます。

送信元リージョン	利用可能な送信先リージョン
アフリカ (ケープタウン)	欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (ロンドン)
アジアパシフィック (香港)	アジアパシフィック (シンガポール)、アジアパシフィック (東京)
アジアパシフィック (台北)	アジアパシフィック (シンガポール)、アジアパシフィック (東京)
アジアパシフィック (ハイデラバード)	アジアパシフィック (ムンバイ)

送信元リージョン	利用可能な送信先リージョン
アジアパシフィック (ジャカルタ)	アジアパシフィック (マレーシア)、アジアパシフィック (シンガポール)、アジアパシフィック (タイ)
アジアパシフィック (メルボルン)	アジアパシフィック (シドニー)、アジアパシフィック (ニュージーランド)
アジアパシフィック (マレーシア)	アジアパシフィック (シンガポール)
アジアパシフィック (ニュージーランド)	アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (メルボルン)
アジアパシフィック (タイ)	アジアパシフィック (シンガポール)、アジアパシフィック (ジャカルタ)
アジアパシフィック (ムンバイ)	アジアパシフィック (ハイデラバード)、アジアパシフィック (シンガポール) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)
アジアパシフィック (大阪)	アジアパシフィック (東京)
アジアパシフィック (ソウル)	アジアパシフィック (シンガポール)、アジアパシフィック (東京) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)
アジアパシフィック (シンガポール)	アジアパシフィック (香港)、アジアパシフィック (台北)、アジアパシフィック (マレーシア)、アジアパシフィック (ニュージーランド)、アジアパシフィック (タイ)、アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シドニー)、アジアパシフィック (東京) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)

送信元リージョン	利用可能な送信先リージョン
アジアパシフィック (シドニー)	アジアパシフィック (メルボルン)、アジアパシフィック (ニュージーランド)、アジアパシフィック (シンガポール) 米国東部 (バージニア北部)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)
アジアパシフィック (東京)	アジアパシフィック (香港)、アジアパシフィック (台北)、アジアパシフィック (大阪)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)
カナダ (中部)	カナダ西部 (カルガリー) 欧州 (アイルランド) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)
カナダ西部 (カルガリー)	カナダ (中部)
中国 (北京)	中国 (寧夏)
中国 (寧夏)	中国 (北京)
欧州 (フランクフルト)	アフリカ (ケープタウン) 欧州 (アイルランド)、欧州 (ロンドン)、欧州 (パリ)、欧州 (ストックホルム)、欧州 (チューリッヒ) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)

送信元リージョン	利用可能な送信先リージョン
欧州 (アイルランド)	アフリカ (ケープタウン) カナダ (中部) 欧州 (フランクフルト)、欧州 (ロンドン)、欧州 (パリ)、欧州 (ストックホルム)、欧州 (チューリッヒ) メキシコ (中部) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)
欧州 (ロンドン)	アフリカ (ケープタウン) 欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (パリ)、欧州 (ストックホルム) 米国東部 (バージニア北部)
欧州 (ミラノ)	欧州 (フランクフルト)
欧州 (パリ)	欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (ロンドン)、欧州 (ストックホルム) 米国東部 (バージニア北部)
欧州 (スペイン)	欧州 (アイルランド)、欧州 (パリ)
欧州 (ストックホルム)	欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (ロンドン)、欧州 (パリ) 米国東部 (バージニア北部)
欧州 (チューリッヒ)	欧州 (フランクフルト)、欧州 (アイルランド)
イスラエル (テルアビブ)	欧州 (アイルランド)
中東 (アラブ首長国連邦)	欧州 (フランクフルト)

送信元リージョン	利用可能な送信先リージョン
メキシコ (中部)	欧州 (アイルランド)、米国西部 (北カリフォルニア)
南米 (サンパウロ)	米国東部 (バージニア北部)、米国東部 (オハイオ)
AWS GovCloud (米国東部)	AWS GovCloud (米国西部)
AWS GovCloud (米国西部)	AWS GovCloud (米国東部)
米国東部 (バージニア北部)	アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京) カナダ (中部) 欧州 (フランクフルト)、欧州 (アイルランド)、欧州 (ロンドン)、欧州 (パリ)、欧州 (ストックホルム) 南米 (サンパウロ) 米国東部 (オハイオ)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)
米国東部 (オハイオ)	アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (東京) カナダ (中部) 欧州 (フランクフルト)、欧州 (アイルランド) 南米 (サンパウロ) 米国東部 (バージニア北部)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)

送信元リージョン	利用可能な送信先リージョン
米国西部 (北カリフォルニア)	アジアパシフィック (シドニー) カナダ (中部) 欧州 (アイルランド) メキシコ (中部) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)
米国西部 (オレゴン)	アジアパシフィック (ムンバイ)、アジアパシフィック (ソウル)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京) カナダ (中部) 欧州 (フランクフルト)、欧州 (アイルランド) 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (北カリフォルニア)

また、`describe-source-regions` AWS CLI コマンドを使用して、相互にレプリケートできる AWS リージョン を調べることもできます。詳細については、「Amazon RDS のレプリケートされたバックアップに関する情報の検索」を参照してください。

制限事項

Amazon RDS のクロスリージョン自動バックアップの制限は次のとおりです。

- 自動バックアップレプリケーションは、マルチ AZ DB クラスターではサポートされていません。
- デフォルトでは、AWS アカウントごとに最大 20 個のクロスリージョン自動バックアップを設定できます。

Amazon RDS のクロスリージョン自動バックアップの有効化

Amazon RDS コンソールを使用して、新規または既存の DB インスタンスでバックアップレプリケーションを有効にできます。`start-db-instance-automated-backups-replication` AWS CLI コマンドまたは `StartDBInstanceAutomatedBackupsReplication` RDS API オペレーションを使用することもできます。各 AWS アカウント の各送信先 AWS リージョン に最大 20 のバックアップを複製できます。

> **Note**
>
> 自動バックアップをレプリケーションするには、必ず有効にしてください。詳しくは、「自動バックアップの有効化」を参照してください。

コンソール

新規または既存の DB インスタンスのバックアップレプリケーションを有効にできます。

- 新しい DB インスタンスの場合、インスタンスを起動する時に有効にします。詳細については、「DB インスタンスの設定」を参照してください。
- 既存の DB インスタンスの場合、以下の手順に従います。

既存の DB インスタンスのバックアップレプリケーションを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups (自動バックアップ)] を選択します。
3. [現在のリージョン] タブで、バックアップレプリケーションを有効にする DB インスタンスを選択します。
4. [アクション] で、[クロスリージョンレプリケーションの管理] を選択します。
5. [Backup replication] (バックアップレプリケーション) で、[Enable replication to another AWS リージョン] (別の へのレプリケーションを有効にする) を選択します。
6. [送信先リージョン] を選択します。
7. [レプリケーションされたバックアップの保持期間] を選択します。
8. ソース DB インスタンスで暗号化を有効にしている場合は、バックアップを暗号化するための [AWS KMS key] を選択するか、キー ARN を入力します。

9. [Save] を選択します。

ソースリージョンでは、レプリケーションされたバックアップは [自動バックアップ] ページの [現在のリージョン] タブに表示されます。送信先リージョンでは、レプリケーションされたバックアップは [自動バックアップ] ページの [レプリケーションされたバックアップ] タブに一覧表示されます。

AWS CLI

AWS CLI の `start-db-instance-automated-backups-replication` コマンドを使用して、バックアップレプリケーションを有効にします。

次の CLI の例では、米国西部 (オレゴン) リージョン の DB インスタンスから 米国東部 (バージニア北部) リージョン への自動バックアップをレプリケーションします。また、送信先リージョンの AWS KMS key を使用して、レプリケートされたバックアップも暗号化されます。

バックアップレプリケーションを有効にするには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds start-db-instance-automated-backups-replication \
--region us-east-1 \
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase" \
--kms-key-id "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE" \
--backup-retention-period 7
```

  Windows の場合:

```
aws rds start-db-instance-automated-backups-replication ^
--region us-east-1 ^
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase" ^
--kms-key-id "arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE" ^
--backup-retention-period 7
```

  `--source-region` オプションは、AWS GovCloud (米国東部) と AWS GovCloud (米国西部) リージョン間でバックアップを暗号化する場合に必要です。`--source-region` には、ソース DB インスタンスの AWS リージョン を指定します。

`--source-region` を指定しない場合、必ず `--pre-signed-url` の値を指定する必要があります。署名付きの URL は、ソースの AWS リージョン で呼び出される `start-db-instance-automated-backups-replication` コマンドに対する、署名バージョン 4 で署名されたリクエストを含む URL です。`pre-signed-url` オプションの詳細は、AWS CLI コマンドリファレンスの「start-db-instance-automated-backups」を参照してください。

RDS API

次のパラメータで `StartDBInstanceAutomatedBackupsReplication` RDS API オペレーションを使用して、バックアップレプリケーションを有効にします。

- `Region` (送信先リージョンから API オペレーションを呼び出さない場合)
- `SourceDBInstanceArn`
- `BackupRetentionPeriod`
- `KmsKeyId` (省略可能)
- `PreSignedUrl` (を使用する場合は必須)`KmsKeyId`

> **Note**
>
> バックアップを暗号化する場合は、署名付き URL も含める必要があります。署名付き URL の詳細については、Amazon Simple Storage Service API リファレンスの「リクエストの認証: クエリパラメータの使用 (AWS 署名バージョン 4)」および AWS 全般のリファレンスの「署名バージョン 4 署名プロセス」を参照してください。

Amazon RDS のレプリケートされたバックアップに関する情報の検索

レプリケーションされたバックアップに関する情報を検索するには、次の CLI コマンドを使用します。

- `describe-source-regions`
- `describe-db-instances`
- `describe-db-instance-automated-backups`

次の describe-source-regions の例では、送信先の 米国西部 (オレゴン) リージョンに自動バックアップをレプリケーションできるソースの AWS リージョン が示されています。

ソースリージョンに関する情報を表示するには

- 次のコマンドを実行します。

```
aws rds describe-source-regions --region us-west-2
```

出力は、バックアップの 米国西部 (オレゴン) へのレプリケーションは US East (N. Virginia) からはできるが 米国東部 (オハイオ) または 米国西部 (北カリフォルニア) からはできないことを示しています。

```
{
    "SourceRegions": [
        ...
        {
            "RegionName": "us-east-1",
            "Endpoint": "https://rds.us-east-1.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": true
        },
        {
            "RegionName": "us-east-2",
            "Endpoint": "https://rds.us-east-2.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": false
        },
            "RegionName": "us-west-1",
            "Endpoint": "https://rds.us-west-1.amazonaws.com",
            "Status": "available",
            "SupportsDBInstanceAutomatedBackupsReplication": false
        }
    ]
}
```

次の describe-db-instances の例では、DB インスタンスの自動バックアップを示しています。

DB インスタンスのレプリケーションされたバックアップを表示するには

- 以下のいずれかのコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
--db-instance-identifier mydatabase
```

Windows の場合:

```
aws rds describe-db-instances ^
--db-instance-identifier mydatabase
```

出力は、レプリケーションされたバックアップを含みます。

```
{
    "DBInstances": [
        {
            "StorageEncrypted": false,
            "Endpoint": {
                "HostedZoneId": "Z1PVIF0B656C1W",
                "Port": 1521,
            ...

            "BackupRetentionPeriod": 7,
            "DBInstanceAutomatedBackupsReplications":
 [{"DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-backup:ab-
L2IJCEXJP7XQ7HOJ4SIEXAMPLE"}]
        }
    ]
}
```

次の `describe-db-instance-automated-backups` の例では、DB インスタンスの自動バックアップを示しています。

DB インスタンスの自動バックアップを表示するには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds describe-db-instance-automated-backups \
```

```
--db-instance-identifier mydatabase
```

Windows の場合:

```
aws rds describe-db-instance-automated-backups ^
--db-instance-identifier mydatabase
```

出力は、バックアップが US East (N. Virginia) にレプリケーションされた 米国西部 (オレゴン) のソース DB インスタンスと自動バックアップを表示します。

```
{
    "DBInstanceAutomatedBackups": [
        {
            "DBInstanceArn": "arn:aws:rds:us-west-2:868710585169:db:mydatabase",
            "DbiResourceId": "db-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-west-2:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "BackupRetentionPeriod": 7,
            "DBInstanceAutomatedBackupsReplications":
 [{"DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-backup:ab-
L2IJCEXJP7XQ7HOJ4SIEXAMPLE"}]
            "Region": "us-west-2",
            "DBInstanceIdentifier": "mydatabase",
            "RestoreWindow": {
                "EarliestTime": "2020-10-26T01:09:07Z",
                "LatestTime": "2020-10-31T19:09:53Z",
            }
            ...
        }
    ]
}
```

次の describe-db-instance-automated-backups の例では、--db-instance-automated-backups-arn オプションを使用して、送信先リージョンでレプリケーションされたバックアップを表示します。

レプリケーションされたバックアップを表示するには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds describe-db-instance-automated-backups \
--db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
```

Windows の場合:

```
aws rds describe-db-instance-automated-backups ^
--db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
```

出力は、バックアップが US East (N. Virginia) でレプリケーションされた 米国西部 (オレゴン) のソース DB インスタンスを表示します。

```
{
    "DBInstanceAutomatedBackups": [
        {
            "DBInstanceArn": "arn:aws:rds:us-west-2:868710585169:db:mydatabase",
            "DbiResourceId": "db-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "DBInstanceAutomatedBackupsArn": "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE",
            "Region": "us-west-2",
            "DBInstanceIdentifier": "mydatabase",
            "RestoreWindow": {
                "EarliestTime": "2020-10-26T01:09:07Z",
                "LatestTime": "2020-10-31T19:01:23Z"
            },
            "AllocatedStorage": 50,
            "BackupRetentionPeriod": 7,
            "Status": "replicating",
            "Port": 1521,
            ...
        }
    ]
}
```

Amazon RDS のレプリケートされたバックアップから指定された時刻への復元

Amazon RDS コンソールを使用して、レプリケーションされたバックアップから、特定の時点にDB インスタンスを復元できます。`restore-db-instance-to-point-in-time` AWS CLI コマンド

または RestoreDBInstanceToPointInTime RDS API オペレーションを使用することもできます。

ポイントインタイムリカバリ (PITR) の一般情報については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

> **Note**
>
> 自動バックアップが AWS リージョン間でレプリケートされる場合、次の DB エンジンの制限に注意してください。
>
> - RDS for SQL Server では、オプショングループはコピーされません。
> - RDS for Oracle では、NATIVE_NETWORK_ENCRYPTION、OEM、OEM_AGENT および SSL のオプションはコピーされません。
>
> カスタムオプショングループを DB インスタンスに関連付けた場合、そのオプショングループを送信先リージョンで再作成できます。その後、送信先リージョンで DB インスタンスを復元し、カスタムオプショングループを関連付けます。詳しくは、「オプショングループを使用する」を参照してください。

コンソール

レプリケーションされたバックアップから指定された時刻に DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. リージョンの選択から、(バックアップがレプリケーションされる) 送信先リージョンを選択します。
3. ナビゲーションペインで、[Automated backups (自動バックアップ)] を選択します。
4. [レプリケーションされたバックアップ] タブで、復元する DB インスタンスを選択します。
5. [アクション] で、[特定時点への復元] を選択します。
6. [Latest restorable time] を選択してできるだけ最新の時点に復元するか、[Custom] を選択して時刻を選択します。

   [Custom (カスタム)] を選択した場合、インスタンスを復元する日時を入力します。

> **Note**
>
> 時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。

7. [DB インスタンス識別子] に、ターゲットが復元された DB インスタンスの名前を入力します。
8. (オプション) 必要に応じて、オートスケーリングを有効にするなど、その他のオプションを選択します。
9. [Restore to point in time (特定時点への復元)] を選択します。

AWS CLI

`restore-db-instance-to-point-in-time` AWS CLI コマンドを使用して、DB インスタンスを作成します。

レプリケーションされたバックアップから指定された時刻に DB インスタンスを復元するには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-automated-backups-arn "arn:aws:rds:us-
east-1:123456789012:auto-backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE" \
    --target-db-instance-identifier mytargetdbinstance \
    --restore-time 2020-10-14T23:45:00.000Z
```

  Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-automated-backups-arn "arn:aws:rds:us-
east-1:123456789012:auto-backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE" ^
    --target-db-instance-identifier mytargetdbinstance ^
    --restore-time 2020-10-14T23:45:00.000Z
```

RDS API

DB インスタンスを指定された時刻に復元するには、以下のパラメータを指定して `RestoreDBInstanceToPointInTime` Amazon RDS API オペレーションを呼び出します。

- `SourceDBInstanceAutomatedBackupsArn`
- `TargetDBInstanceIdentifier`
- `RestoreTime`

Amazon RDS の自動バックアップレプリケーションの停止

Amazon RDSコンソールを使用して、DB インスタンスのバックアップレプリケーションを停止できます。`stop-db-instance-automated-backups-replication` AWS CLI コマンドまたは `StopDBInstanceAutomatedBackupsReplication` RDS API オペレーションを使用することもできます。

レプリケーションされたバックアップは、作成時に設定したバックアップの保持期間に従って保持されます。

コンソール

ソースリージョンの [自動バックアップ] ページからバックアップレプリケーションを停止します。

AWS リージョン へのバックアップレプリケーションを停止するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. リージョンの選択からソースリージョンを選択します。
3. ナビゲーションペインで、[Automated backups (自動バックアップ)] を選択します。
4. [現在のリージョン] タブで、バックアップレプリケーションを停止する DB インスタンスを選択します。
5. [アクション] で、[クロスリージョンレプリケーションの管理] を選択します。
6. [Backup replication] (バックアップレプリケーション) で、[Enable replication to another AWS リージョン] (別の へのレプリケーションを有効にする) チェックボックスをオフにします。
7. [Save] を選択します。

レプリケーションされたバックアップは、送信先リージョンの [自動バックアップ] ページの [保持] タブに一覧表示されます。

AWS CLI

stop-db-instance-automated-backups-replication AWS CLIコマンドを使用して、バックアップレプリケーションを停止します。

次の CLI の例では、DB インスタンスの自動バックアップを 米国西部 (オレゴン) リージョンでレプリケーションするのを停止します。

バックアップレプリケーションを停止するには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds stop-db-instance-automated-backups-replication \
--region us-east-1 \
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase"
```

  Windows の場合:

```
aws rds stop-db-instance-automated-backups-replication ^
--region us-east-1 ^
--source-db-instance-arn "arn:aws:rds:us-west-2:123456789012:db:mydatabase"
```

RDS API

次のパラメータを指定して StopDBInstanceAutomatedBackupsReplication RDS API オペレーションを使用して、バックアップレプリケーションを停止します。

- `Region`
- `SourceDBInstanceArn`

Amazon RDS のレプリケートされたバックアップの削除

Amazon RDS コンソールを使用して、DB インスタンスのレプリケーションされたバックアップを削除できます。`delete-db-instance-automated-backups` AWS CLI コマンドまたは `DeleteDBInstanceAutomatedBackup` RDS API オペレーションを使用することもできます。

コンソール

[自動バックアップ] ページから、送信先リージョンでレプリケーションされたバックアップを削除します。

レプリケーションされたバックアップを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. リージョンの選択から送信先リージョンを選択します。
3. ナビゲーションペインで、[Automated backups (自動バックアップ)] を選択します。
4. [レプリケーションされたバックアップ] タブで、レプリケーションされたバックアップを削除する DB インスタンスを選択します。
5. [アクション] で、[削除] を選択します。
6. 確認ページで、「**`delete me`**」を入力し、[Delete (削除)] を選択します。

AWS CLI

`delete-db-instance-automated-backup` AWS CLI コマンドを使用して、レプリケーションされたバックアップを削除します。

`describe-db-instances` CLI コマンドを使用して、レプリケーションされたバックアップの Amazon リソースネーム (ARN) を検索できます。詳細については、「Amazon RDS のレプリケートされたバックアップに関する情報の検索」を参照してください。

レプリケーションされたバックアップを削除するには

- 以下のいずれかのコマンドを実行します。

  Linux、macOS、Unix の場合:

```
aws rds delete-db-instance-automated-backup \
```

```
--db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
```

Windows の場合:

```
aws rds delete-db-instance-automated-backup ^
--db-instance-automated-backups-arn "arn:aws:rds:us-east-1:123456789012:auto-
backup:ab-L2IJCEXJP7XQ7HOJ4SIEXAMPLE"
```

RDS API

`DeleteDBInstanceAutomatedBackup` パラメータを指定して `DBInstanceAutomatedBackupsArn` RDS API オペレーションを使用して、レプリケーションされたバックアップを削除します。

停止したクロスリージョン自動バックアップのトラブルシューティング

Amazon RDS は、特定の状況下でクロスリージョン自動バックアップレプリケーションを自動的に停止して、データを保護し、AWS 運用要件への準拠を維持します。

アカウントの停止

AWS アカウントが停止されている場合、Amazon RDS はそのアカウント内のすべての DB インスタンスのクロスリージョン自動バックアップレプリケーションを自動的に停止します。レプリケート先の AWS リージョンに存在するレプリケートされたバックアップは、指定した保持期間まで保持されます。

アカウントの停止が解決されたら、クロスリージョン自動バックアップレプリケーションを手動で再度有効にして、レプリケート先の AWS リージョンへのバックアップのレプリケーションを再開する必要があります。

オプトインリージョンの変更

クロスリージョン自動バックアップレプリケーションは、ソース AWS リージョン (プライマリ DB インスタンスがある場所) または宛先 AWS リージョン (バックアップがレプリケートされる場所) のいずれかをオプトアウトすると、自動的に停止します。オプトインリージョンの詳細については、「AWS リージョンの管理」を参照してください。

レプリケート先の AWS リージョンに存在するレプリケートされたバックアップは、指定した保持期間まで保持されます。レプリケーションを再開するには、必要な AWS リージョンにオプトバックし直し、手動でクロスリージョン自動バックアップレプリケーションを再度有効にします。

AWS KMS キーの問題

暗号化された DB インスタンスの場合、Amazon RDS はバックアップをレプリケートするためにソースと宛先 AWS リージョンの両方の AWS KMS キーにアクセスする必要があります。いずれかの AWS リージョン で AWS KMS キーを無効化または削除すると、クロスリージョン自動バックアップレプリケーションは自動的に停止します。レプリケート先の AWS リージョンに存在するレプリケートされたバックアップは、指定した保持期間まで保持されます。

クロスリージョン自動バックアップレプリケーションを再開するには

1. 無効化されている AWS KMS キーを再度有効にするか、元のキーが削除された場合は新しい AWS KMS キーを作成します。
2. 新しい AWS KMS キーを使用する場合、前のキーで暗号化された送信先 AWS リージョンの既存のレプリケートされたバックアップを削除します。
3. インスタンスのクロスリージョン自動バックアップレプリケーションを再度有効にする

AWS KMS キーの詳細については、「AWS Key Management Service のドキュメント」を参照してください。

> **Note**
>
> 前の AWS KMS キーが再度有効になっていない限り、送信先の AWS リージョンで以前にレプリケートされたバックアップから復元することはできません。バックアップは暗号化されたままで、有効な AWS KMS キーなしでアクセスできません。

バックアップレプリケーションステータスのモニタリング

Amazon RDS コンソール、AWS CLI、または RDS API を使用して、クロスリージョン自動バックアップのステータスをモニタリングできます。詳細については、「Amazon RDS のレプリケートされたバックアップに関する情報の検索」を参照してください。

手動バックアップの管理

このセクションでは、DB インスタンスと DB クラスターの手動バックアップを管理する方法について説明します。

Single-AZ DB インスタンスの DB スナップショットの作成について詳しくは、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

マルチ AZ DB クラスターの DB スナップショットの作成については、「Amazon RDS のマルチ AZ DB クラスターのスナップショットの作成」を参照してください。

DB スナップショットの削除については、「Amazon RDS の DB スナップショットの削除」を参照してください。

Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成

Amazon RDS は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、その DB インスタンス全体をバックアップします。Single-AZ DB インスタンスでこの DB スナップショットを作成すると、I/O が短時間中断します。この時間は、DB インスタンスのサイズやクラスによって異なり、数秒から数分になります。MariaDB、MySQL、Oracle、PostgreSQL の場合、バックアップはスタンバイから取得されるため、マルチ AZ 配置のバックアップ中プライマリで I/O アクティビティは中断しません。SQL Server の場合、マルチ AZ 配置のバックアップ中 I/O アクティビティが一時中断します。

DB スナップショットを作成したら、バックアップする DB インスタンスを識別した後、DB スナップショットに名前を付けて後で復元できるようにする必要があります。スナップショットの作成にかかる時間は、データベースのサイズによって異なります。スナップショットにはストレージボリューム全体が含まれているため、テンポラリファイルなどのファイルのサイズも、スナップショットを作成するための時間に影響します。

> **Note**
>
> DB スナップショットを行うには、DB インスタンスが `available` 状態になっている必要があります。
> PostgreSQL DB インスタンスの場合、ログに記録されていないテーブルのデータはスナップショットから復元されないことがあります。詳細については、「PostgreSQL を使用するためのベストプラクティス」を参照してください。

自動バックアップとは異なり、手動スナップショットはバックアップ保持期間の影響を受けません。スナップショットは期限切れになりません。

MariaDB、MySQL、および PostgreSQL データの非常に長期間のバックアップの場合、スナップショットデータを Amazon S3 にエクスポートすることをお勧めします。DB エンジンのメジャーバージョンがサポートされなくなった場合、スナップショットからそのバージョンに復元することはできません。詳細については、「Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB スナップショットを作成できます。

コンソール

DB スナップショットを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。

   手動スナップショットリストが表示されます。
3. [スナップショットの取得] を選択します。

   [Take DB snapshot] (DB スナップショットの取得) ウィンドウが表示されます。
   スナップショットを取得します。
4. [DB インスタンス] のリストで、スナップショットを取得する DB インスタンスを選択します。
5. [スナップショット名] を入力します。
6. [スナップショットの取得] を選択します。

[スナップショット] ページが表示され、新しい DB スナップショットのステータスが「Creating」として表示されます。ステータスが Available になると、その作成時間が表示されます。

AWS CLI

AWS CLI を使用して DB スナップショットを作成したら、バックアップする DB インスタンスを識別した後、DB スナップショットに名前を付けて後で復元できるようにする必要があります。そのためには、以下のパラメータを指定して AWS CLI の `create-db-snapshot` コマンドを使用します。

- `--db-instance-identifier`
- `--db-snapshot-identifier`

この例では、*`mydbinstance`* という DB インスタンスに *`mydbsnapshot`* という DB スナップショットを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-snapshot \
```

```
    --db-instance-identifier mydbinstance \
    --db-snapshot-identifier mydbsnapshot
```

Windows の場合:

```
aws rds create-db-snapshot ^
    --db-instance-identifier mydbinstance ^
    --db-snapshot-identifier mydbsnapshot
```

RDS API

Amazon RDS API を使用して DB スナップショットを作成したら、バックアップする DB インスタンスを識別した後、DB スナップショットに名前を付けて後で復元できるようにする必要があります。そのためには、以下のパラメータを指定して Amazon RDS API の `CreateDBSnapshot` コマンドを使用します。

- `DBInstanceIdentifier`
- `DBSnapshotIdentifier`

Amazon RDS のマルチ AZ DB クラスターのスナップショットの作成

マルチ AZ DB クラススナップショットを作成したら、バックアップするマルチ AZ DB クラスターを特定してから、DB クラススナップショットに名前を付けて後で復元できるようにする必要があります。マルチ AZ DB クラスターのスナップを共有することもできます。手順については、the section called “DB スナップショットを共有する” を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、マルチ AZ DB クラススナップショットを作成できます。

非常に長期間のバックアップの場合、スナップショットデータを Amazon S3 にエクスポートすることをお勧めします。DB エンジンのメジャーバージョンがサポートされなくなった場合、スナップショットからそのバージョンに復元することはできません。詳細については、「Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。

コンソール

DB クラススナップショットを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベースを選択します。
3. リストで、スナップショットを作成したいマルチ AZ DB クラスターを選択します。
4. [アクション] で、[スナップショットの取得] を選択します。

   [Take DB snapshot] (DB スナップショットの取得) ウィンドウが表示されます。
5. [スナップショット名] に、スナップショットの名前を入力します。
6. [スナップショットの取得] を選択します。

スナップショットページが表示され、新しい マルチ AZ DB スナップショットのステータスがCreatingとして表示されます。ステータスが Available になると、その作成時間が表示されます。

AWS CLI

次のオプションを指定してAWS CLIcreate-db-cluster-snapshotコマンドを使用すると、マルチ AZ DB クラススナップショットを作成できます。

- `--db-cluster-identifier`

- `--db-cluster-snapshot-identifier`

この例では、*mymultiazdbcluster* という DB クラスターの *mymultiazdbclustersnapshot* というマルチ AZ DB クラスタースナップショットを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mymultiazdbcluster \
    --db-cluster-snapshot-identifier mymultiazdbclustersnapshot
```

Windows の場合:

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mymultiazdbcluster ^
    --db-cluster snapshot-identifier mymultiazdbclustersnapshot
```

RDS API

次のパラメータを指定して Amazon RDS API CreateDBClusterSnapshotオペレーションを使用すると、マルチ AZ DB クラスタースナップショットを作成できます。

- `DBClusterIdentifier`
- `DBClusterSnapshotIdentifier`

マルチ AZ DB クラスターのスナップショットの削除

Amazon RDS によって管理されているマルチ AZ DB のスナップショットは、不要になったら削除できます。手順については、the section called “DB スナップショットの削除” を参照してください。

Amazon RDS の DB スナップショットの削除

Amazon RDS で管理されている DB のスナップショットは、不要になったら削除することができます。

> **Note**
>
> AWS Backup で管理されているバックアップを削除するには、AWS Backup コンソールを使用します。AWS Backup の詳細については、「AWS Backup デベロッパーガイド」を参照してください。

DB スナップショットの削除

手動、共有、またはパブリックの DB スナップショットを削除するには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用します。

共有またはパブリックのスナップショットを削除するには、そのスナップショットを所有する AWS アカウントにサインインする必要があります。

DB インスタンスを削除せずに削除する自動 DB スナップショットがある場合は、DB インスタンスのバックアップ保持期間を 0 に変更します。自動スナップショットは、変更適用時に削除されます。次回メンテナンス期間まで待機せずに、即時に変更を適用することもできます。変更が完了したら、バックアップ保持期間を 0 を超える値に設定することで、自動バックアップを再度有効にすることができます。DB インスタンスの変更については、Amazon RDS DB インスタンスを変更するを参照してください。

自動バックアップと手動スナップショットを保持すると、削除されるまで請求が発生します。詳細については、「保持コスト」を参照してください。

DB インスタンスを削除したら、その DB インスタンスの自動バックアップを削除することで、自動 DB スナップショットを削除することができます。自動バックアップの詳細については、「バックアップの概要」を参照してください。

コンソール

DB スナップショットを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[Snapshots] を選択します。

   手動スナップショットリストが表示されます。
3. 削除する DB スナップショットを選択します。
4. 「アクション」 で、「スナップショットの削除」 を選択します。
   スナップショットを削除する。
5. 確認ページで、「削除」 を選択します。
   スナップショットの削除を確認。

AWS CLI

DB スナップショットを削除するには、AWS CLI の delete-db-snapshot コマンドを使用します。

DB スナップショットを削除するには、以下のオプションを使用します。

- `--db-snapshot-identifier` - DB スナップショットの識別子。

Example

次のコードは、DB スナップショット `mydbsnapshot` を削除します。

Linux、macOS、Unix の場合:

```
aws rds delete-db-snapshot \
    --db-snapshot-identifier mydbsnapshot
```

Windows の場合:

```
aws rds delete-db-snapshot ^
    --db-snapshot-identifier mydbsnapshot
```

RDS API

DB スナップショットを削除するには、Amazon RDS API オペレーションの DeleteDBSnapshot を使用します。

DB スナップショットを削除するには、以下のパラメータを使用します。

- `DBSnapshotIdentifier` - DB スナップショットの識別子。

DB インスタンスへの復元

このセクションでは、DB インスタンスに復元する方法を示します。このページでは、DB スナップショットから Amazon RDS DB インスタンスに復元する方法を説明します。

Amazon RDS は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、その DB インスタンス全体をバックアップします。DB スナップショットからの復元で、新しい DB インスタンスを作成できます。復元の元となる DB スナップショットの名前を指定し、復元によって作成される新しい DB インスタンスの名前を指定します。DB スナップショットから既存の DB インスタンスに復元することはできません。スナップショットを復元すると新しい DB インスタンスが作成されます。

ステータスが `available` になると、復元された DB インスタンスを使用できます。DB インスタンスはバックグラウンドでデータをロードし続けます。これは遅延ロードと呼ばれています。まだロードされていないデータにアクセスする場合、DB インスタンスはリクエストされたデータを Amazon S3 から即座にダウンロードし、残りのデータをバックグラウンドでロードし続けます。詳細については、「Amazon EBS スナップショット」を参照してください。

クイックアクセスが必要なテーブルに対する遅延ロードの影響を軽減するには、`SELECT * など、テーブル全体をスキャンするようなオペレーションを実行します。これにより、Amazon RDS はバックアップされたテーブルデータをすべて S3 からダウンロードできます。

復元された DB インスタンスの初期化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。初期化中、`StorageOperationStatus` は `Initializing` に設定され、初期化が完了すると両方のフィールドは存在しなくなります。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

> **Important**
>
> 2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。マグネティックボリュームのスナップショットを復元する場合は、汎用 SSD (gp3) やプロビジョンド IOPS SSD (io2 Block Express) など、別のストレージタイプを選択する必要があります。Amazon RDS ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

DB インスタンスを復元し、出典 DB スナップショットとは異なるストレージタイプを使用できます。この場合、新しいストレージタイプにデータを移行するための追加作業が必要になるため、復元処理が遅くなります。

CloudFormationを使用して、DB インスタンスのスナップショットから DB インスタンスを復元できます。詳細については、AWS CloudFormationユーザーガイドの AWS::RDS::DBInstance を参照してください。

> **Note**
>
> 共有され暗号化された DB スナップショットから、DB インスタンスを復元することはできません。代わりに、DB スナップショットのコピーを作成し、そのコピーから DB インスタンスを復元できます。詳細については、「Amazon RDS の DB スナップショットのコピー」を参照してください。

RDS 延長サポートバージョンで DB インスタンスを復元する方法については、「Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

スナップショットからの復元

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB のスナップショットから DB インスタンスを復元できます。

> **Note**
>
> DB インスタンスを復元するときに、ストレージ容量を減らすことはできません。ストレージ割り当てを増やす場合は、少なくとも 10 パーセント単位で増やす必要があります。10 パーセントに満たない単位で増やそうとすると、エラーになります。RDS for SQL Server DB インスタンスを復元するときに、割り当てストレージを増やすことはできません。

コンソール

DB スナップショットから DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。

3. 復元の元にする DB スナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。
5. [Restore snapshot] (スナップショットの復元) ページで、[DB instance identifier] (DB インスタンス識別子) に、復元された DB インスタンスの名前を入力します。
6. 割り当てられたストレージサイズなど、その他の設定を指定します。

   各設定の詳細については、「DB インスタンスの設定」を参照してください。
7. [DB インスタンスの復元] を選択します。

AWS CLI

DB スナップショットから DB インスタンスを復元するには、AWS CLI の DBスナップショットから DBインスタンスを復元する コマンドを使用します。

この例では、「mydbsnapshot」という以前作成した DB スナップショットから復元します。mynewdbinstance という名前の新しい DB インスタンスに復元します。この例では、割り当てられたストレージサイズも設定しています。

他の設定を指定できます。各設定の詳細については、「DB インスタンスの設定」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier mynewdbinstance \
    --db-snapshot-identifier mydbsnapshot \
    --allocated-storage 100
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier mynewdbinstance ^
    --db-snapshot-identifier mydbsnapshot ^
    --allocated-storage 100
```

このコマンドは、次のような出力を返します :

```
DBINSTANCE  mynewdbinstance  db.t3.small  MySQL     50       sa              creating
 3  n  8.0.28  general-public-license
```

Example

次の例は、新しく作成されたインスタンスにストレージボリュームを追加しながらスナップショットを復元する方法を示しています。スナップショットには、追加のボリューム rdsdbdata2 が含まれていました。復元オペレーションでは、rdsdbdata3 が追加され、新しく作成されたインスタンスに合計 3 つのボリュームが作成されます。スナップショットを復元するときにボリュームを削除することはできません。

```
aws rds restore-db-instance-from-db-snapshot \
     --db-instance-identifier my-restored-instance \
     --db-snapshot-identifier my-asv-snapshot \
     --additional-storage-volumes '[{ \
             "VolumeName": "rdsdbdata3", \
             "StorageType":"gp3", \
             "AllocatedStorage": 5000, \
             "IOPS": 12000 \
         }]'
```

RDS API

DB スナップショットから DB インスタンスを復元するには、以下のパラメータを指定して Amazon RDS API 関数 RestoreDBInstanceFromDBSnapshot を呼び出します。

- DBInstanceIdentifier
- DBSnapshotIdentifier

考慮事項

DB スナップショットから DB インスタンスに復元する際の考慮事項については、以下のトピックを参照してください。

トピック

- パラメータグループに関する考慮事項
- セキュリティグループに関する考慮事項
- オプショングループに関する考慮事項

- リソースへのタグ付けに関する考慮事項
- Db2 に関する考慮事項
- Microsoft SQL Server に関する考慮事項
- MySQL に関する考慮事項
- Oracle データベースに関する考慮事項

パラメータグループに関する考慮事項

復元された DB インスタンスを適切なパラメータグループと関連付けることができるように、作成する DB スナップショットの DB パラメータグループは保持しておくことをお勧めします。

別のパラメータグループを選択しない限り、デフォルトの DB パラメータグループが、復元されたインスタンスに関連付けられます。デフォルトのパラメータグループでは、カスタムのパラメータ設定は使用できません。

パラメータグループは、DB インスタンスを復元する際に指定できます。

DB パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

セキュリティグループに関する考慮事項

DB インスタンスを復元すると、仮想プライベートクラウド (VPC)、DB サブネットグループ、および VPC セキュリティグループはデフォルトのものが、(それらに別のものを選択しない限り) 復元されたインスタンスに関連付けられます。

- Amazon RDS コンソールを使用している場合は、カスタムの VPC セキュリティグループを指定してインスタンスに関連付けるか、新しい VPC セキュリティグループを作成できます。
- AWS CLI を使用している場合、`restore-db-instance-from-db-snapshot` コマンドで `--vpc-security-group-ids` オプションを指定することにより、カスタムの VPC セキュリティグループを指定して、それをインスタンスに関連付けることができます。
- Amazon RDS API を使用している場合、`VpcSecurityGroupIds.VpcSecurityGroupId.N` パラメータを `RestoreDBInstanceFromDBSnapshot` アクションに含むことができます。

復元が完了し、新しい DB インスタンスが使用可能になり次第、その DB インスタンスを変更して VPC 設定を変更することもできます。詳しくは、「Amazon RDS DB インスタンスを変更する」を参照してください。

オプショングループに関する考慮事項

DB インスタンスを復元すると、多くの場合、デフォルトの DB オプショングループが復元された DB インスタンスに関連付けられます。

この例外となるのは、ソース DB インスタンスが、永続的もしくは不変のオプションを含むオプショングループに関連付けられている場合です。例えば、ソース DB インスタンスで Oracle TDE (Transparent Data Encryption) が使用されている場合、復元された DB インスタンスでは、TDE オプションを含むオプショングループを使用する必要があります。

DB インスタンスを異なる VPC に復元する場合は、以下のいずれかの操作を実行して、そのインスタンスに DB オプショングループを割り当てる必要があります。

- その VPC グループのデフォルトのオプショングループを、復元したインスタンスに割り当てる。
- VPC にリンクされているオプショングループを、復元したインスタンスに割り当てる。
- 新しいオプショングループを作成し、DB インスタンスに割り当てます。Oracle TDE などの永続的または不変のオプションを使用する場合は、その永続的または不変のオプションを含む新しいオプショングループを作成する必要があります。

DB オプショングループの詳細については、「オプショングループを使用する」を参照してください。

リソースへのタグ付けに関する考慮事項

DB スナップショットから DB インスタンスが復元されると、RDS は新しいタグが指定されているかどうかを確認します。新しいタグが指定されている場合、そのタグは復元された DB インスタンスに追加されます。新しいタグがない場合、RDS は、スナップショットの作成時にソース DB インスタンスから復元された DB インスタンスにタグを追加します。

詳細については、「タグを DB スナップショットにコピーする」を参照してください。

Db2 に関する考慮事項

BYOL モデルでは、Amazon RDS for Db2 DB インスタンスを、IBM Site ID と IBM Customer ID を含むカスタムパラメータグループに関連付ける必要があります。そうしないと、スナップショットから DB インスタンスを復元しようとすると失敗します。Amazon RDS for Db2 DB インスタンスは、AWS License Manager セルフマネージドライセンスにも関連付ける必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL)」を参照してください。

AWS Marketplace 経由の Db2 ライセンスモデルでは、使用する特定の IBM Db2 エディションの有効な AWS Marketplace サブスクリプションが必要です。まだお持ちでない場合は、IBM Db2 エディションの Db2 サブスクリプションを AWS Marketplace で購入する必要があります。詳細については、「AWS Marketplace 経由の Db2 ライセンス」を参照してください。

Microsoft SQL Server に関する考慮事項

RDS for Microsoft SQL Server DB スナップショットを新しいインスタンスに復元すると、スナップショットと同じエディションに常に復元できるようになります。場合によっては、DB インスタンスのエディションを変更することもできます。エディションを変更する場合の制限は次のとおりです。

- DB スナップショットでは、新しいエディション用に十分なストレージを割り当てる必要があります。
- 次のエディションの変更以外はサポートされていません。
  - Standard Edition から Enterprise Edition
  - Web Edition から Standard Edition または Enterprise Edition
  - Express Edition から Web Edition、Standard Edition、Enterprise Edition のいずれか

スナップショットを復元して、サポートされていない新しいエディションに変更する場合は、ネイティブバックアップおよび復元機能を試すことができます。SQL Server は、データベース上で有効にした SQL Server の機能に基づき、データベースが新しいエディションと互換性があるかどうかを検証します。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

MySQL に関する考慮事項

サポートされていないエンジンバージョンを使用して RDS for MySQL DB スナップショットから復元するには、DB スナップショットを複数回アップグレードする必要がある場合があります。アップグレードオプションの詳細については、「RDS for MySQL でサポートされていないエンジンバージョンの DB スナップショットのアップグレードオプション」を参照してください。

RDS for MySQL DB スナップショットのエンジンバージョンのアップグレードの詳細については、「MySQL DB スナップショットエンジンバージョンのアップグレード」を参照してください。

Oracle データベースに関する考慮事項

Oracle データベースを DB スナップショットから復元する場合、次の点に注意します。

- DB スナップショットを復元する前に、それを新しい Oracle データベースリリースにアップグレードできます。詳細については、「Oracle DB スナップショットのアップグレード」を参照してください。
- シングルテナント設定を使用する CDB インスタンスのスナップショットを復元する場合は、PDB 名を変更できます。CDB インスタンスがマルチテナント設定を使用している場合は PDB 名を変更できません。詳細については、「CDB のバックアップと復元」を参照してください。
- CDB 名は変更できません。これは常に RDSCDB です。この CDB 名は、すべての CDB インスタンスで同じです。
- DB スナップショット内のテナントデータベースを直接操作することはできません。マルチテナント設定を使用する CDB インスタンスのスナップショットを復元する場合は、そのテナントデータベースをすべて復元します。describe-db-snapshot-tenant-databases を使用すると、復元する前に DB スナップショット内のテナントデータベースを点検できます。
- Oracle GoldenGate を使用している場合、必ず `compatible` パラメータと同時にパラメータグループを保持します。DB スナップショットから DB インスタンスを復元するときは、等しいか大きい `compatible` 値を使用したパラメータグループを指定します。
- DB スナップショットを復元するときに、データベースの名前を変更することもできます。オンライン REDO ログの合計サイズが 20GB を超える場合、RDS はオンライン REDO ログのサイズをデフォルト設定の 512MB (4 x 128MB) にリセットすることがあります。サイズが小さいほど、妥当な時間内に復元オペレーションを完了できます。後でオンライン REDO ログを再作成し、サイズを変更できます。
- マスターユーザーパスワードは AWS Secrets Manager で管理できます。詳細については、「AWS Secrets Manager を使用したマスターユーザーパスワード管理の概要」を参照してください。

Amazon RDS の DB インスタンスを特定の時点に復元する

DB インスタンスを特定の時点に復元し、ソース DB インスタンスを変更せずに新しい DB インスタンスを作成できます。

DB インスタンスを特定の時点に復元する場合、デフォルトの仮想プライベートクラウド (VPC) のセキュリティグループを選択できます。また、DB インスタンスにカスタム VPC セキュリティグループを適用することもできます。

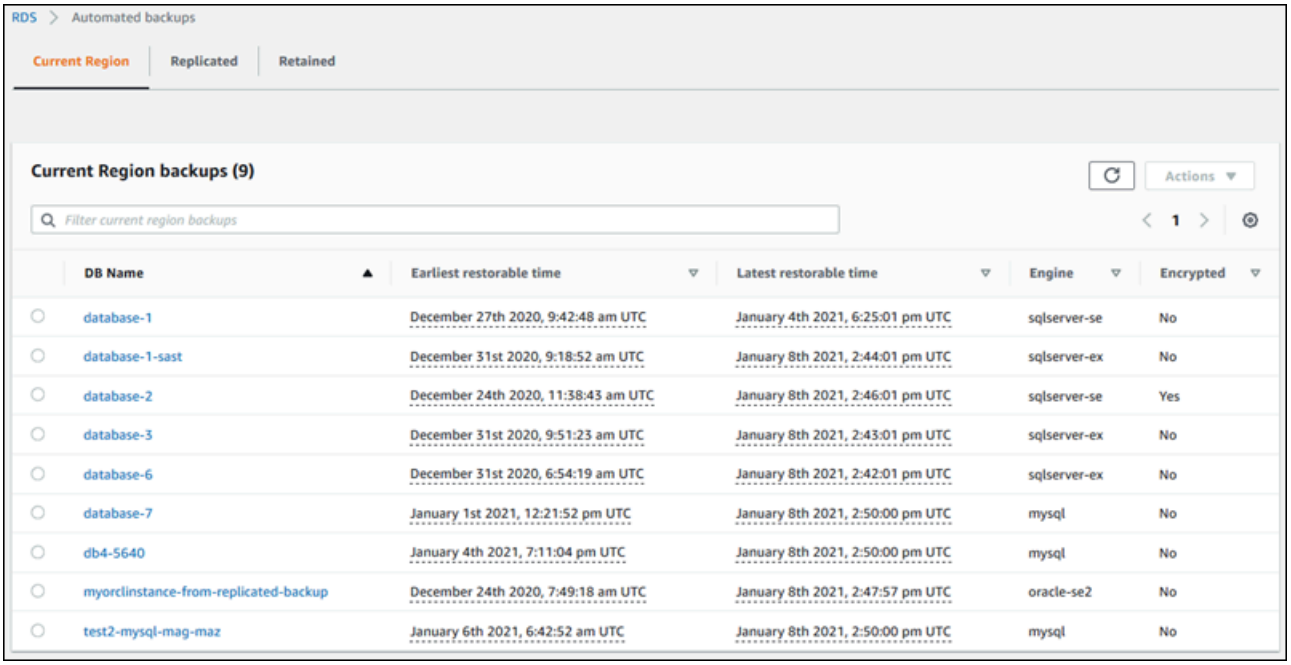
復元が完了し、DB インスタンスが利用可能になると、そのボリュームは Amazon S3 からのデータブロックをバックグラウンドでロードし続けます。この期間中、DB インスタンスは完全に動作しますが、初期化が完了するまでパフォーマンスが最大にならない場合があります。ボリュームごとの初期化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。初期化中、`StorageOperationStatus` は `Initializing` に設定され、初期化が完了すると両方のフィールドは存在しなくなります。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

復元された DB インスタンスは、デフォルトの DB パラメータグループとオプショングループに自動的に関連付けられます。ただし、カスタムパラメータグループとオプショングループは、AWS CLI または RDS API を使用して復元中に指定することで適用できます。コンソールを使用して復元する場合は、復元の完了後にパラメータグループとオプショングループを変更できます。

リクエストでタグが指定されている場合、指定されたタグは復元された DB インスタンスに適用されます。リクエストでタグが指定されておらず、ソース DB インスタンスがリージョン内でアクティブでタグがある場合、RDS はソース DB インスタンスから復元された DB インスタンスに最新のタグを追加します。

RDS は、DB インスタンスのトランザクションログを 5 分ごとに Amazon S3 にアップロードします。DB インスタンスの復元可能な直近の時間を確認するには、AWS CLI の describe-db-instances コマンドを使用し、DB インスタンスの [`LatestRestorableTime`] フィールドに返される値を確認します。Amazon RDS コンソールで各 DB インスタンスの復元可能な直近の時間を表示するには、[自動バックアップ] をクリックします。

バックアップ保持期間の任意の時点に復元できます。各 DB インスタンスの復元可能な最も早い時間を表示するには、Amazon RDS コンソールで [自動バックアップ] をクリックします。



> **Note**
>
> 出典 DB インスタンスとしてプロビジョンド IOPS ストレージを使用する場合は、同じ、または類似の DB インスタンスサイズと IOPS に復元することをお勧めします。例えば、互換性のない IOPS 値を持つ DB インスタンスのサイズを選択した場合、エラーが発生することがあります。

RDS 延長サポートバージョンで DB インスタンスを復元する方法については、「Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

いくつかの Amazon RDS データベースエンジンには、特定の時点から復元する際の特別な考慮事項があります。

- Amazon RDS for Db2 DB インスタンスでパスワード認証を使用する場合、`rdsadmin.add_user` を含むユーザー管理アクションはログにキャプチャされません。これらのアクションには、完全なスナップショットバックアップが必要です。

  BYOL モデルでは、RDS for Db2 DB インスタンスを、IBM Site ID と IBM Customer ID を含むカスタムパラメータグループに関連付ける必要があります。そうしないと、DB インスタンスを特定の時点に復元しようとすると失敗します。Amazon RDS for Db2 DB インスタンスは、AWS License Manager セルフマネージドライセンスにも関連付ける必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL)」を参照してください。

AWS Marketplace 経由の Db2 ライセンスモデルでは、使用する特定の IBM Db2 エディションの有効な AWS Marketplace サブスクリプションが必要です。まだお持ちでない場合は、IBM Db2 エディションの Db2 サブスクリプションを AWS Marketplace で購入する必要があります。詳細については、「AWS Marketplace 経由の Db2 ライセンス」を参照してください。

- 特定の時点に RDS for Oracle DB インスタンスを復元するとき、復元された DB インスタンスに異なる DB エンジン、ライセンスモデル、DBName (SID) を指定できます。RDS がマスターユーザーパスワードを AWS Secrets Manager に保存するように指定することもできます。詳細については、「AWS Secrets Manager を使用したマスターユーザーパスワード管理の概要」を参照してください。
- 特定の時点に Microsoft SQL Server DB インスタンスを復元する際、そのインスタンス内の各データベースは、インスタンス内の他のデータベースからそれぞれ 1 秒以内の時点に復元されます。インスタンス内の複数のデータベースにまたがるトランザクションは、復元後の整合性がない可能性があります。
- SQL Server DB インスタンスの場合、`OFFLINE`、`EMERGENCY`、`SINGLE_USER` のモードはサポートされていません。任意のデータベースをこれらのモードの 1 つに設定すると、復元可能な直近の時間がインスタンス全体で前に進まなくなります。
- SQL Server データベースの復旧モデルを変更するなど、一部のアクションの結果、ポイントインタイムリカバリに使用されるログの順序が乱れる可能性があります。場合によっては、Amazon RDS がこの問題を検出し、復元可能な最新の時間が先に進まなくなります。SQL Server データベースが`BULK_LOGGED`リカバリモデルを使用する場合など、その他の場合、ログシーケンスの中断は検出されません。ログ順序に乱れがある場合、SQL Server DB インスタンスを特定の時点に復元することができなくなる可能性があります。これらの理由により、Amazon RDS では SQL Server データベースの復旧モデルの変更はサポートされていません。

また、AWS Backup を使用して、Amazon RDS DB インスタンスのバックアップを管理することもできます。DB インスタンスが AWS Backup のバックアッププランに関連付けられている場合、そのバックアッププランはポイントインタイムリカバリに使用されます。AWS Backup で作成されたバックアップの名前は `awsbackup:`*`AWS-Backup-job-number`* で終わります。AWS Backup の詳細については、「AWS Backup デベロッパーガイド」を参照してください。

> **Note**
>
> このトピックの情報は、Amazon RDS に適用されます。Amazon Aurora DB クラスターの復元についての詳細は、「DBinstance を指定した時点へ復元する」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB インスタンスを特定の時点に復元できます。

> **Note**
>
> DB インスタンスを復元するとき、ストレージの量を削減することはできません。ストレージ割り当てを増やす場合は、少なくとも 10 パーセント単位で増やす必要があります。10 パーセントに満たない単位で増やそうとすると、エラーになります。RDS for SQL Server DB インスタンスを復元するときに、割り当てストレージを増やすことはできません。

コンソール

特定の時点に DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups] (自動バックアップ) を選択します。

   [Current Region] (現在のリージョン) タブに自動バックアップが表示されます。
3. 復元する DB インスタンスを選択します。
4. 「アクション」 で、「特定時点への復元」 を選択します。

   [特定時点への復元] ウィンドウが表示されます。
5. 「Latest restorable time」 を選択してできるだけ最新の時点に復元するか、「カスタム」 を選択して時刻を選択します。

   「カスタム」 を選択した場合、インスタンスクラスターを復元する日時を入力します。

   > **Note**
   >
   > 時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。
6. [DB インスタンス識別子] に、ターゲットが復元された DB インスタンスの名前を入力します。名前は一意である必要があります。
7. 必要に応じて、DB インスタンスクラス、ストレージ、ストレージのオートスケーリングを使用するかどうかなど、他のオプションを選択します。

各設定の詳細については、「DB インスタンスの設定」を参照してください。

8. [Restore to point in time] (特定時点への復元) を選択します。

AWS CLI

特定の時点に DB インスタンスを復元するには、AWS CLI コマンド restore-db-instance-to-point-in-time コマンドを使用して、新しい DB インスタンスを作成します。この例では、割り当てられたストレージサイズを設定し、ストレージの自動スケーリングを有効にします。

このオペレーションでは、リソースのタグ付けがサポートされています。--tags オプションを使用すると、ソース DB インスタンスのタグは無視され、指定されたタグが使用されます。それ以外の場合は、ソースインスタンスの最新のタグが使用されます。

他の設定を指定できます。各設定の詳細については、「DB インスタンスの設定」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier mysourcedbinstance \
    --target-db-instance-identifier mytargetdbinstance \
    --restore-time 2017-10-14T23:45:00.000Z \
    --allocated-storage 100 \
    --max-allocated-storage 1000
```

Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier mysourcedbinstance ^
    --target-db-instance-identifier mytargetdbinstance ^
    --restore-time 2017-10-14T23:45:00.000Z ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000
```

Example

次の例は、インスタンスを特定の時点に復元するときにボリュームを追加する方法を示しています。

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
     --source-db-instance-identifier my-asv-instance \
     --target-db-instance-identifier my-pitr-instance \
     --use-latest-restorable-time \
     --additional-storage-volumes '[{ \
             "VolumeName": "rdsdbdata2", \
             "StorageType":"gp3", \
             "AllocatedStorage": 5000, \
             "IOPS": 12000 \
         }]'
```

Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
     --source-db-instance-identifier my-asv-instance ^
     --target-db-instance-identifier my-pitr-instance ^
     --use-latest-restorable-time ^
     --additional-storage-volumes '[{ ^
             "VolumeName": "rdsdbdata2", ^
             "StorageType":"gp3", ^
             "AllocatedStorage": 5000, ^
             "IOPS": 12000 ^
         }]'
```

RDS API

DB インスタンスを特定の時間に復元するには、以下のパラメータを指定して Amazon RDS API の RestoreDBInstanceToPointInTime オペレーションを呼び出します。

- SourceDBInstanceIdentifier
- TargetDBInstanceIdentifier
- RestoreTime

マルチ AZ DB クラスターを指定の時点の状態に復元する

マルチ AZ DB クラスターを指定の時点に復元し、新しいマルチ AZ DB クラスターを作成できます。

RDSは、マルチ AZ DB クラスターのトランザクションログを Amazon S3 に継続的にアップロードします。バックアップ保持期間の任意の時点に復元できます。マルチ AZ DB クラスターの復元可能な最も早い時間を表示するには、AWS CLI describe-db-clustersコマンドを使用します。DB クラスターの`EarliestRestorableTime`フィールドに返される値を確認します。マルチ AZ DB クラスターの復元可能な直近の時間を確認するには、DB クラスターの `LatestRestorableTime` フィールドに返される値を確認します。

マルチ AZ DB クラスターを特定の時点に復元する場合、マルチ AZ DBクラスターのデフォルトの VPC セキュリティグループを選択するか、カスタム VPC セキュリティグループをマルチ AZ DB クラスターに適用できます。

復元されたマルチ AZ DB クラスターは、デフォルトの DB クラスターパラメータグループに自動的に関連付けられます。ただし、復元時に指定することで、カスタム DB クラスターパラメータグループを適用できます。

ソース DB クラスターにリソースタグがある場合は、RDS では復元される DB クラスターに最新のタグを追加します。

> **Note**
>
> ソース DB クラスターと同じまたは類似のサイズでマルチ AZ DB クラスターを復元することをお勧めします。また、プロビジョンド IOPS ストレージを使用している場合は、同じまたは類似の IOPS 値で復元することをお勧めします。例えば、互換性のない IOPS 値を持つ DB クラスターサイズを選択した場合、エラーが発生する可能性があります。
> ソースのマルチ AZ DB クラスターが汎用 SSD (gp3) ストレージを使用しており、割り当てられたストレージが 400 GiB 未満の場合、復元された DB クラスターのリードレプリカのプロビジョンド IOPS は変更できません。

RDS 延長サポートバージョンを使用したマルチ AZ DB クラスターの復元については、「Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、マルチ AZ DB クラスターを特定の時点に復元できます。

コンソール

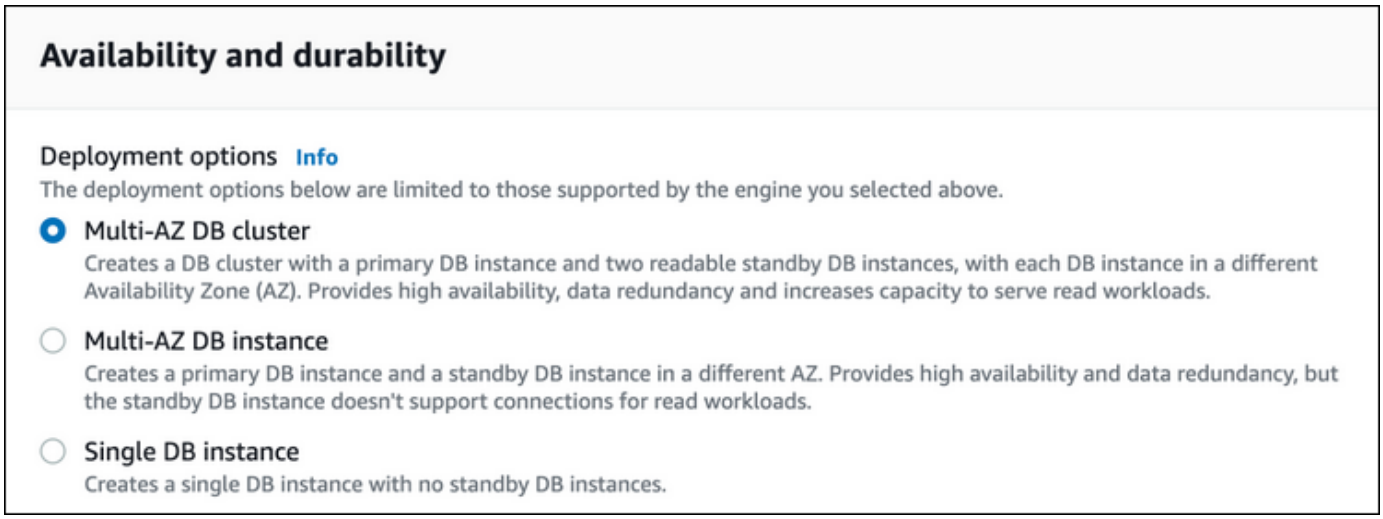
マルチ AZ DB クラスターを特定の時点に復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. 復元するマルチ AZ DBクラスターを選択します。
4. 「アクション」 で、「特定時点への復元」 を選択します。

   [特定時点への復元] ウィンドウが表示されます。
5. 「Latest restorable time」 を選択してできるだけ最新の時点に復元するか、「カスタム」 を選択して時刻を選択します。

   「カスタム」 を選択した場合、マルチAZ DB クラスターを復元する日時を入力します。

   > **Note**
   >
   > 時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。
6. 「DB クラスター識別子」 に、復元されるマルチ AZ DB クラスターの名前を入力します。
7. [Availability and durability] (可用性と耐久性) で、[Multi-AZ DB cluster] (マルチ AZ DB クラスター)-選択します。

8. 「DB インスタンスクラス」 で、DB インスタンスのクラスを選択します。

現在、マルチ AZ DB クラスターは db.m6gd および db.r6gd DB インスタンスクラスのみをサポートしています。DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

9. 残りのセクションで、DB クラスター設定を指定します。各設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。
10. [Restore to point in time] (特定時点への復元) を選択します。

AWS CLI

マルチ AZ DB クラスターを指定の時点の状態に復元するには、AWS CLIコマンドrestore-db-cluster-to-point-in-timeを使用して、マルチ AZ DB クラスターを作成します。

現在、マルチ AZ DB クラスターは db.m6gd および db.r6gd DB インスタンスクラスのみをサポートしています。DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier mysourcemultiazdbcluster \
    --db-cluster-identifier mytargetmultiazdbcluster \
    --restore-to-time 2021-08-14T23:45:00.000Z \
    --db-cluster-instance-class db.r6gd.xlarge
```

Windows の場合:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier mysourcemultiazdbcluster ^
    --db-cluster-identifier mytargetmultiazdbcluster ^
    --restore-to-time 2021-08-14T23:45:00.000Z ^
    --db-cluster-instance-class db.r6gd.xlarge
```

RDS API

DB クラスターを特定の時間に復元するには、以下のパラメータを指定して Amazon RDS API のRestoreDBClusterToPointInTimeオペレーションを呼び出します。

- SourceDBClusterIdentifier
- DBClusterIdentifier
- RestoreToTime

スナップショットからマルチ AZ DB クラスターへの復元

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、スナップショットをマルチ AZ DB クラスタに復元できます。次のタイプのスナップショットは、マルチ AZ DB クラスターに復元できます。

- シングル AZ デプロイのスナップショット
- 単一の DB インスタンスを使用したマルチ AZ DB クラスターのデプロイのスナップショット
- マルチ AZ DB クラスターのスナップショット

マルチ AZ デプロイについては、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

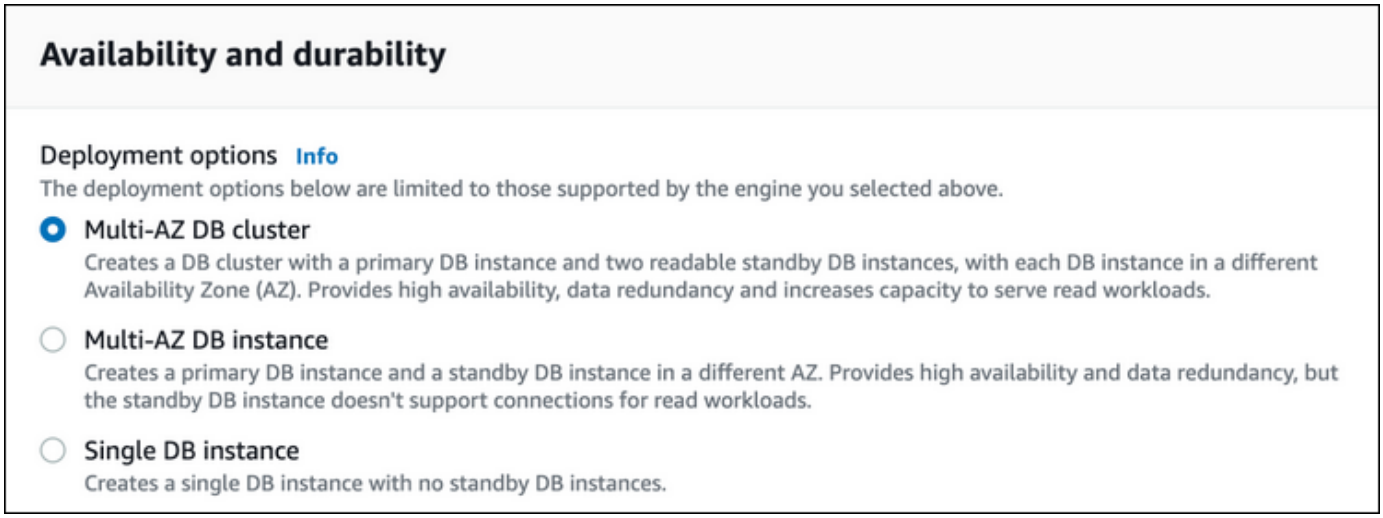
> **Tip**
>
> スナップショットを復元することで、シングル AZ デプロイまたはマルチ AZ DB クラスターデプロイをマルチ AZ DB クラスターデプロイに移行できます。

RDS 延長サポートバージョンを使用したマルチ AZ DB クラスターの復元については、「Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

コンソール

スナップショットをマルチ AZ DB クラスターに復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. 復元の元にする スナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。
5. [Restore snapshot] (スナップショットの復元) ページの [Availability and durability] (可用性と耐久性) で、[Multi-AZ DB cluster] (マルチ AZ DB クラスター) を選択します。



6. 「DB クラスター識別子」 に、新しく復元される マルチ AZ DB クラスターの名前を入力します。
7. 残りのセクションで、DB クラスター設定を指定します。各設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。
8. DB インスタンスの復元 を選択します。

AWS CLI

スナップショットをマルチ AZ DB クラスターに復元するには、AWS CLIコマンドrestore-db-cluster-from-snapshotを使用します。

次の例では、以前作成した「mysnapshot」という名前のスナップショットから復元します。mynewmultiazdbclusterと言う名前の新しいマルチ AZ DBクラスターに復元します。また、マルチ AZ DB クラスター内の DB インスタンスによって使用される DB インスタンスクラスも指定します。DB エンジンにmysqlまたはpostgresを指定します。

--snapshot-identifier オプションでは、名前または Amazon リソースネーム (ARN) のいずれかを使用して、DB クラススナップショットを指定できます。ただし、DB スナップショットを指定するには ARN のみを使用できます。

--db-cluster-instance-class オプションとして、新しいマルチ AZ DB クラスターの DB インスタンスクラスを指定します。マルチ AZ DB クラスターは、db.m6gd および db.r6gd DB インスタンスクラスなど、特定の DB インスタンスクラスのみをサポートしています。DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。

他のオプションも指定できます。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewmultiazdbcluster \
    --snapshot-identifier mysnapshot \
    --engine mysql|postgres \
    --db-cluster-instance-class db.r6gd.xlarge
```

Windows の場合:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewmultiazdbcluster ^
    --snapshot-identifier mysnapshot ^
    --engine mysql|postgres ^
    --db-cluster-instance-class db.r6gd.xlarge
```

DB クラスターを復元した後、スナップショットの作成に使用した DB クラスターまたは DB インスタンスに関連付けられたセキュリティグループに、マルチ AZ DB クラスターを追加することができます。このアクションを完了すると、以前の DB クラスターまたは DB インスタンスと同じ機能が得られます。

RDS API

スナップショットをマルチ AZ DBクラスターに復元するには、次のパラメータを使用してRDS API オペレーションRestoreDBClusterFromSnapshotを呼び出します。

- `DBClusterIdentifier`
- `SnapshotIdentifier`
- `Engine`

他のオプションのパラメータを指定することもできます。

DB クラスターを復元した後、スナップショットの作成に使用した DB クラスターまたは DB インスタンスに関連付けられたセキュリティグループに、マルチ AZ DB クラスターを追加することができます。このアクションを完了すると、以前の DB クラスターまたは DB インスタンスと同じ機能が得られます。

マルチ AZ DB クラスターのスナップショットから DB インスタンスへの復元

マルチ AZ DB クラスターのスナップショットは、DB クラスターのストレージボリュームスナップショットであり、個々のデータベースだけでなく、DB クラスター全体をバックアップします。マルチ AZ DB クラスターのスナップショットをシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイに復元することができます。マルチ AZ デプロイについては、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

> **Note**
>
> マルチ AZ DB クラスターのスナップショットを新しいマルチ AZ DB クラスターに復元することもできます。手順については、「スナップショットからマルチ AZ DB クラスターへの復元」を参照してください。

RDS 延長サポートバージョンを使用したマルチ AZ DB クラスターの復元については、「Amazon RDS 延長サポートでの DB インスタンスまたはマルチ AZ DB クラスターの復元」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、マルチ AZ DB クラスターのスナップショットをシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイに復元します。

コンソール

マルチ AZ DB クラスターのスナップショットをシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイに復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. 復元の元にするマルチ AZ DB クラスターのスナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。
5. [Restore snapshot] (スナップショットの復元) ページの [Availability and durability] (可用性と耐久性) で、次のいずれかを選択します。

- [Single DB instance] (シングル DB インスタンス) — スタンバイ DB インスタンスのない 1 つの DB インスタンスにスナップショットを復元します。
- [Multi-AZ DB instance] (マルチ AZ DB インスタンス) — スナップショットを 1 つのプライマリ DB インスタンスと 1 つのスタンバイ DB インスタンスを持つマルチ AZ DB インスタンスデプロイに復元します。

6. [DB instance identifier] (DB インスタンス識別子) として、復元された DB インスタンスの名前を入力します。
7. 残りのセクションで、DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。
8. DB インスタンスの復元 を選択します。

AWS CLI

マルチ AZ DB クラスターのスナップショットを DB インスタンスデプロイに復元するには、AWS CLI コマンド restore-db-instance-from-db-snapshot を使用します。

次の例では、以前作成した `myclustersnapshot` という名前のマルチ AZ DB クラスターのスナップショットから復元します。`mynewdbinstance` という名前のプライマリ DB インスタンスを持つ新しいマルチ AZ DB インスタンスデプロイに復元します。`--db-cluster-snapshot-identifier` オプションとして、マルチ AZ DB クラスタースナップショットの名前を指定します。

`--db-instance-class` オプションとして、新しい DB インスタンスデプロイの DB インスタンスクラスを指定します。DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。

他のオプションも指定できます。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier mynewdbinstance \
    --db-cluster-snapshot-identifier myclustersnapshot \
    --engine mysql \
    --multi-az \
    --db-instance-class db.r6g.xlarge
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier mynewdbinstance ^
    --db-cluster-snapshot-identifier myclustersnapshot ^
    --engine mysql ^
    --multi-az ^
    --db-instance-class db.r6g.xlarge
```

DB インスタンスを復元した後、スナップショットの作成に使用したマルチ AZ DB クラスターに関連付けられたセキュリティグループに、DB インスタンスを追加できます。このアクションを完了すると、以前の マルチ AZ DB クラスターと同じ機能が得られます。

RDS API

マルチ AZ DB クラスターのスナップショットを DB インスタンスデプロイに復元するには、次のパラメータを指定して、RDS APIオペレーションRestoreDBInstanceFromDBSnapshot を呼び出します。

- `DBInstanceIdentifier`
- `DBClusterSnapshotIdentifier`
- `Engine`

他のオプションのパラメータを指定することもできます。

DB インスタンスを復元した後、スナップショットの作成に使用したマルチ AZ DB クラスターに関連付けられたセキュリティグループに、DB インスタンスを追加できます。このアクションを完了すると、以前の マルチ AZ DB クラスターと同じ機能が得られます。

チュートリアル: DB スナップショットからの Amazon RDS DB インスタンスの復元

Amazon RDS で作業していると、ときどき DB インスタンスで作業をするが常に必要なわけではないということがよくあります。例えば、四半期ごとに行う顧客アンケートで、Amazon EC2 インスタンスを使用して顧客アンケートのウェブサイトをホストしているとします。また、アンケート結果の保存に使用される DB インスタンスもあります。このようなシナリオで経費を節約する 1 つの方法として、アンケートが完了した後、DB インスタンスの DB スナップショットを作成することが挙げられます。その後、アンケートを再度実施する必要がある場合は、DB インスタンスを削除して復元します。

DB インスタンスを復元するときは、復元の元となる DB スナップショットの名前を指定します。次に、復元オペレーションから作成される新しい DB インスタンスの名前を指定します。

スナップショットから DB インスタンスを復元する方法の詳細については、「DB インスタンスへの復元」を参照してください。

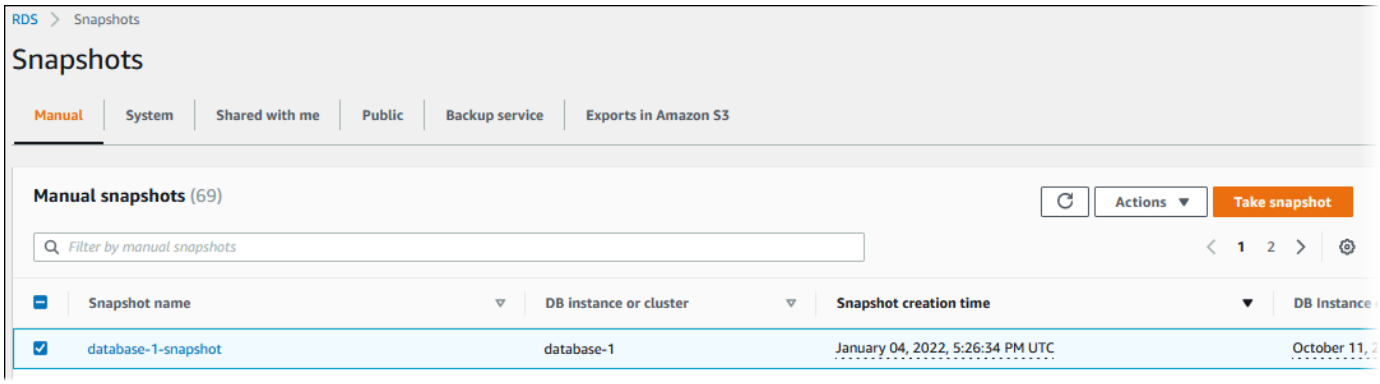
Amazon RDS の AWS KMS キー管理については、「AWS KMS key 管理」を参照してください。

DB スナップショットからの DB インスタンスの復元

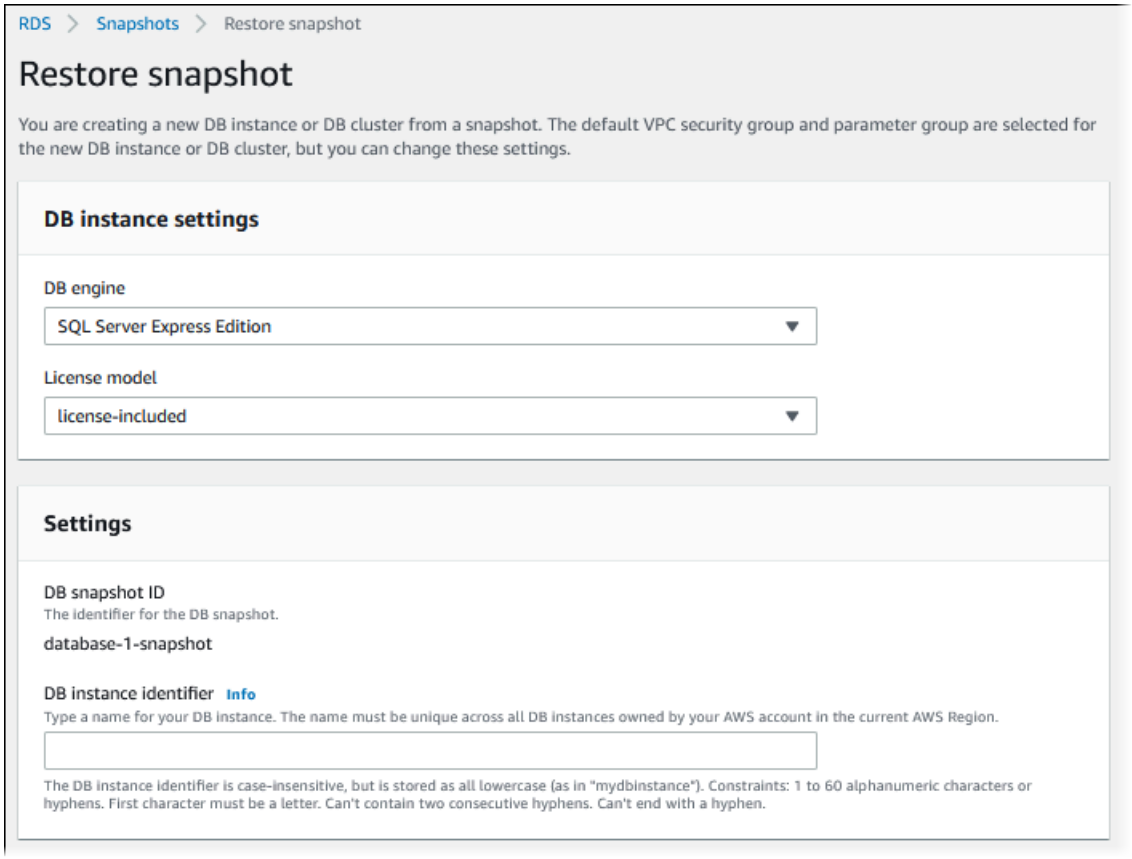
次の手順に従って、AWS マネジメントコンソール でスナップショットから復元します。

DB スナップショットから DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. 復元の元にする DB スナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。



[Restore snapshot (スナップショットの復元)] ページが表示されます。



5. [DB instance settings] (DB インスタンスの設定) で、[DB engine] (DB エンジン) および [License model] (ライセンスモデル) にデフォルトの設定を使用します (Oracle または Microsoft SQL Server の場合)。
6. [Settings] (設定) の [DB Instance identifier] (DB インスタンス識別子) に、復元する DB インスタンスに使用する名前を入力します (例: **`mynewdbinstance`**)。

   DB スナップショットの作成後に削除した DB インスタンスから復元する場合は、その DB インスタンスの名前を使用できます。
7. [可用性と耐久性] で、別のアベイラビリティーゾーンでスタンバイインスタンスを作成するかどうかを選択します。

   このチュートリアルでは、スタンバイインスタンスを作成しないでください。
8. [Connectivity] (接続) では、次のデフォルトの設定を使用します。

   - 仮想プライベートクラウド (VPC)
   - DB サブネットグループ

- パブリックアクセス
- VPC セキュリティグループ (ファイアウォール)

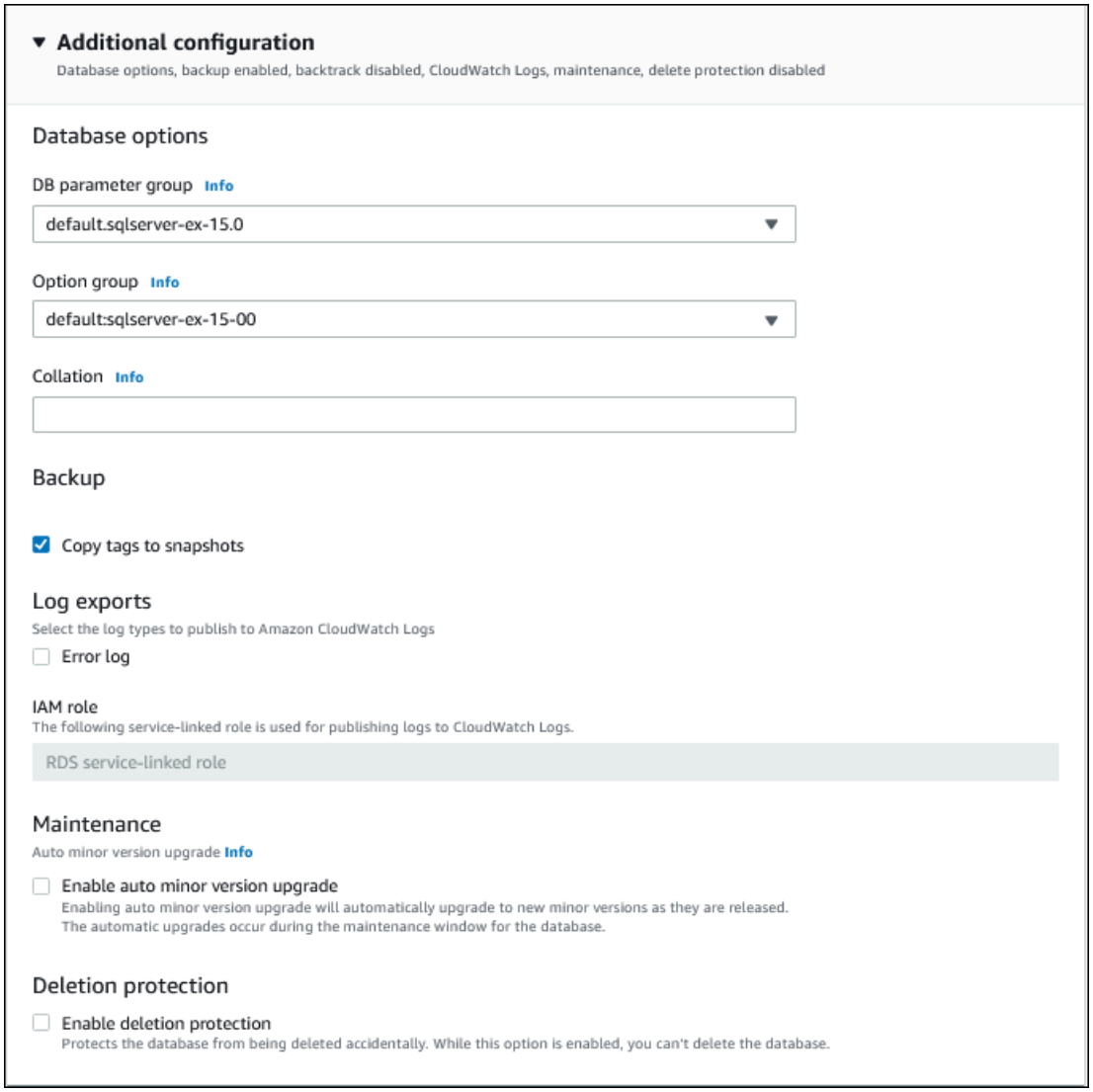
9. DB インスタンスクラスを選択します。

   このチュートリアルでは、[Burstable classes (includes t classes)] (バースト可能クラス (t クラスを含む)) を選択してから、[db.t3.small] を選択します。

10. [Encryption] (暗号化) では、デフォルトの設定を使用します。

    スナップショットのソース DB インスタンスが暗号化されている場合、復元される DB インスタンスも暗号化されます。暗号化を解除することはできません。

11. [Additional configuration] (追加設定) を展開します。



12. [Database options] (データベースオプション) で、次の操作を行います。

a. [DB parameter group] (DB パラメータグループ) を選択します。

このチュートリアルでは、デフォルトのパラメータグループを使用します。

b. [Option group] (オプショングループ) を選択します。

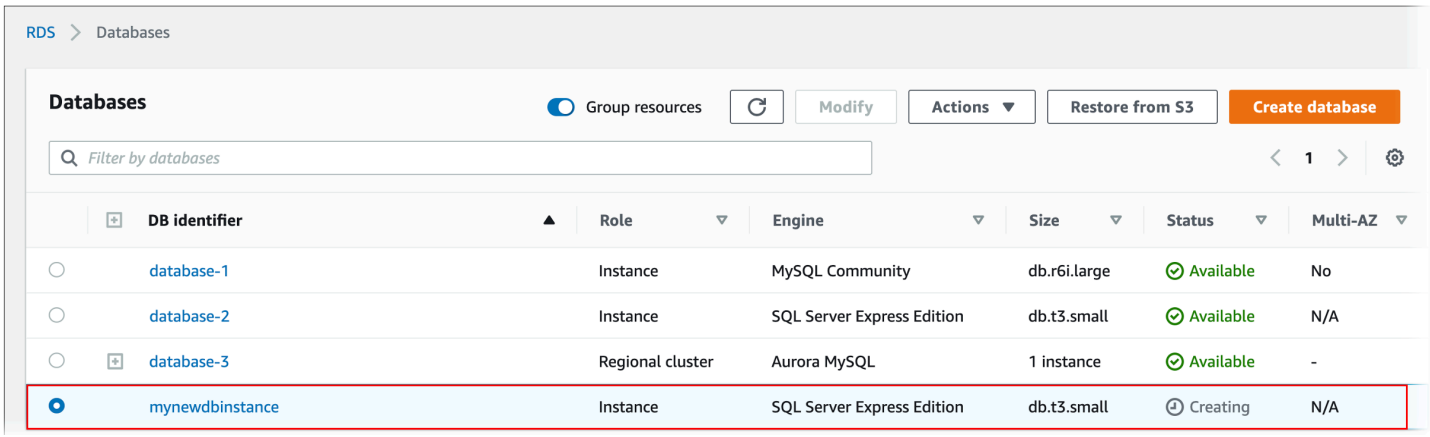
このチュートリアルでは、デフォルトのオプショングループを使用します。

> **Important**
>
> 場合によっては、持続的オプションまたは永続的オプションを使用する DB インスタンスの DB スナップショットから復元を行うことができます。その場合は、同じオプションを使用するオプショングループを選択してください。

c. [Deletion protection] (削除保護) で、[Enable deletion protection] (削除保護の有効化) チェックボックスをオンにします。

13. DB インスタンスの復元 を選択します。

[Databases] (データベース) ページには、ステータスが `Creating` である復元された DB インスタンスが表示されます。



Amazon RDS の DB スナップショットのコピー

Amazon RDS を使用すると、自動バックアップまたは手動 DB スナップショットをコピーできます。スナップショットをコピーすると、そのコピーは手動スナップショットになります。自動バックアップまたは手動スナップショットは複数のコピーを作成できますが、各コピーには一意の識別子が必要です。

同じ AWS リージョン 内、AWS リージョン 間でスナップショットをコピーできます。また、共有スナップショットをコピーできます。スナップショットは、1 つのステップで別の AWS リージョンまたはアカウントにコピーできます。

DB スナップショットのコピー

AWS アカウント ごとに、AWS リージョン 間で同時に最大 20 個の DB スナップショットをコピーできます。別の AWS リージョン に DB スナップショットをコピーする場合には、その AWS リージョン に保持されている手動 DB スナップショットを作成します。コピー元の AWS リージョン から DB スナップショットをコピーすると、Amazon RDS のデータ転送料金が発生します。

データ転送料金の詳細については、「Amazon RDS の料金」を参照してください。

DB スナップショットのコピーが新しい AWS リージョン に作成されると、その DB スナップショットのコピーは、AWS リージョン にある他のすべての DB スナップショットと同じように動作します。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB スナップショットをコピーできます。

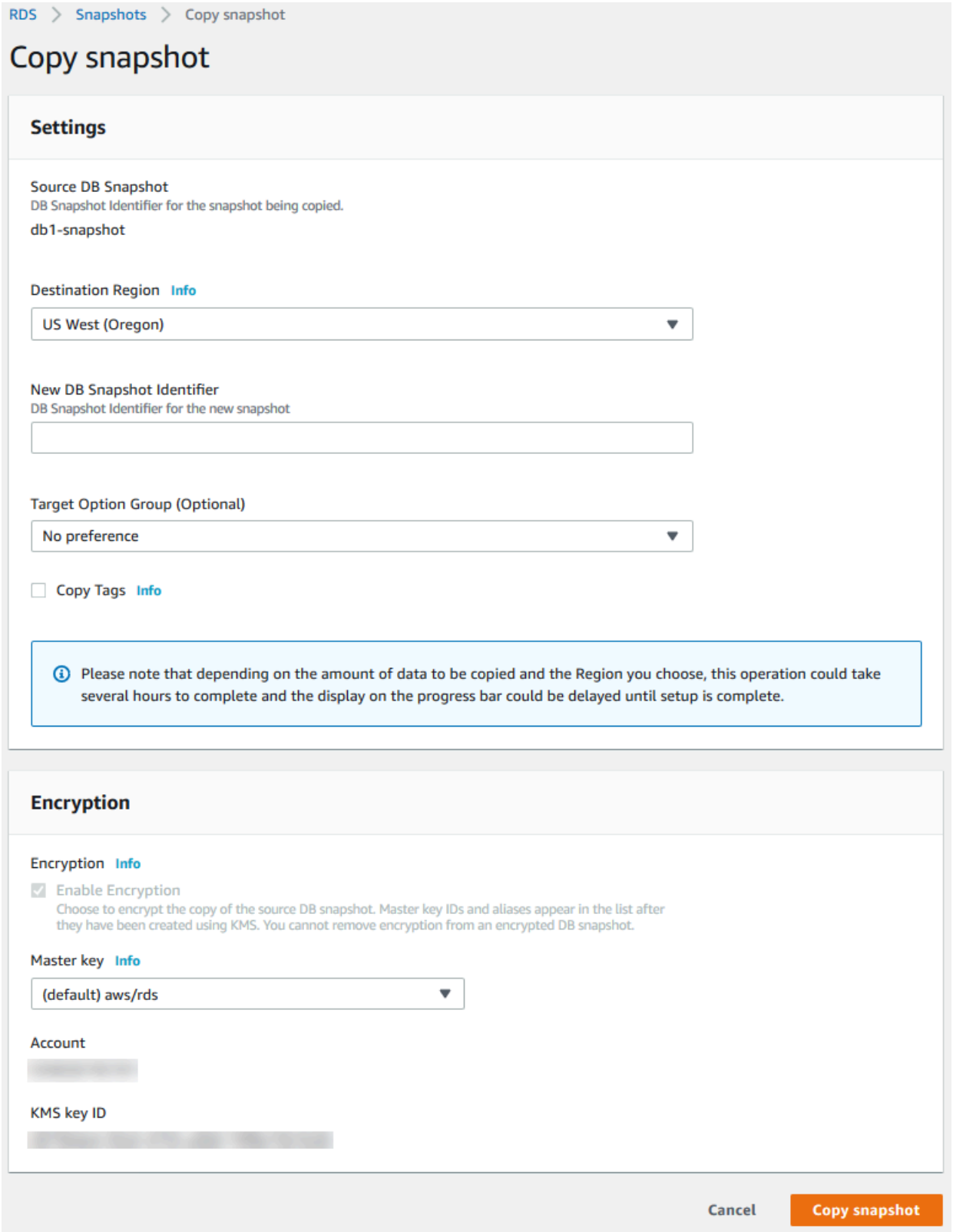
コンソール

以下の手順では、AWS マネジメントコンソール を使用して、暗号化されている DB スナップショットや暗号化されていない DB スナップショットを、同じ AWS リージョン 内または異なるリージョン間でコピーできます。

DB スナップショットをコピーするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. コピーする DB スナップショットを選択します。
4. [アクション] で、[スナップショットをコピー] を選択します。

[スナップショットをコピー] ページが表示されます。



5. (オプション) [Target Option Group] (ターゲットオプショングループ)–には、必要に応じて新しいオプショングループを選択します。

   AWS リージョン 間でスナップショットをコピーし、DB インスタンスでデフォルト以外のオプショングループを使用する場合に、このオプションを指定します。

   コピー元の DB インスタンスで Transparent Data Encryption for Oracle または Microsoft SQL Server を使用する場合は、リージョン間でコピーするときに、このオプションを指定する必要があります。詳細については、「オプショングループの考慮事項」を参照してください。

6. (オプション) 別の AWS リージョン に DB スナップショットをコピーするには、[Destination Region] (コピー先リージョン) でその新しい AWS リージョン を選択します。

   > **Note**
   >
   > コピー先 AWS リージョン は、コピー元 AWS リージョン と同じデータベースエンジンのバージョンを使用できる必要があります。

7. [New DB Snapshot Identifier] (新しい DB スナップショット ID) には、DB スナップショットのコピーの名前を入力します。

   自動バックアップまたは手動スナップショットは複数のコピーを作成できますが、各コピーには一意の識別子が必要です。

8. (オプション) スナップショットからスナップショットのコピーにタグと値をコピーするには、[Copy Tags] を選択します。

9. (オプション) [暗号化] では、次の操作を行います。

   a. DB スナップショットは暗号化されていないが、コピーを暗号化する場合、[暗号を有効化] を選択します。

      > **Note**
      >
      > DB スナップショットが暗号化されている場合は、コピーを暗号化する必要があります。そのため、チェックボックスは既にオンになっています。

   b. [AWS KMS key] で、DB スナップショットコピーの暗号化に使用する KMS キー識別子を指定します。

10. [スナップショットのコピー] を選択します。

AWS CLI

AWS CLI の copy-db-snapshot コマンドを使用して、DB スナップショットをコピーできます。スナップショットのコピー先が新しい AWS リージョン である場合は、その新しい AWS リージョン でコマンドを実行します。

DB スナップショットをコピーするには、以下のオプションを使用します。状況に応じてオプションを使い分けることができます。以下の説明と例を参考にして、どのオプションを使用するか判断してください。

- `--source-db-snapshot-identifier` - コピー元の DB スナップショットの識別子。
  - スナップショットのコピー元とコピー先が同じ AWS リージョン にある場合は、有効な DB スナップショットの識別子を指定します。例えば、`rds:mysql-instance1-snapshot-20130805`。
  - スナップショットのコピー元とコピー先が同じ AWS リージョン にあり、AWS アカウント と共有されている場合は、有効な DB スナップショットの ARN を指定します。例えば、`arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`。
  - スナップショットのコピー元とコピー先が異なる AWS リージョン である場合は、有効な DB スナップショットの ARN を指定します。例えば、`arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`。
  - 共有された手動 DB スナップショットからコピーする場合、このパラメータは、共有された DB スナップショットの Amazon リソースネーム (ARN) であることが必要です。
  - 暗号化されたスナップショットをコピーする場合、このパラメータは、コピー元の AWS リージョン の ARN 形式であること、さらに `PreSignedUrl` パラメータの `SourceDBSnapshotIdentifier` と一致することが必要です。
- `--target-db-snapshot-identifier` - 暗号化された DB スナップショットの新しいコピーの識別子。
- `--copy-option-group` – と共有されているスナップショットからオプショングループをコピーします。AWS アカウント
- `--copy-tags` - スナップショットからスナップショットのコピーにタグと値をコピーするために使用します。
- `--option-group-name` - スナップショットのコピーに関連付けるオプショングループ。

  AWS リージョン 間でスナップショットをコピーし、DB インスタンスでデフォルト以外のオプショングループを使用する場合に、このオプションを指定します。

コピー元の DB インスタンスで Transparent Data Encryption for Oracle または Microsoft SQL Server を使用する場合は、リージョン間でコピーするときに、このオプションを指定する必要があります。詳しくは、「オプショングループの考慮事項」を参照してください。

- `--kms-key-id` - 暗号化された DB スナップショットの KMS キー識別子。KMS キー識別子は、KMS キーの Amazon リソースネーム (ARN)、キー識別子、またはキーエイリアスです。
  - AWS アカウント から暗号化された DB スナップショットをコピーする場合、このパラメータの値を指定して、新しい KMS キーでコピーを暗号化できます。このパラメータの値を指定しないと、DB スナップショットのコピーはコピー元の DB スナップショットと同じ KMS キーで暗号化されます。
  - 別の AWS アカウント から共有されている暗号化された DB スナップショットをコピーする場合、このパラメータの値を指定する必要があります。
  - このパラメータを指定して暗号化されていないスナップショットをコピーすると、コピーは暗号化されます。
  - 暗号化されたスナップショットを別の AWS リージョン にコピーする場合は、コピー先 AWS リージョン の KMS キーを指定する必要があります。KMS キーは、それらが作成された AWS リージョン に固有のものであるため、ある AWS リージョン の暗号化キーを別の AWS リージョン で使用することはできません。

Example暗号化されていない出典を同じリージョン内でコピー

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で、コピー元のスナップショットと同じ AWS リージョン に作成します。コピーが作成されると、元のスナップショット DB オプショングループとタグがコピーされたスナップショットにコピーされます。

Linux、macOS、Unix の場合:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --copy-option-group \
    --copy-tags
```

Windows の場合:

```
aws rds copy-db-snapshot ^
```

```
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-
instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --copy-option-group ^
    --copy-tags
```

Example暗号化されていない出典をリージョン間でコピー

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で、コマンドの実行先の AWS リージョン に作成します。

Linux、macOS、Unix の場合:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-
instance1-snapshot-20130805 \
    --target-db-snapshot-identifier mydbsnapshotcopy
```

Windows の場合:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-
instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy
```

Example暗号化された出典をリージョン間でコピー

次のコード例では、米国西部 (オレゴン) リージョンから US East (N. Virginia) リージョンに暗号化された DB スナップショットをコピーします。コピー先リージョン (us-east-1) でコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-
instance1-snapshot-20161115 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --kms-key-id my-us-east-1-key \
    --option-group-name custom-option-group-name
```

Windows の場合:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --kms-key-id my-us-east-1-key ^
    --option-group-name custom-option-group-name
```

--source-region パラメータは、AWS GovCloud (米国東部) と AWS GovCloud (米国西部) リージョン間で暗号化されたスナップショットをコピーする場合に必要です。--source-region には、ソース DB インスタンスの AWS リージョン を指定します。

--source-region を指定しない場合、--pre-signed-url の値を指定する必要があります。署名付きの URL は、ソースの AWS リージョン で呼び出される copy-db-snapshot コマンドに対する、署名バージョン 4 で署名されたリクエストを含む URL です。pre-signed-url オプションの詳細については、AWS CLI コマンドリファレンスの「copy-db-snapshot」を参照してください。

RDS API

Amazon RDS API の CopyDBSnapshot オペレーションを使用して、DB スナップショットをコピーできます。スナップショットのコピー先が新しい AWS リージョン である場合は、その新しい AWS リージョン でアクションを実行します。

DB スナップショットをコピーするには、以下のパラメータを使用します。状況に応じてパラメータを使い分けることができます。以下の説明と例を参考にして、どのパラメータを使用するか判断してください。

- SourceDBSnapshotIdentifier - コピー元の DB スナップショットの識別子。
  - スナップショットのコピー元とコピー先が同じ AWS リージョン にある場合は、有効な DB スナップショットの識別子を指定します。例えば、rds:mysql-instance1-snapshot-20130805。
  - スナップショットのコピー元とコピー先が同じ AWS リージョン にあり、AWS アカウント と共有されている場合は、有効な DB スナップショットの ARN を指定します。例えば、arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805。
  - スナップショットのコピー元とコピー先が異なる AWS リージョン である場合は、有効な DB スナップショットの ARN を指定します。例えば、arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805。

  - 共有された手動 DB スナップショットからコピーする場合、このパラメータは、共有された DB スナップショットの Amazon リソースネーム (ARN) であることが必要です。
  - 暗号化されたスナップショットをコピーする場合、このパラメータは、コピー元の AWS リージョン の ARN 形式であること、さらに `PreSignedUrl` パラメータの `SourceDBSnapshotIdentifier` と一致することが必要です。
- `TargetDBSnapshotIdentifier` - 暗号化された DB スナップショットの新しいコピーの識別子。
- `CopyOptionGroup` - 共有スナップショットからスナップショットのコピーにオプショングループをコピーするには、このパラメータを `true` に設定します。デフォルトは `false` です。
- `CopyTags` - スナップショットからスナップショットのコピーにタグと値をコピーするには、このパラメータを `true` に設定します。デフォルト: `false`。
- `OptionGroupName` - スナップショットのコピーに関連付けるオプショングループ。

  AWS リージョン 間でスナップショットをコピーし、DB インスタンスでデフォルト以外のオプショングループを使用する場合に、このパラメータを指定します。

  コピー元の DB インスタンスで Transparent Data Encryption for Oracle または Microsoft SQL Server を使用する場合は、リージョン間でコピーするときに、このパラメータを指定する必要があります。詳しくは、「オプショングループの考慮事項」を参照してください。
- `KmsKeyId` - 暗号化された DB スナップショットの KMS キー識別子。KMS キー識別子は、KMS キーの Amazon リソースネーム (ARN)、キー識別子、またはキーエイリアスです。
  - AWS アカウント から暗号化された DB スナップショットをコピーする場合、このパラメータの値を指定して、新しい KMS キーでコピーを暗号化できます。このパラメータの値を指定しないと、DB スナップショットのコピーはコピー元の DB スナップショットと同じ KMS キーで暗号化されます。
  - 別の AWS アカウント から共有されている暗号化された DB スナップショットをコピーする場合、このパラメータの値を指定する必要があります。
  - このパラメータを指定して暗号化されていないスナップショットをコピーすると、コピーは暗号化されます。
  - 暗号化されたスナップショットを別の AWS リージョン にコピーする場合は、コピー先 AWS リージョン の KMS キーを指定する必要があります。KMS キーは、それらが作成された AWS リージョン に固有のものであるため、ある AWS リージョン の暗号化キーを別の AWS リージョン で使用することはできません。

- `PreSignedUrl` - コピー元の DB スナップショットがあるコピー元の AWS リージョン で `CopyDBSnapshot` API オペレーションに対する署名バージョン 4 で署名されたリクエストを含む URL。

  Amazon RDS API を使用して、暗号化された DB スナップショットを別の AWS リージョン からコピーするときに、このパラメータを指定します。AWS CLI を使用して、暗号化された DB スナップショットを別の AWS リージョン からコピーするときは、このパラメータの代わりにコピー元のリージョンのオプションを指定できます。

  署名付き URL は、コピー元の暗号化された DB スナップショットがあるコピー元の AWS リージョン で実行できる `CopyDBSnapshot` API オペレーションに対する有効なリクエストでなければなりません。署名付き URL リクエストでは、以下のパラメータ値を指定する必要があります。

  - `DestinationRegion` - 暗号化された DB スナップショットのコピー先の AWS リージョン。これは、この署名付き URL がある、`CopyDBSnapshot` オペレーションの呼び出し元の AWS リージョン と同じです。

    例えば、us-west-2 リージョンから us-east-1 リージョンに暗号化された DB スナップショットをコピーするとします。次に、us-east-1 リージョンで `CopyDBSnapshot` オペレーションを呼び出し、us-west-2 リージョンでの `CopyDBSnapshot` オペレーションへの呼び出しを含む事前署名された URL を指定します。この例では、署名付き URL の `DestinationRegion` を us-east-1 リージョンに設定する必要があります。

  - `KmsKeyId` - コピー先 で DB スナップショットのコピーを暗号化するのに使用するキーの KMS キー識別子。AWS リージョンこれは、コピー先の AWS リージョン で呼び出す `CopyDBSnapshot` オペレーションおよび署名付き URL に含まれているオペレーションの両方で同じ識別子です。

  - `SourceDBSnapshotIdentifier` - コピー元の暗号化されたスナップショットの DB スナップショット識別子。この識別子は、ソース AWS リージョン の Amazon リソースネーム (ARN) 形式であることが必要です。例えば、暗号化された DB スナップショットを us-west-2 リージョンからコピーする場合、`SourceDBSnapshotIdentifier` は次の `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115` の例のようになります。

  書名バージョン 4 の 署名要件についての詳細は、以下を参照してください。

  - Amazon Simple Storage Service API リファレンスの「リクエストの認証: クエリパラメータの使用 (AWS 署名バージョン 4)」
  - AWS 全般のリファレンスの「署名バージョン 4 の署名プロセス」

Example暗号化されていない出典を同じリージョン内でコピー

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で、コピー元のスナップショットと同じ AWS リージョン に作成します。コピーが作成されると、元のスナップショットのすべてのタグはコピーされたスナップショットにコピーされます。

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &CopyTags=true
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=mysql-instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example暗号化されていない出典をリージョン間でコピー

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で 米国西部 (北カリフォルニア) リージョンに作成します。

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example暗号化された出典をリージョン間でコピー

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で米国東部 (バージニア北部) リージョンに作成します。

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBSnapshot
    &KmsKeyId=my-us-east-1-key
    &OptionGroupName=custom-option-group-name
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBSnapshotIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Asnapshot%25253Amysql-instance1-snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20161115
    &TargetDBSnapshotIdentifier=mydbsnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

制限事項

スナップショットをコピーする際の制約は以下のとおりです。

- 中国 (北京) リージョンまたは 中国 (寧夏) リージョンとの間でスナップショットをコピーすることはできません。
- AWS GovCloud (米国東部) と AWS GovCloud (US-West) の間でスナップショットをコピーすることはできます。ただし、これらの GovCloud (US) リージョンと GovCloud (US) リージョン以外のリージョンの間でスナップショットをコピーすることはできません。
- ターゲットスナップショットが使用可能になる前に出典スナップショットを削除すると、スナップショットはコピーされない場合があります。ターゲットスナップショットのステータスが AVAILABLE になったことを確認してから、出典スナップショットを削除してください。
- アカウントあたり 1 つのコピー先リージョンに対して最大 20 のスナップショットコピーリクエストを実行できます。
- 同じソース DB インスタンスに対して複数のスナップショットコピーをリクエストすると、それらは内部的にキューに追加されます。後でリクエストされたコピーは、それ以前のスナップショットコピーが完了するまで開始されません。詳細については、AWS ナレッジセンターの「Why is my EC2 AMI or EBS snapshot creation slow? (EC2 AMI または EBS スナップショットの作成が遅いのはなぜですか?)」を参照してください。
- 関連する AWS リージョン およびデータのコピー量に応じて、リージョン間のスナップショットのコピーは完了するまでに長時間かかることがあります。場合によっては、特定のコピー元 リージョンから多数のクロスリージョンスナップショットコピーのリクエストが発生することがあります。このような場合、Amazon RDS は進行中のいくつかのコピーが完了するまで、そのコピー元リージョンからの新しいクロスリージョンコピーリクエストをキューに入れることがあります。コピーリクエストがキューに入っている間は、そのリクエストに関する進捗情報は表示されません。コピーがスタートしたときに、進捗情報は表示されます。
- 別のコピーをスタートするときにコピーがまだ保留中の場合は、初期のコピーが終了するまで 2 番目のコピーはスタートされません。
- マルチ AZ DB クラスターのスナップショットはコピーできません。
- io2 ボリュームを使用する DB インスタンスのスナップショットは、io2 Block Express ボリュームが利用可能な AWS リージョン にのみコピーできます。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

考慮事項

DB スナップショットをコピーする際の考慮事項については、以下のトピックを参照してください。

トピック

- スナップショット保持期限

- 共有スナップショットコピーに関する考慮事項
- 暗号化されたスナップショットのコピーに関する考慮事項
- 増分スナップショットのコピーに関する考慮事項
- リージョン間スナップショットのコピーに関する考慮事項
- オプショングループの考慮事項
- パラメータグループの考慮事項

スナップショット保持期限

Amazon RDS は自動スナップショットをいくつかの状況で削除します。

- 保持期間の終了時。
- DB インスタンスの自動スナップショットを無効にした場合。
- DB インスタンスを削除した場合。

自動スナップショットをより長期間保持したい場合は、コピーを手動スナップショットとして作成しすると、削除するまで保持されます。デフォルトのストレージ領域を超える場合、手動スナップショットにAmazon RDS ストレージコストが適用される場合があります。

バックアップストレージコストの詳細については、「Amazon RDS の料金」を参照してください。

共有スナップショットコピーに関する考慮事項

他の AWS アカウント アカウントにより共有されているスナップショットをコピーすることができます。場合によっては、別の AWS アカウント から共有された暗号化されたスナップショットをコピーすることがあります。このような場合、スナップショットの暗号化に使用された AWS KMS key へのアクセス権が必要になります。

> **Note**
>
> Amazon RDS ストレージの費用は、コピーした共有スナップショットに適用されます。Amazon RDS は、コピーしたスナップショットにソース DB インスタンスの ARN をアタッチする場合があります。

スナップショットが暗号化されていない場合、共有 DB スナップショットは、別の AWS リージョンにコピーできます。

> **Note**
>
> 同じ AWS リージョン 内の共有増分スナップショットのコピーは、暗号化されていない場合や、初期のフルスナップショットと同じ KMS キーを使用して暗号化されている場合にサポートされます。コピー時に別の KMS キーを使用して後続のスナップショットを暗号化すると、それらの共有スナップショットはフルスナップショットになります。詳細については、「増分スナップショットのコピーに関する考慮事項」を参照してください。

1 つのステップでのクロスリージョンコピーとクロスアカウントコピー

スナップショットのクロスリージョンおよびクロスアカウントを 1 つのアクションでコピーするには、まずスナップショットをターゲット AWS アカウントと共有する必要があります。スナップショットが暗号化されている場合は、AWS KMS キーをターゲット AWS アカウントと共有する必要があります。スナップショットがデフォルト AWS KMS キーで暗号化されている場合は、まずスナップショットをコピーしてカスタマーマネージドキーで再暗号化してから、ターゲットアカウントと共有する必要があります。共有されると、ターゲットアカウントからそのアカウント (リージョン内またはクロスリージョン) へのコピーを開始できます。

暗号化されたスナップショットのコピーに関する考慮事項

KMS キーを使用して暗号化されたスナップショットをコピーできます。暗号化された スナップショットをコピーする場合は、スナップショットのコピーも暗号化する必要があります。同じ AWS リージョン 内で暗号化されているスナップショットをコピーする場合、元のスナップショットと同じ KMS キーを使用してコピーを暗号化できます。または、別の KMS キー を指定することもできます。

リージョン間で、暗号化されたスナップショットをコピーする場合は、送信先の AWS リージョンで有効な KMS キーを指定する必要があります。これは、リージョン固有の KMS キーでも、マルチリージョンのキーでもかまいません。マルチリージョンの KMS キーの詳細については、「AWS KMS でマルチリージョンキーを使用する」を参照してください。

ソーススナップショットはコピープロセス全体で暗号化されたままになります。詳細については、「Amazon RDS の暗号化された DB インスタンスの制限事項」を参照してください。

さらに、暗号化されていない スナップショットのコピーも暗号化できます。この方法で、以前には暗号化されていなかった DB インスタンスに簡単に暗号化を追加できます。このためには、暗号化の準備ができた段階で、DB インスタンスのスナップショットを作成します。次に、そのスナップショットのコピーを作成し、KMS キーを指定してそのスナップショットコピーを暗号化します。こ

うすることで、この暗号化されたスナップショットから暗号化された DB インスタンスを復元できます。

> **Important**
>
> Amazon RDS などのマネージドサービスを使用する AWS KMS キーポリシーですべてのリソース (*) に明示的な拒否ステートメントを使用する場合は、リソース所有アカウントを許可する条件を指定する必要があります。拒否ルールに IAM ユーザーの例外が含まれている場合でも、この条件がないとコピー操作が失敗することがあります。

Amazon RDS の AWS KMS キー管理の詳細については、「AWS KMS key 管理」を参照してください。

暗号化されたスナップショットをコピーするために必要なアクセス許可

暗号された DB スナップショットをコピーするには、ユーザーに Amazon RDS 暗号化を使用するための次のアクセス許可が必要です。

- `kms:DescribeKey`
- `kms:CreateGrant`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKey`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncrypt`

IAM ポリシー条件を使用して AWS KMS の許可オペレーションを制限する場合は、Amazon RDS に必要なすべてのオペレーションがポリシーに含まれていることを確認してください。リージョン間で暗号化されたスナップショットをコピーするときに `KMSKeyNotAccessibleFault` エラーが表示された場合は、IAM ポリシーに上記のすべての必要な AWS KMS 権限オペレーションが含まれていることを確認します。

増分スナップショットのコピーに関する考慮事項

差分スナップショットには、同じ DB インスタンスの最新のスナップショット以降に変更されたデータのみが含まれます。差分スナップショットのコピーは高速であり、フルスナップショットコピーよりストレージコストが低くなります。

スナップショットコピーが増分かどうかは、最近完了したスナップショットコピーとソーススナップショットによって決定されます。最近のスナップショットコピーが削除され、次のコピーがフルコピーである場合、増分コピーではありません。スナップショットコピーは、ソーススナップショットと同じタイプになります。ソーススナップショットが差分スナップショットの場合、スナップショットのコピーは差分スナップショットになります。増分は、最新のスナップショット以降にソース DB インスタンスで発生した変更の数によっても決まります。

AWS アカウント 間でスナップショットをコピーするとき、次のすべての条件に合致する場合のみ、コピーは増分コピーになります。

- 最新のスナップショットコピーは同じソース DB インスタンスのもので、送信先アカウントに存在します。
- 送信先アカウントのスナップショットのコピーがすべて、暗号化されていないか、あるいは同じ CMK キーを使って暗号化されていた。暗号化されている場合、同じ系列である必要があります。つまり、継承の深さ、および対応するレベルで一貫した暗号化キーが一致している必要があります。
- ソース DB インスタンスがマルチ AZ インスタンスの場合、最後のスナップショットが取得されてから別の AZ にフェイルオーバーしていません。

次の例は、フルスナップショットと増分スナップショットの違いを示しています。これらは、共有スナップショットと非共有スナップショットの両方に適用されます。

スナップショット	Encryption key	フルまたは増分
S1	K1	フル
S2	K1	S1 の増分
S3	K1	S2 の増分
S4	K1	S3 の増分
S1 のコピー (S1C)	K2	フル
S2 のコピー (S2C)	K3	フル
S3 のコピー (S3C)	K3	S2C の増分

スナップショット	Encryption key	フルまたは増分
S4 のコピー (S4C)	K3	S3C の増分
S4 のコピー 2 (S4C2)	K4	フル

> **Note**
>
> これらの例では、スナップショット S2、S3、および S4 は、前のスナップショットがまだ存在する場合にのみ増分になります。
> コピーも同じです。スナップショットのコピー S3C および S4C は、前のコピーがまだ存在する場合にのみ増分になります。

AWS リージョン 間で増分スナップショットをコピーする方法については、「フルコピーと増分コピー」を参照してください。

リージョン間スナップショットのコピーに関する考慮事項

AWS リージョン 間で DB スナップショットをコピーできます。クロスリージョンスナップショットコピーにはデータ転送料金が適用されます。クロスリージョンのスナップショットコピーには、特定の制約と考慮事項が適用されます。

クロスリージョン DB スナップショットコピーのリクエスト

出典リージョンと通信してクロスリージョン DB のスナップショットコピーをリクエストするには、リクエスタ (IAM ロールまたは IAM ユーザー) が出典 DB スナップショットおよび出典リージョンへのアクセス権を持っている必要があります。

リクエスタの IAM ポリシーの特定の条件により、リクエストが失敗する可能性があります。次の例では、DB スナップショットを 米国東部 (オハイオ) から US East (N. Virginia) にコピーしていることを前提としています。これらの例は、リクエストが失敗する原因となるリクエスタの IAM ポリシー内の条件を示しています。

- リクエスタのポリシーには、`aws:RequestedRegion` の条件があります。

```
...
"Effect": "Allow",
```

```
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": "us-east-1"
    }
}
```

ポリシーが出典リージョンへのアクセスを許可していないため、リクエストは失敗します。リクエストを正常に実行するには、出典リージョンとコピー先リージョンの両方を指定します。

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:RequestedRegion": [
            "us-east-1",
            "us-east-2"
        ]
    }
}
```

- リクエスタのポリシーでは、出典 DB スナップショットへのアクセスが許可されていません。

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "arn:aws:rds:us-east-1:123456789012:snapshot:target-snapshot"
...
```

リクエストを正常に実行するには、出典スナップショットとターゲットスナップショットの両方を指定します。

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": [
    "arn:aws:rds:us-east-1:123456789012:snapshot:target-snapshot",
    "arn:aws:rds:us-east-2:123456789012:snapshot:source-snapshot"
]
```

```
...
```

- リクエスターのポリシーが aws:ViaAWSService を拒否します。

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "Bool": {"aws:ViaAWSService": "false"}
}
```

  出典リージョンとの通信は、リクエスタに代わって RDS によって行われます。リクエストを正常に実行するには、AWS のサービスからの呼び出しを拒否しないでください。

- リクエスタのポリシーには、aws:SourceVpc または aws:SourceVpce の条件があります。

  RDS がリモートリージョンへの呼び出しを行うとき、指定された VPC または VPC エンドポイントからの呼び出しではないため、これらのリクエストは失敗する可能性があります。

リクエストが失敗する原因となる前述の条件のいずれかを使用する必要がある場合は、ポリシーに aws:CalledVia とともに 2 番目のステートメントを含めて、リクエストを成功させることができます。例えば、次のように aws:CalledVia と aws:SourceVpce を使用できます。

```
...
"Effect": "Allow",
"Action": "rds:CopyDBSnapshot",
"Resource": "*",
"Condition": {
    "Condition" : {
        "ForAnyValue:StringEquals" : {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
     }
},
{
    "Effect": "Allow",
    "Action": [
        "rds:CopyDBSnapshot"
    ],
    "Resource": "*",
    "Condition": {
```

```
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": [
                "rds.amazonaws.com"
            ]
        }
    }
}
```

詳細については、IAM ユーザーガイド の 「IAM のポリシーとアクセス許可」を参照してください。

スナップショットコピーの承認

クロスリージョン DB スナップショットコピーリクエストが success を返した後、RDS はバックグラウンドでコピーをスタートします。RDS が出典スナップショットにアクセスするための承認が作成されます。この承認は、出典 DB スナップショットをターゲット DB スナップショットにリンクし、RDS が指定されたターゲットスナップショットにのみコピーできるようにします。

承認は、サービスにリンクされた IAM ロールの rds:CrossRegionCommunication アクセス許可を使用して RDS によって検証されます。コピーが承認されると、RDS は出典リージョンと通信し、コピーを完了します。

RDS は、CopyDBSnapshot リクエストによって以前に許可されていない DB スナップショットにアクセスできません。コピーが完了すると、認可は取り消されます。

RDS は、サービスリンクされたロールを使用して、出典リージョンでの承認を確認します。コピープロセス中にサービスリンクされたロールを削除すると、コピーは失敗します。

詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの使用」を参照してください。

AWS Security Token Service 認証情報の使用

グローバル AWS Security Token Service (AWS STS) エンドポイントからのセッショントークンは、デフォルトで有効になっている AWS リージョン (商用リージョン) でのみ有効です。assumeRole の AWS STS API 操作からの認証情報を使用する場合、出典リージョンがオプトインリージョンである場合は、そのリージョンのエンドポイントを使用します。それ以外の場合、このリクエストは失敗します。これは、認証情報が両方のリージョンで有効である必要があるために発生します。これは、そのリージョンの AWS STS エンドポイントが使用されている場合にのみオプトインリージョンに当てはまります。

グローバルエンドポイントを使用するには、オペレーションで両方のリージョンで有効になっていることを確認します。`Valid in all AWS #####` アカウント設定でグローバルエンドポイントを AWS STS に設定します。

署名付き URL パラメータの認証情報にも同じルールが適用されます。

詳細については、IAM ユーザーガイドの「AWS リージョン での AWS STS の管理」を参照してください。

レイテンシーと複数のコピーリクエスト

関連する AWS リージョン およびデータのコピー量に応じて、リージョン間のスナップショットのコピーは完了するまでに長時間かかることがあります。

場合によっては、特定のコピー元 AWS リージョン からの多数のクロスリージョンスナップショットコピーのリクエストが発生することがあります。このような場合、Amazon RDS は進行中のいくつかのコピーが完了するまで、そのコピー元 AWS リージョン からの新しいクロスリージョンコピーリクエストをキューに入れることがあります。コピーリクエストがキューに入っている間は、そのリクエストに関する進捗情報は表示されません。コピーがスタートされたときに、進捗情報は表示されます。

フルコピーと増分コピー

コピー元スナップショットとは異なる AWS リージョン にスナップショットをコピーする場合、初期のコピーでは、増分スナップショットをコピーした場合でもフルスナップショットコピーになります。フルスナップショットコピーには、DB インスタンスを復元するために必要なデータやメタデータすべてが含まれます。最初のスナップショットコピーの後、同じ DB インスタンスの増分スナップショットを同じ AWS アカウント 内の同じデスティネーションリージョンにコピーできます。増分スナップショットの詳細については、「増分スナップショットのコピーに関する考慮事項」を参照してください。

AWS リージョン 間での増分スナップショットコピーは、暗号化されていないスナップショットと暗号化されたスナップショットの両方がサポートされています。

別の AWS リージョン にスナップショットをコピーする場合、次の条件に合致すればコピーは増分となります。

- スナップショットが以前、コピー先のリージョンにコピーされたことがある。
- 最近のスナップショットコピーがコピー先のリージョンにまだ存在する。

- 送信先リージョンのスナップショットのコピーがすべて、暗号化されていないか、あるいは同じ KMS キーを使って暗号化されていた。

オプショングループの考慮事項

DB オプショングループは、作成元の AWS リージョン 専用であり、作成元の AWS リージョン と異なる AWS リージョン では使用できません。

Oracle データベースの場合、AWS CLI または RDS API を使用して、AWS アカウント と共有されているスナップショットからカスタム DB オプショングループをコピーできます。同じ AWS リージョン 内のオプショングループのみコピーできます。オプショングループは、宛先アカウントに既にコピーされており、コピー後に変更が加えられていない場合はコピーされません。ソースのオプショングループが以前にコピーされ、コピー後に変更されている場合、RDS では新しいバージョンを宛先アカウントにコピーします。デフォルトのオプショングループはコピーされません。

リージョン間でスナップショットをコピーする際に、スナップショットの新しいオプショングループを指定できます。新しいオプショングループは、スナップショットをコピーする前に作成することをお勧めします。オプショングループを作成するには、コピー先の AWS リージョン で、元の DB インスタンスと同じ設定を使用します。新しい AWS リージョン に既にある場合は、それを使用できます。

場合によっては、スナップショットをコピーし、そのスナップショットの新しいオプショングループを指定しないこともあるでしょう。このような場合、スナップショットを復元すると、DB インスタンスはデフォルトのオプショングループを取得します。新しい DB インスタンスでコピー元と同じ設定を使用するには、以下の操作を行います。

1. オプショングループを作成するには、コピー先の AWS リージョン で、元の DB インスタンスと同じ設定を使用します。新しい AWS リージョン に既にある場合は、それを使用できます。
2. コピー先の AWS リージョン でスナップショットを復元したら、新しい DB インスタンスを変更し、前のステップの新規または既存のオプショングループを追加します。

パラメータグループの考慮事項

リージョン間でスナップショットをコピーすると、コピーにはコピー元の DB インスタンスで使用されているパラメータグループは含まれません。スナップショットを復元して新しい DB インスタンスを作成すると、その DB インスタンスには、作成先 AWS リージョン でのデフォルトのパラメータグループが適用されます。新しい DB インスタンスでコピー元と同じパラメータを使用するには、以下の操作を行います。

1. DB パラメータグループを作成するには、コピー先の AWS リージョン で元の DB インスタンスと同じ設定を使用します。新しい AWS リージョン に既にある場合は、それを使用できます。
2. コピー先の AWS リージョン でスナップショットを復元したら、新しい DB インスタンスを変更し、前のステップの新規または既存のパラメータグループを追加します。

Amazon RDS の DB スナップショットを共有する

Amazon RDS を使用すると、次の方法で手動 DB スナップショットを共有できます。

- 手動 DB スナップショットを共有すると、暗号化されているかいないかに関係なく、権限のある AWS アカウント がスナップショットをコピーできるようになります。
- 暗号化されていない手動 DB スナップショットを共有すると、権限のある AWS アカウント が、DB インスタンスをコピーしてそこから復元するのではなく、スナップショットから DB インスタンスを直接復元できるようになります。ただし、共有され暗号化された DB スナップショットから、DB インスタンスを復元することはできません。代わりに、DB スナップショットのコピーを作成し、そのコピーから DB インスタンスを復元できます。

> **Note**
>
> 自動 DB スナップショットを共有するには、自動 DB スナップショットをコピーしてそのコピーを共有することで、手動スナップショットを作成します。このプロセスは、AWS Backup で生成されたリソースにも適用されます。

スナップショットのコピーの詳細については、「Amazon RDS の DB スナップショットのコピー」を参照してください。DB スナップショットから DB インスタンスを復元する方法については、「DB インスタンスへの復元」を参照してください。

手動スナップショットを最大 20 のその他の AWS アカウント と共有することができます。

手動スナップショットを他の AWS アカウントと共有する場合には、以下の制限が適用されます。

- AWS Command Line Interface (AWS CLI) または Amazon RDS API を使用して共有スナップショットから DB インスタンスを復元する際、スナップショット識別子として共有スナップショットの Amazon リソースネーム (ARN) を指定する必要があります。
- 固定オプションまたは永続オプションを持つオプショングループを使用する DB スナップショットを共有することはできません。`Timezone` または `OLS` オプション (あるいはその両方) を持つ Oracle DB インスタンスを除きます。

  固定オプションはオプショングループから削除できません。永続オプションを含むオプショングループは、そのオプショングループが DB インスタンスに割り当てられると、DB インスタンスから削除できなくなります。

次の表は、固定オプションおよび永続オプションと、それらに関連する DB エンジンをリストしています。

オプション名	永続	固定	DB エンジン
TDE	はい	いいえ	Microsoft SQL Server Enterprise Edition
TDE	はい	はい	Oracle Enterprise Edition
タイムゾーン	はい	はい	Oracle Enterprise Edition Oracle Standard Edition Oracle Standard Edition One Oracle Standard Edition 2

Oracle DB インスタンスの場合、`Timezone` または `OLS` オプション (あるいはその両方) を持つ共有 DB スナップショットをコピーできます。そのためには、DB スナップショットをコピーするときにこれらのオプションを含むターゲットオプショングループを指定します。OLS オプションは、Oracle バージョン 12.2 以降を実行している Oracle DB インスタンスに対してのみ恒久的かつ永続的です。これらのオプションの詳細については、「Oracle のタイムゾーン」および「Oracle Label Security」を参照してください。

- マルチ AZ DB クラスターのスナップショットは共有できません。

公開スナップショットの共有、暗号化されたスナップショットの共有、スナップショットの共有の停止については、以下のトピックを参照してください。

トピック

- Amazon RDS の公開スナップショットの共有
- Amazon RDS の暗号化されたスナップショットの共有
- Amazon RDS のスナップショット共有の停止

スナップショットの共有

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB スナップショットを共有できます。

コンソール

Amazon RDS コンソールを使用して、手動 DB スナップショットを最大 20 の AWS アカウント と共有することができます。また、コンソールを使用して、手動 スナップショットの 1 つ以上のアカウントとの共有を停止することもできます。

Amazon RDS コンソールを使用して、手動 DB スナップショットを共有するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. 共有する手動スナップショットを選択します。
4. [Actions] (アクション) で、[Share snapshot] (スナップショットの共有) を選択します。
5. [DB snapshot visibility] で次のいずれかのオプションを選択します。

   - ソースが暗号化されていない場合、[パブリック] を選択して、すべての AWS アカウントが DB インスタンスをマニュアル DB スナップショットから復元できるようにするか、[プライベート] を選択して、指定した AWS アカウントだけが、DB インスタンスをマニュアル DB スナップショットから復元できるようにします。

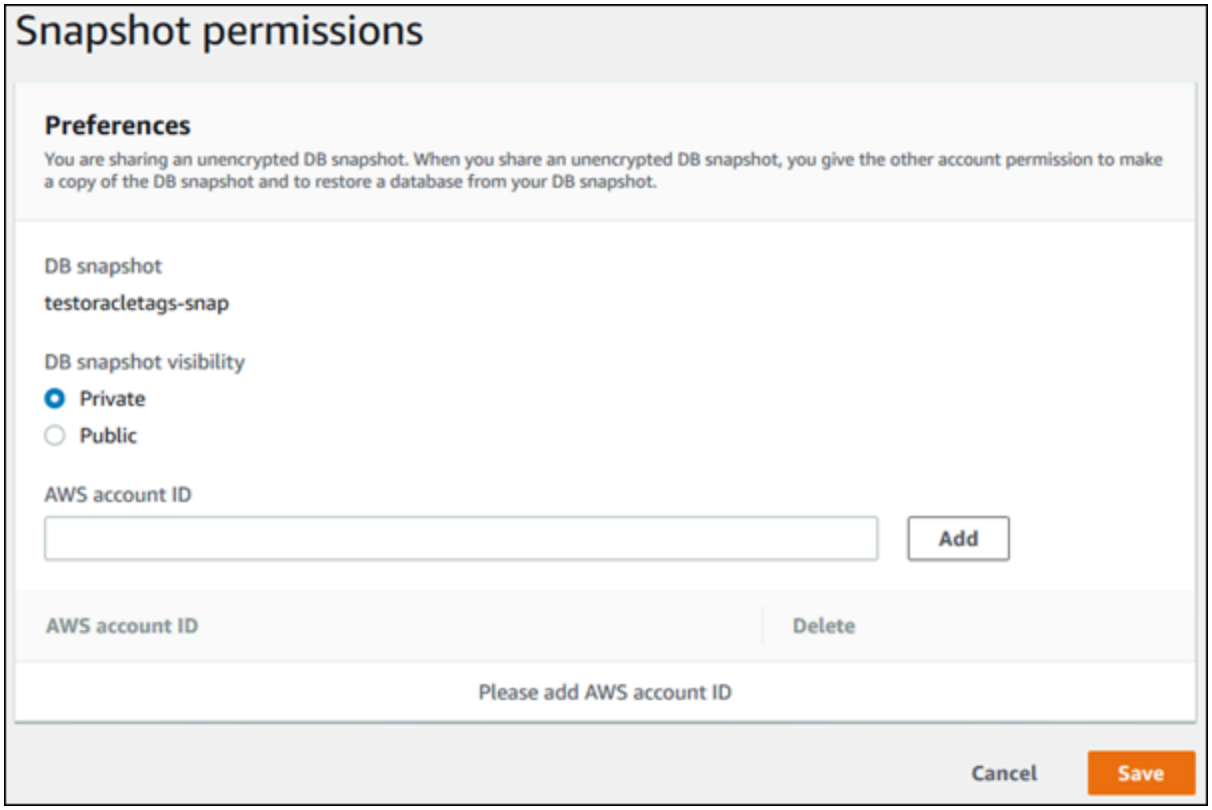
     > **Warning**
     >
     > [DB スナップショットの可視性] を [パブリック] に設定した場合、すべての AWS アカウントが手動 DB スナップショットから DB インスタンスを復元し、データにアクセスできるようになります。プライベート情報を含む手動 DB スナップショットは、[Public] として共有しないでください。
     > 詳細については、「Amazon RDS の公開スナップショットの共有」を参照してください。

   - 出典 DB クラスターが暗号化されている場合、暗号化されているスナップショットはパブリックとして共有できないため、[DB snapshot visibility] が [Private] に設定されます。

> **Note**
>
> デフォルトの AWS KMS key で暗号化されたスナップショットは共有できません。この問題を回避する方法については、「Amazon RDS の暗号化されたスナップショットの共有」を参照してください。

6. [AWS アカウント ID] では、手動スナップショットからの DB インスタンスの復元を許可するアカウントの AWS アカウント 識別子を入力してから、[追加] を選択します。この操作を繰り返して、AWS アカウント 識別子を最大 20 AWS アカウント まで含めます。

   アクセス権限が付与されたアカウントのリストに AWS アカウント 識別子を誤って追加した場合には、その AWS アカウント ID の右側にある [削除] を選択すれば、削除することができます。



7. 手動スナップショットの復元を許可するすべての AWS アカウント の識別子を追加した後、[保存] を選択して、変更を保存します。

AWS CLI

DB スナップショットを共有するには、aws rds modify-db-snapshot-attribute コマンドを使用します。--values-to-add のパラメータを使用して、手動スナップショットの復元が許可されている AWS アカウント のための ID リストを追加します。

Exampleスナップショットを 1 つのアカウントで共有する

次の例では、AWS アカウント 識別子 123456789012 が db7-snapshot という名前の DB スナップショットを復元できるようにします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier db7-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

Windows の場合:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier db7-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example複数のアカウントでスナップショットを共有する

次の例では、2 つの AWS アカウント 識別子 111122223333 および 444455556666 が manual-snapshot1 という名前の DB スナップショットを復元できるようにします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Windows の場合:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier manual-snapshot1 ^
```

```
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

スナップショットの復元が有効になっている AWS アカウント を一覧表示するには、`describe-db-snapshot-attributes` AWS CLI コマンドを使用します。

RDS API

Amazon RDS API を使用することで、手動 DB スナップショットを他の AWS アカウント と共有することもできます。そのためには、`ModifyDBSnapshotAttribute` オペレーションを呼び出します。`AttributeName` に `restore` を指定し、`ValuesToAdd` パラメータを使用して、手動 スナップショットの復元が許可されている AWS アカウント の ID のリストを追加します。

手動スナップショットをパブリックにして、すべての AWS アカウント による復元を可能にするには、値 `all` を使用します。ただし、すべての AWS アカウント には利用させたくないプライベート情報を含む手動スナップショットについては、値 `all` を追加しないように注意してください。また、暗号化されているスナップショットでは `all` を指定しないでください。そのようなスナップショットをパブリックにすることはできないためです。

スナップショットの復元が許可されているすべての AWS アカウント を一覧表示するには、`DescribeDBSnapshotAttributes` API 操作を使用します。

Amazon RDS の公開スナップショットの共有

暗号化されていない手動スナップショットをパブリックとして共有できます。これにより、このスナップショットをすべての AWS アカウント が使用できるようになります。スナップショットを公開として共有する場合には、公開スナップショットにプライベート情報が含まれないように注意してください。

スナップショットがパブリックに共有されると、スナップショットをコピーし、そこから DB インスタンスを作成するためのすべての AWS アカウント アクセス許可が付与されます。

他のアカウントが所有する公開スナップショットのバックアップストレージについては課金されません。課金されるのは、所有しているスナップショットに対してのみです。

公開スナップショットをコピーする場合は、そのコピーを所有します。スナップショットコピーのバックアップストレージに対しては課金されます。DB インスタンスを公開スナップショットから作成する場合、その DB インスタンスに対して課金されます。Amazon RDS の料金情報については、Amazon RDS の製品ページを参照してください。

削除できるのは、所有している公開スナップショットのみです。共有またはパブリックスナップショットを削除するには、そのスナップショットを所有する AWS アカウント にログインできることを確認してください。

他の AWS アカウント が所有するパブリックスナップショットの表示

Amazon RDS コンソールの [Snapshots] (スナップショット) ページにある [Public] (公開) タブの特定の AWS リージョンで、他のアカウントが所有する公開スナップショットを表示できます。(自分のアカウントが所有する) スナップショットは、このタブには表示されません。

公開スナップショットを表示するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[スナップショット] を選択します。
3. [Public (公開)] タブを選択します。

   公開スナップショットが表示されます。[Owner (所有者)] 列に、公開スナップショットを所有しているアカウントが表示されます。

   > **Note**
   >
   > ページ設定を変更する必要がある場合、[パブリックスナップショット] リストの歯車アイコンを選択して、この列を表示させます。

独自の公開スナップショットの表示

次の AWS CLI コマンド (Unix のみ) を使用して、特定の AWS リージョンの AWS アカウント が所有するパブリックスナップショットを表示することができます。

```
aws rds describe-db-snapshots --snapshot-type public --include-public |
 grep account_number
```

公開スナップショットがある場合、次の例のような出力が返されます。

```
"DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mysnapshot1",
"DBSnapshotArn": "arn:aws:rds:us-east-1:123456789012:snapshot:mysnapshot2",
```

> **Note**
>
> DBSnapshotIdentifier または SourceDBSnapshotIdentifier のエントリが重複する場合があります。

廃止された DB エンジンバージョンからのパブリックスナップショットの共有

廃止された DB エンジンバージョンからのパブリックスナップショットの復元またはコピーはサポートされていません。

RDS for Oracle エンジンと RDS for PostgreSQL DB エンジンは、DB スナップショットのエンジンバージョンの直接アップグレードをサポートしています。スナップショットをアップグレードして、再度パブリックに共有できます。詳細については次を参照してください:

- Oracle DB スナップショットのアップグレード
- PostgreSQL DB スナップショットエンジンのバージョンのアップグレード

その他の DB エンジンでは、以下の手順を実行して、サポート対象外の既存のパブリックスナップショットを復元したりコピーしたりできます。

1. スナップショットをプライベートとしてマークします。
2. スナップショットを復元します。
3. 復元した DB インスタンスを、サポートされているエンジンバージョンにアップグレードします。
4. 新しいスナップショットを作成します。
5. スナップショットをパブリックに再共有します。

Amazon RDS の暗号化されたスナップショットの共有

「Amazon RDS リソースの暗号化」で説明しているように、AES-256 暗号化アルゴリズムを使用して暗号化された「保存中」の DB スナップショットを共有できます。

以下の制限は、暗号化されたスナップショットの共有に適用されます。

- 暗号化されたスナップショットをパブリックとして共有することはできません。
- Transparent Data Encryption (TDE) を使用して暗号化されている Oracle または Microsoft SQL Server のスナップショットを共有することはできません。
- スナップショットを共有した AWS アカウント のデフォルト KMS キーを使用して暗号化されたスナップショットを共有することはできません。

  Amazon RDS の AWS KMS キー管理の詳細については、「AWS KMS key 管理」を参照してください。

デフォルトの KMS キーの問題を回避するには、以下のタスクを実行します。

1. カスタマーマネージドキーを作成し、そのキーへのアクセス権を付与する.
2. ソースアカウントからスナップショットをコピーして共有する.
3. ターゲットアカウントに共有したスナップショットをコピーします。.

カスタマーマネージドキーを作成し、そのキーへのアクセス権を付与する

まず、暗号化された DB スナップショットと同じ AWS リージョン にカスタム KMS キーを作成します。カスタマーマネージドキーの作成中に、別の AWS アカウント にそのキーへのアクセス権を付与します。

> **Note**
>
> キーポリシーがソースアカウントとターゲットアカウントへのアクセスを許可する場合、別の AWS アカウントの KMS キーを使用することもできます。

カスタマーマネージドキーを作成し、そのキーへのアクセス権を付与するには

1. ソース AWS アカウント から AWS マネジメントコンソール にサインインします。
2. AWS KMS コンソール (https://console.aws.amazon.com/kms) を開きます。
3. AWS リージョン を変更するには、ページの右上隅にあるリージョンセレクターを使用します。
4. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
5. [Create key] (キーの作成) を選択します。
6. [キーの設定] ページで、次の操作を行います。

a. [キータイプ] として、[対称] を選択します。
b. [キーの使用] で、[暗号化および復号化] を選択します。
c. [詳細オプション] を展開します。
d. [キーマテリアルオリジン] として、[KMS] を選択します。
e. [リージョン] で、[単一リージョンキー] を選択します。
f. [次へ] を選択します。

7. [ラベルを追加] ページで以下のように操作します。

a. [エイリアス] には、**share-snapshot** のように KMS キーの表示名を入力します。
b. (オプション) KMS キーの説明を入力します。
c. (オプション) KMS キーにタグを追加します。
d. [次へ] を選択します。

8. [キー管理アクセス許可の定義] ページで、[次へ] をクリックします。
9. [キーの使用アクセス許可の定義] ページで、次の操作を行います。

a. [その他の AWS アカウント] では、[別の AWS アカウント の追加] を選択します。
b. アクセスを許可する AWS アカウント の ID を入力します。

   複数の AWS アカウント にアクセス権を付与できます。

c. [次へ] を選択します。

10. KMS キーを確認し、[終了] を選択します。

ソースアカウントからスナップショットをコピーして共有する

次に、カスタマーマネージドキーを使用して、ソース DB スナップショットを新しいスナップショットにコピーします。次に、ターゲット AWS アカウント と共有します。

スナップショットをコピーして共有するには

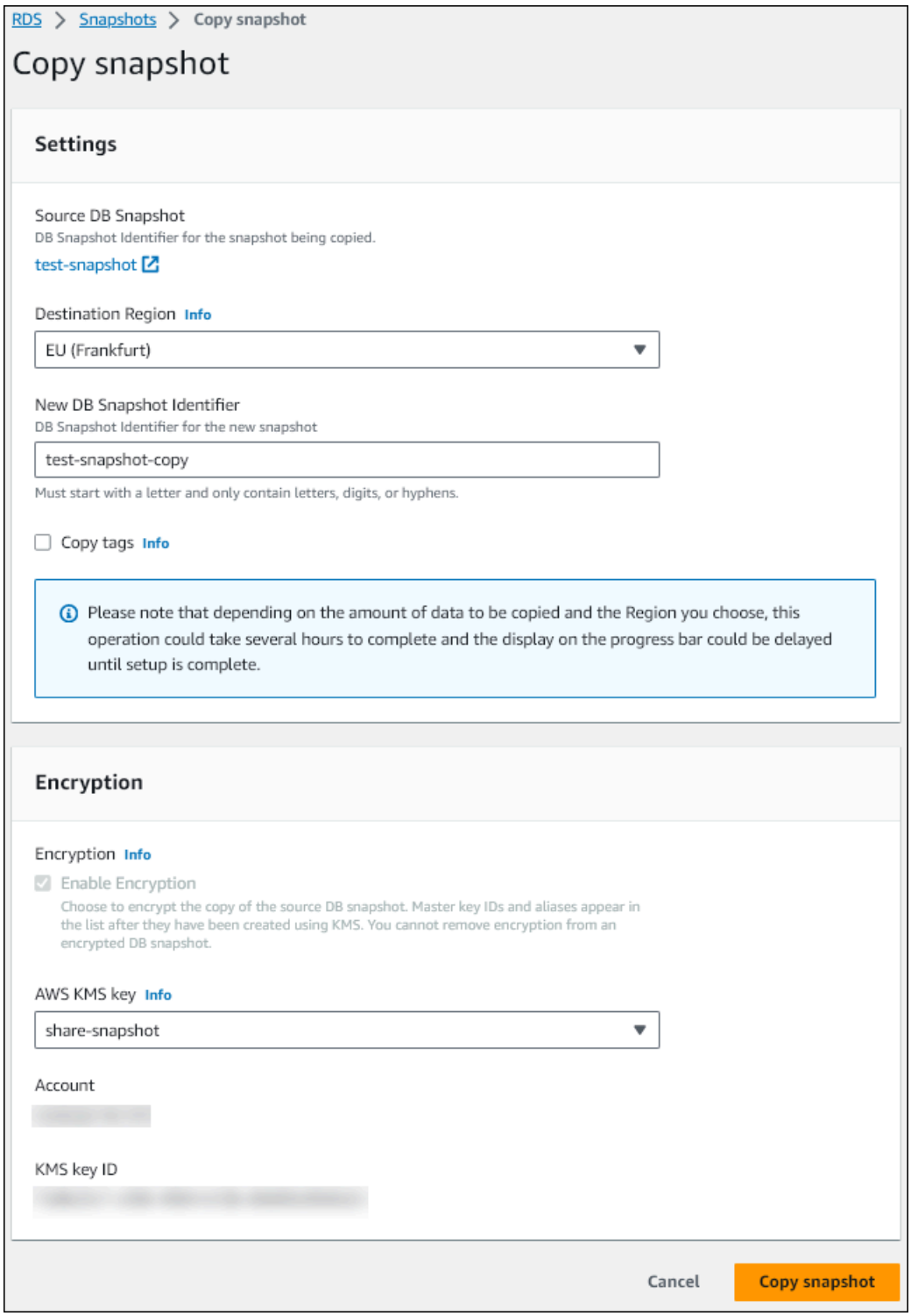
1. ソース AWS アカウント から AWS マネジメントコンソール にサインインします。
2. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
3. ナビゲーションペインで、[Snapshots] を選択してください。
4. コピーする DB スナップショットを選択します。
5. [アクション] で、[スナップショットをコピー] を選択します。

6. [スナップショットのコピー] ページで、次の操作を行います。

   a. [送信先リージョン] で、前の手順でカスタマーマネージドキーを作成した AWS リージョンを選択します。

   b. [新しい DB スナップショットの識別子] に、DB スナップショットのコピーの名前を入力します。

   c. [AWS KMS key] で、作成したカスタマーマネージドキーを選択します。



d. [スナップショットをコピー] を選択します。

7. スナップショットのコピーが使用可能になったら、それを選択します。
8. [アクション] で、[スナップショットを共有] を選択します。
9. [スナップショットのアクセス許可] ページで、次の操作を行います。

a. スナップショットのコピーを共有する [AWS アカウント ID] を入力し、[追加] を選択します。
b. [保存] を選択します。

スナップショットが共有されます。

ターゲットアカウントに共有したスナップショットをコピーします。

これで、ターゲット AWS アカウント で共有スナップショットをコピーできます。

共有したスナップショットをコピーするには

1. ターゲット AWS アカウント から AWS マネジメントコンソール にサインインします。
2. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
3. ナビゲーションペインで、[Snapshots] を選択してください。
4. [自分と共有] タブを選択します。
5. 共有スナップショットを選択します。
6. [アクション] で、[スナップショットをコピー] を選択します。
7. 前の手順のようにスナップショットをコピーするための設定を選択しますが、ターゲットアカウントに属する AWS KMS key を使用します。

   [スナップショットをコピー] を選択します。

Amazon RDS のスナップショット共有の停止

DB スナップショットの共有を停止するには、ターゲット AWS アカウント からアクセス許可を削除します。

コンソール

AWS アカウント との手動 DB スナップショットの共有を停止するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。

3. 共有を停止する手動スナップショットを選択します。
4. [Actions] (アクション) を選択してから、[Share Snapshot] (スナップショットの共有) を選択します。
5. AWS アカウント のアクセス許可を削除するには、アクセス権限が付与されたアカウントのリストからそのアカウントの AWS アカウント識別子を選択し、[削除] を選択します。
6. [保存] を選択して変更を保存します。

CLI

リストから AWS アカウント の識別子を削除するには、`--values-to-remove` のパラメータを使用します。

Exampleスナップショット共有を停止する

次の例は、AWS アカウント ID 444455556666 がスナップショットを復元できないようにします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot-attribute \
--db-snapshot-identifier manual-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Windows の場合:

```
aws rds modify-db-snapshot-attribute ^
--db-snapshot-identifier manual-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

RDS API

AWS アカウント でのアクセス許可の共有を削除するには、`AttributeName` を `restore` に設定し、かつ `ValuesToRemove` パラメータを含めて、`ModifyDBSnapshotAttribute` オペレーションを使用します。手動 スナップショットをプライベートとしてマークするには、`all` 属性の値リストから値 `restore` を削除します。

Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート

DB スナップショットデータを Amazon S3 バケットにエクスポートできます。エクスポートプロセスはバックグラウンドで実行されるため、アクティブな DB クラスターのパフォーマンスには影響しません。

DB スナップショットをエクスポートすると、Amazon RDS はスナップショットからデータを抽出して Amazon S3 バケットに保存します。データは Apache Parquet 形式で一貫して圧縮され、保存されます。

すべてのタイプの DB スナップショット (手動スナップショット、自動システムスナップショット、AWS Backup サービスで作成されたスナップショットなど) をエクスポートできます。デフォルトでは、スナップショット内のすべてのデータがエクスポートされます。ただし、特定のデータベース、スキーマ、またはテーブルのセットをエクスポートすることもできます。

データをエクスポートすると、Amazon Athena や Amazon Redshift Spectrum などのツールを使用して、エクスポートしたデータを直接分析できます。Athena を使用して Parquet データを読み取る方法の詳細については、Amazon Athena ユーザーガイド の Parquet SerDe を参照してください。Redshift Spectrum を使用して Parquet データを読み取る方法の詳細については、Amazon Redshift Database デベロッパーガイドの「列指向データ形式からの COPY」を参照してください。

> **Warning**
>
> エクスポートされたスナップショットデータを S3 から新しい DB インスタンスに復元したり、S3 から既存の DB インスタンスにスナップショットデータをインポートしたりすることはできません。ただし、分析のために Amazon Athena または Redshift Spectrum を使用してデータを処理することはできます。さらに、AWS Glue を使用してデータを変換し、AWS DMS やカスタムスクリプトなどのツールを使用して Amazon RDS にインポートできます。

DB スナップショットの Amazon S3 へのエクスポートの詳細については、以下のトピックを参照してください。

トピック

- Amazon RDS のスナップショットエクスポートのモニタリング
- Amazon RDS のスナップショットのエクスポートタスクのキャンセル

- Amazon RDS の Amazon S3 エクスポートタスクの障害メッセージ
- RDS for PostgreSQL のアクセス許可エラーのトラブルシューティング
- Amazon RDS の Amazon S3 へのエクスポートのファイル命名規則
- Amazon RDS の Amazon S3 バケットにエクスポートする際のデータ変換

スナップショットデータのエクスポートの概要

次のプロセスを使用して、DB スナップショットデータを Amazon S3 バケットにエクスポートします。詳細については、次のセクションを参照してください。

1. エクスポートするスナップショットを特定します。

   既存の自動スナップショットまたは手動スナップショットを使用するか、DB インスタンスまたはマルチ AZ DB クラスターの手動スナップショットを作成します。

2. Amazon S3 バケットへのアクセスを設定します。

   バケットとは、Amazon S3 オブジェクトまたはファイルのコンテナです。バケットにアクセスするための情報を指定するには、次のステップに従います。

   a. スナップショットのエクスポート先の S3 バケットを特定します。S3 バケットはスナップショット同じ AWS リージョンに存在する必要があります。詳細については、「エクスポート先の Amazon S3 バケットの特定」を参照してください。

   b. スナップショットエクスポートタスクに対して S3 バケットへのアクセスを許可する AWS Identity and Access Management (IAM) ロールを作成します。詳細については、「IAM ロールを使用した Amazon S3 バケットへのアクセスの提供」を参照してください。

3. サーバー側の暗号化用の対称暗号化 AWS KMS key を作成します。KMS キーは、エクスポートデータを S3 に書き込むときに AWS KMS サーバー側の暗号化を設定するために、スナップショットエクスポートタスクによって使用されます。

   KMS キーポリシーには、`kms:CreateGrant` と `kms:DescribeKey` の両方のアクセス許可を含める必要があります。Amazon RDS での KMS キーの使用方法の詳細については、「AWS KMS key 管理」を参照してください。

   KMS キーポリシーに deny ステートメントがある場合は、必ず AWS サービスプリンシパル `export.rds.amazonaws.com` を明示的に除外してください。

AWSアカウント内で KMS キーを使用することも、クロスアカウント KMS キーを使用することもできます。詳細については、「AWS KMS keyAmazon S3 エクスポートの暗号化にクロスアカウントを使用する」を参照してください。

4. コンソールまたは `start-export-task` CLI コマンドを使用して、スナップショットを Amazon S3 にエクスポートします。詳細については、「Amazon S3 バケットへの DB スナップショットのエクスポート」を参照してください。
5. Amazon S3 バケット内のエクスポートされたデータにアクセスするには、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのアップロード、ダウンロード、管理」を参照してください。

Amazon S3 バケットへのアクセスを設定する

DB スナップショットデータを Amazon S3 ファイルにエクスポートするには、まず Amazon S3 バケットにアクセスするためのアクセス許可をスナップショットに付与します。次に、Amazon S3 バケットへの書き込みを Amazon RDS サービスに許可するための IAM ロールを作成します。

トピック

- エクスポート先の Amazon S3 バケットの特定
- IAM ロールを使用した Amazon S3 バケットへのアクセスの提供
- クロスアカウント Amazon S3 バケットを使用する
- AWS KMS keyAmazon S3 エクスポートの暗号化にクロスアカウントを使用する

エクスポート先の Amazon S3 バケットの特定

DB スナップショットをエクスポートする先の Amazon S3 バケットを特定します。既存の S3 バケットを使用するか、新しい S3 バケットを作成します。

> **Note**
>
> エクスポート先の S3 バケットは、スナップショットと同じ AWS リージョンに存在している必要があります。

Amazon S3 バケットの操作の詳細については、Amazon Simple Storage Service ユーザーガイドで次のトピックを参照してください。

- S3 バケットのプロパティを表示する方法
- Amazon S3 バケットのデフォルト暗号化を有効にする方法
- S3 バケットを作成する方法

IAM ロールを使用した Amazon S3 バケットへのアクセスの提供

DB スナップショットデータを Amazon S3 にエクスポートする前に、スナップショットエクスポートタスクに対して Amazon S3 バケットへの書き込みアクセス権限を付与します。

このアクセス権限を付与するには、バケットへのアクセスを可能にする IAM ポリシーを作成し、次に IAM ロールを作成して、このロールにポリシーをアタッチします。後で IAM ロールをスナップショットエクスポートタスクに割り当てます。

Amazon S3 のその他の管理ツールの詳細については、「Amazon S3 ユーザーガイド」の「Amazon S3 でのアクセスコントロール」を参照してください。

> **Important**
>
> AWS マネジメントコンソール を使用してスナップショットをエクスポートしようとする場合は、スナップショットをエクスポートするときに IAM ポリシーとロールを自動的に作成するように選択できます。手順については、「Amazon S3 バケットへの DB スナップショットのエクスポート」を参照してください。

Amazon S3 へのアクセスを DB スナップショットタスクに許可するには

1. IAM ポリシーを作成します。このポリシーでバケットおよびオブジェクトへのアクセス許可を提供することにより、スナップショットエクスポートタスクから Amazon S3 にアクセスできるようにします。

   ポリシーには、Amazon RDS から S3 バケットへのファイル転送を許可するために、以下の必須アクションを含めます。

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

ポリシーには、S3 バケットとバケット内のオブジェクトを識別するために、以下のリソースを含めます。次のリソースのリストは、Amazon S3 にアクセスするための Amazon リソースネーム (ARN) 形式を示しています。

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Amazon RDS の IAM ポリシーの作成の詳細については、「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。IAM ユーザーガイドの「チュートリアル: はじめてのカスタマー管理ポリシーの作成とアタッチ」も参照してください。

以下の AWS CLI コマンドでは、これらのオプションを指定して、ExportPolicy という名前の IAM ポリシーを作成します。*amzn-s3-demo-bucket* という名前のバケットへのアクセスを許可します。

> **Note**
>
> ポリシーを作成したら、ポリシーの ARN を書き留めます。ポリシーを IAM ロールにアタッチする場合、後続のステップで ARN が必要です。

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
```

```
        }
    ]
}'
```

2. IAM ロールを作成することで、お客様に代わって Amazon RDS が IAM ロールを引き受け、Amazon S3 バケットにアクセスできるようにします。詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

   以下の例は、AWS CLI コマンドを使用して、`rds-s3-export-role` という名前のロールを作成する例を示しています。

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. 作成した IAM ポリシーを、作成した IAM ロールにアタッチします。

   次の AWS CLI コマンドでは、先ほど作成したポリシーを `rds-s3-export-role` という名前のロールにアタッチします。*`your-policy-arn`* は、以前のステップで書き留めたポリシー ARN に置き換えます。

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

クロスアカウント Amazon S3 バケットを使用する

Amazon S3 バケットはAWSアカウント全体で使用できます。クロスアカウントバケットを使用するには、S3 エクスポートに使用している IAM ロールへのアクセスを許可するバケットポリシーを追加してください。詳細については、「例 2: クロスアカウントバケット権限を付与するバケット所有者」を参照してください。

次の例のように、バケットポリシーをバケットに添付します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

AWS KMS keyAmazon S3 エクスポートの暗号化にクロスアカウントを使用する

AWS KMS keyAmazon S3 エクスポートの暗号化にクロスアカウントを使用できます。まずローカルアカウントにキーポリシーを追加し、次に外部アカウントに IAM ポリシーを追加してください。詳細については、「その他のアカウントのユーザーに KMS キーの使用を許可する」を参照してください。

クロスアカウント KMS キーを使用するには

1. ローカルアカウントにキーポリシーを追加します。

   次の例は、外部アカウント 444455556666 のExampleRoleとExampleUserに、ローカルアカウント 123456789012 のアクセス許可を付与します。

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

2. 外部アカウントに IAM ポリシーを追加する

以下の IAM ポリシーの例では、プリンシパルがアカウント 123456789012 の KMS キーを暗号化オペレーションに使用することを許可します。アカウント 444455556666 のExampleRoleとExampleUserにこの許可を与えるには、そのアカウントにポリシーをアタッチします。

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
}
```

Amazon S3 バケットへの DB スナップショットのエクスポート

1 つの AWS アカウント につき、最大 5 つの DB スナップショットエクスポートタスクを同時に実行できます。

> **Note**
>
> データベースのタイプとサイズによっては、RDS スナップショットのエクスポートに時間がかかることがあります。エクスポートタスクでは、Amazon S3 にデータを抽出する前に、データベース全体を復元およびスケールします。このフェーズでのタスクの進捗状況は、[起動中] と表示されます。タスクが S3 へのデータのエクスポートに切り替わると、進捗状況は [進行中] と表示されます。
>
> エクスポートが完了するまでにかかる時間は、データベースに格納されているデータによって異なります。例えば、数値のプライマリキーまたはインデックス列が適切に配信されているテーブルは、最も速くエクスポートされます。パーティション化に適した列が含まれていないテーブルや、文字列ベースの列に 1 つのインデックスしかないテーブルは処理に時間がかかります。エクスポートに低速のシングルスレッドプロセスを使用するため、このような長いエクスポート時間となります。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB スナップショットを Amazon S3 にエクスポートできます。DB スナップショットをクロスアカウント Amazon S3 バケットにエクスポートするには、AWS CLI または RDS API を使用します。

Lambda 関数を使用してスナップショットをエクスポートする場合は、Lambda 関数ポリシーに `kms:DescribeKey` アクションを追加します。詳細については、「AWS Lambda のアクセス許可」を参照してください。

コンソール

[Amazon S3 へのエクスポート] コンソールオプションは、Amazon S3 にエクスポートできるスナップショットに対してのみ表示されます。スナップショットは、次の理由により、エクスポートに使用できない場合があります。

- DB エンジンが S3 エクスポートでサポートされていない。
- DB エンジンのバージョンが S3 エクスポートでサポートされていません。

- スナップショットを作成した AWS リージョンで S3 エクスポートがサポートされていない。

DB スナップショットをエクスポートするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[スナップショット] を選択します。
3. タブから、エクスポートするスナップショットのタイプを選択します。
4. スナップショットのリストで、エクスポートするスナップショットを選択します。
5. [アクション] で、[Amazon S3 にエクスポート] を選択します。

   [EAmazon S3 にエクスポート] ウィンドウが表示されます。
6. [エクスポート識別子] に、エクスポートタスクを識別する名前を入力します。この値は、S3 バケットで作成されるファイルの名前としても使用されます。
7. エクスポートするデータを選択します。

   - [すべて] を選択すると、スナップショット内のすべてのデータがエクスポートされます。
   - [部分的] を選択すると、スナップショットの特定部分がエクスポートされます。スナップショットのどの部分をエクスポートするかを特定するには、[識別子] に 1 つ以上のデータベース、スキーマ、またはテーブルをスペースで区切って入力します。

     次の形式を使用します。

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.scheman]
     [.tablen]
     ```

     次に例を示します。

     ```
     mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
      mydatabase2.myschema2.mytable2
     ```
8. [S3 バケット] で、エクスポート先のバケットを選択します。

   エクスポートされたデータを S3 バケット内のフォルダパスに割り当てるには、[S3 プレフィックス] にオプションのパスを入力します。
9. IAM ロール の場合は、選択した S3 バケットへの書き込みアクセスを許可するロールを選択するか、新しいロールを作成します。

- 「IAM ロールを使用した Amazon S3 バケットへのアクセスの提供」のステップに従ってロールを作成した場合は、そのロールを選択します。
- 選択した S3 バケットへの書き込みアクセス権を付与するロールを作成しなかった場合は、[Create a new role] (新しいロールの作成) を選択して、ロールを自動的に作成します。次に、[IAM role name] (IAM ロール名) にロールの名前を入力します。

> **Note**
>
> アカウントでロールマネージャーが有効になっている場合、Amazon RDS は自動的にロールをアタッチします。また、ここで説明するロールオプションは [カスタマイズ] オプションに置き換えられます。別のロールを使用するには、[カスタマイズ] を選択します。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

10. [AWS KMS key] で、エクスポートされたデータの暗号化に使用するキーの ARN を入力します。
11. [Amazon S3 にエクスポート] を選択します。

AWS CLI

AWS CLI を使用して DB スナップショットを Amazon S3 にエクスポートするには、以下の必須オプションを指定して start-export-task コマンドを使用します。

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

以下の例では、スナップショットエクスポートタスクは *`my_snapshot_export`* と名前が付けられ、スナップショットを *`amzn-s3-demo-bucket`* という名前の S3 バケットにエクスポートします。

Example

Linux、macOS、Unix の場合:

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Windows の場合:

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

サンプル出力を次に示します。

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "my-export-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

スナップショットのエクスポート先である S3 バケット内のフォルダパスを指定するには、start-export-task コマンドに `--s3-prefix` オプションを含めます。

RDS API

Amazon RDS API を使用して DB スナップショットを Amazon S3 にエクスポートするには、以下の必須パラメータを指定して StartExportTask オペレーションを使用します。

- ExportTaskIdentifier
- SourceArn
- S3BucketName
- IamRoleArn
- KmsKeyId

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョンと AWS リージョン によって異なります。S3 へのスナップショットのエクスポートによるバージョンとリージョンの可用性の詳細については、「Amazon RDS の S3 へのスナップショットデータエクスポートでサポートされているリージョンと DB エンジン」を参照してください。

制限事項

DB スナップショットデータの Amazon S3 へのエクスポートには、次の制限があります。

- 同じ DB クラスタースナップショットに対して複数のエクスポートタスクを同時に実行することはできません。これは、フルエクスポートと部分エクスポートの両方に当てはまります。
- 磁気ストレージを使用するデータベースからのスナップショットのエクスポートはサポートされていません。
- S3 へのエクスポートでは、コロン (:) を含む S3 プレフィックスをサポートしていません。
- S3 ファイルパスの次の文字は、エクスポート時にアンダースコア (_) に変換されます。

```
\ ` " (space)
```

- データベース、スキーマ、またはテーブルの名前に次の文字以外の文字が含まれている場合、部分的なエクスポートはサポートされません。ただし、DB スナップショット全体をエクスポートすることはできます。
  - ラテン文字 (A-Z)
  - 数字 (0-9)
  - ドル記号 ($)
  - 下線 (_)
- データベーステーブルの列名では、一部の文字と空白文字の使用はサポートされていません。列名に次の文字が含まれるテーブルは、エクスポート時にスキップされます。

```
, ; { } ( ) \n \t = (space)
```

- 名前にスラッシュ (/) が含まれるテーブルは、エクスポート時にスキップされます。
- RDS for PostgreSQL の一時テーブルとログに記録されていないテーブルは、エクスポート中にスキップされます。
- データに BLOB や CLOB などの大きいオブジェクト (500 MB に近いか、それ以上) が含まれている場合、エクスポートは失敗します。
- テーブルに、2 GB に近いか、それ以上のサイズの大きな行が含まれている場合、そのテーブルはエクスポート時にスキップされます。
- 部分エクスポートの場合、`ExportOnly` リストの最大サイズは 200 KB です。
- エクスポートタスクごとに一意の名前を使用することを強くお勧めします。一意のタスク名を使用しない場合、次のエラーメッセージが表示されることがあります。

  ExportTaskAlreadyExistsFault: StartExportTask オペレーションを呼び出すときにエラー (ExportTaskAlreadyExists) が発生しました。ID *xxxxx* のエクスポートタスクは既に存在します。
- データを S3 にエクスポートしている間はスナップショットを削除できますが、エクスポートタスクが完了するまで、そのスナップショットのストレージコストは引き続き課金されます。
- エクスポートされたスナップショットデータを S3 から新しい DB インスタンスに復元したり、S3 から既存の DB インスタンスにスナップショットデータをインポートしたりすることはできません。
- 1 つの AWS アカウント につき、最大 5 つの DB スナップショットエクスポートタスクを同時に実行できます。
- DB スナップショットをクロスアカウント Amazon S3 バケットにエクスポートするには、AWS CLI または RDS API を使用する必要があります。
- Amazon RDS のエクスポートタスクが完了したら、同じ DB スナップショットから別のエクスポートタスクが開始されるまで少し待つことが必要な場合があります。
- ビューまたはマテリアライズドビューをエクスポートすることはできません。
- RDS から S3 へのエクスポートでは、GuardDuty Malware Protection for S3 のタグベースのアクセスコントロールがサポートされません。

Amazon RDS のスナップショットエクスポートのモニタリング

DB スナップショットのエクスポートは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用してモニタリングできます。

コンソール

DB スナップショットのエクスポートをモニタリングするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. スナップショットのエクスポート一覧を表示するには、[Exports in Amazon S3] タブを選択します。
4. 特定のスナップショットのエクスポートに関する情報を表示するには、エクスポートタスクを選択します。

AWS CLI

AWS CLI を使用して DB スナップショットのエクスポートをモニタリングするには、describe-export-tasks コマンドを使用します。

次の例は、すべてのスナップショットのエクスポートに関する最新情報を表示する方法を示しています。

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
```

```
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket edgcuc-export isn't located in the current
 AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

特定のスナップショットのエクスポートに関する情報を表示するには、--export-task-identifier コマンドに describe-export-tasks オプションを含めます。出力をフィルタリ

ングするには、`--Filters` オプションを含めます。その他のオプションについては、 describe-export-tasks コマンドを参照してください。

RDS API

Amazon RDS API を使用して DB スナップショットのエクスポートに関する情報を表示するには、DescribeExportTasks オペレーションを使用します。

エクスポートワークフローの完了を追跡したり、別のワークフローを開始したりするには、Amazon Simple Notification Service トピックをサブスクライブします。Amazon SNS の詳細については、「Amazon RDS イベント通知の操作」を参照してください。

Amazon RDS のスナップショットのエクスポートタスクのキャンセル

DB スナップショットのエクスポートタスクをキャンセルするには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用できます。

> **Note**
>
> スナップショットのエクスポートタスクをキャンセルしても、Amazon S3 にエクスポート済みのデータは削除されません。コンソールを使用してデータを削除する方法については、「S3 バケットからオブジェクトを削除する方法」を参照してください。CLI を使用してデータを削除するには、delete-object コマンドを使用します。

コンソール

スナップショットのエクスポートタスクをキャンセルするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[スナップショット] を選択します。
3. [Exports in Amazon S3] タブを選択します。
4. キャンセルするスナップショットのエクスポートタスクを選択します。
5. [キャンセル] を選択します。
6. 確認ページで [エクスポートタスクをキャンセル] を選択します。

AWS CLI

AWS CLI を使用してスナップショットのエクスポートタスクをキャンセルするには、cancel-export-task コマンドを使用します。このコマンドには、--export-task-identifier オプションが必要です。

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
}
```

RDS API

Amazon RDS API を使用してスナップショットのエクスポートタスクをキャンセルするには、ExportTaskIdentifier パラメータを指定して CancelExportTask オペレーションを使用します。

Amazon RDS の Amazon S3 エクスポートタスクの障害メッセージ

次の表では、Amazon S3 エクスポートタスクに障害が発生したときに返されるメッセージについて説明します。

障害メッセージ	説明
不明な内部エラーが発生しました。	不明なエラー、例外、または障害により、タスクが失敗しました。
エクスポートタスクのメタデータを S3 バケット 「バケット名」 に書き込む	不明なエラー、例外、または障害により、タスクが失敗しました。

障害メッセージ	説明
際に、不明な内部エラーが発生しました。	
RDS エクスポートは、IAM ロール「ロール ARN」を引き受けることができないため、エクスポートタスクのメタデータを書き込めませんでした。	エクスポートタスクは IAM ロールを引き受け、S3 バケットへのメタデータの書き込みが許可されているかどうかを検証します。タスクが IAM ロールを引き受けられない場合は失敗します。
RDS エクスポートで、KMS キー「キー ID」 を持つ IAM ロール 「ロール ARN」 を使用した S3 バケット 「バケット名」へのエクスポートタスクのメタデータの書き込みに失敗しました。エラーコード:「エラーコード」	1 つ以上のアクセス許可が不足しているため、エクスポートタスクが S3 バケットにアクセスできません。この障害メッセージは、次のいずれかのエラーコードを受信したときにレイズされます。 • エラーコードAccessDenied のAWSSecurityTokenServiceException • エラーコードNoSuchBucket 、AccessDenied 、KMS.KMSInvalidStateException 、403 Forbidden 、またはKMS.DisabledException のAmazonS3Exception これらのエラーコードは、IAM ロール、S3 バケット、または KMS キーの設定が間違っていることを示しています。
IAM ロール [ロール ARN]には、S3 バケット [バケット名] で [S3 アクション] を呼び出す権限がありません。許可を確認してエクスポートを再試行してください。	IAM ポリシーの設定が間違っています。S3 バケットに対する特定の S3 アクションのためのアクセス許可がないため、エクスポートタスクが失敗します。
KMS キーチェックに失敗しました。KMS キーの認証情報をチェックして、再試行してください。	KMS キーの認証情報のチェックに失敗しました。

障害メッセージ	説明
S3 の認証情報のチェックに失敗しました。S3 バケットと IAM ポリシーに対するアクセス許可をチェックします。	S3 の認証情報のチェックに失敗しました。
S3 バケット「バケット名」が無効です。どちらかが現在のAWSリージョンにないか、存在しません。S3 バケット名を確認して、エクスポートを再試行してください。	S3 バケットが無効です。
S3 バケット 「バケット名」が現在のAWSリージョンにありません。S3 バケット名を確認して、エクスポートを再試行してください。	S3 バケットが間違ったAWSリージョンにあります。

RDS for PostgreSQL のアクセス許可エラーのトラブルシューティング

PostgreSQL データベースを Amazon S3 にエクスポートするときに、特定のテーブルがスキップされたことを示す PERMISSIONS_DO_NOT_EXIST エラーが表示される場合があります。このエラーは、通常、データベースの作成時に指定したスーパーユーザーが、これらのテーブルにアクセスする許可を持っていない場合に発生します。

このエラーを修正するには、次のコマンドを実行します。

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

スーパーユーザー権限の詳細については、「マスターユーザーアカウント権限」を参照してください。

Amazon RDS の Amazon S3 へのエクスポートのファイル命名規則

特定のテーブルのエクスポートされたデータは、*base_prefix*/*files* の形式で保存されます。そのベースプレフィックスは次のとおりです。

```
export_identifier/database_name/schema_name.table_name/
```

例:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

ファイルを名付ける方法には、次の 2 つの規則があります。

- 現在の規則:

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  バッチインデックスは、テーブルから読み込まれたデータのバッチを表すシーケンス番号です。テーブルを小さなチャンクに分割し、並列でエクスポートできない場合は、複数のバッチインデックスになります。テーブルが複数のテーブルにパーティション化されている場合にも同じことが起こります。メインテーブルのテーブルパーティションごとに 1 つずつ、複数のバッチインデックスがあります。

  テーブルを小さなチャンクに分割し、並列で読み取ることができる場合は、バッチインデックス 1 フォルダのみになります。

  バッチインデックスフォルダ内には、テーブルのデータを含む 1 つまたは複数の Parquet ファイルがあります。Parquet ファイル名のプレフィックスは part-*partition_index* です。テーブルがパーティション化されている場合、パーティションインデックス 00000 で始まる複数のファイルになります。

  パーティションインデックスシーケンスにギャップが生じる可能性があります。これは、各パーティションがテーブル内の範囲クエリから取得されるためです。そのパーティションの範囲内にデータがない場合、そのシーケンス番号はスキップされます。

  例えば、id 列がテーブルのプライマリキーで、その最小値と最大値が 100 と 1000 であるとします。このテーブルを 9 つのパーティションでエクスポートしようとすると、次のような並列クエリで読み取られます。

  ```
  SELECT * FROM table WHERE id <= 100 AND id < 200
  SELECT * FROM table WHERE id <= 200 AND id < 300
  ```

  これにより、part-00000-*random_uuid*.gz.parquet から part-00008-*random_uuid*.gz.parquet までの 9 つのファイルが生成されます。ただし、200 と 350 の間に ID を持つ行がない場合、完了したパーティションの 1 つが空になり、それに対す

るファイルも作成されません。前の例では、part-00001-*random_uuid*.gz.parquet は作成されません。

- 以前の規則:

```
part-partition_index-random_uuid.format-based_extension
```

これは現在の規則と同じですが、例えば、*batch_index* プレフィックスは除きます。

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

ファイルの命名規則は変更されることがあります。したがって、ターゲットテーブルを読み込む場合は、テーブルのベースプレフィックス内のすべてを読み込むことをお勧めします。

Amazon RDS の Amazon S3 バケットにエクスポートする際のデータ変換

DB スナップショットを Amazon S3 バケットにエクスポートすると、Amazon RDS はデータを Parquet 形式に変換してエクスポートし、保存します。Parquet の詳細については、Apache Parquet のウェブサイトを参照してください。

Parquet は、すべてのデータを次のプリミティブ型の 1 つとして格納します。

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY - バイナリとも呼ばれる可変長のバイト配列
- FIXED_LEN_BYTE_ARRAY - 値が一定のサイズを持つ場合に使用される固定長のバイト配列

Parquet のデータ型はほとんど存在せず、この形式の読み書きに伴う複雑さが軽減されるようになっています。Parquet は、プリミティブ型を拡張するための論理的な型を提供します。論理的な型は、LogicalType メタデータフィールドにデータを持つ注釈として実装されます。論理的な型の注釈は、プリミティブ型の解釈方法を示します。

STRING 論理的な型が BYTE_ARRAY 型に注釈を付けた場合は、このバイト配列を UTF-8 でエンコードされた文字列として解釈する必要があることを示します。エクスポートタスクが完了すると、Amazon RDS は文字列変換が発生したかどうかを通知します。エクスポートされた基になるデータは、常に送信元データと同じです。ただし、UTF-8 のエンコーディングに伴う差異により、Athena などのツールで読み取ると、一部の文字はソースと異なるように表示される場合があります。

詳細については、Parquet ドキュメントの「Parquet Logical Type Definitions」を参照してください。

トピック

- MySQL およびMariaDB データ型の Parquet へのマッピング
- PostgreSQL データ型の Parquet へのマッピング

MySQL およびMariaDB データ型の Parquet へのマッピング

次の表は、データが変換されて Amazon S3 にエクスポートされる際の MySQL および MariaDB データ型から Parquet データ型へのマッピングを示しています。

出典データ型	Parquet プリミティブ型	論理的な型の注釈	変換に関するメモ
数値データ型			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN_BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet は符号付き型のみをサポートしているため、マッピングは BIGINT_UNSIGNED 型を格納するために追加のバイト (8 プラス 1) を必要とします。
BIT	BYTE_ARRAY		

出典データ型	Parquet プリミティブ型	論理的な型の注釈	変換に関するメモ
DECIMAL	INT32	DECIMAL(p,s)	出典値が 2^{31} 未満の場合は、INT32 として格納されます。
	INT64	DECIMAL(p,s)	出典値が 2^{31} 以上で 2^{63} 未満の場合は、INT64 として格納されます。
	FIXED_LEN_BYTE_ARRAY(N)	DECIMAL(p,s)	出典値が 2^{63} 以上の場合は、FIXED_LEN_BYTE_ARRAY(N) として格納されます。
	BYTE_ARRAY	STRING	Parquet は、38 を超える 10 進精度をサポートしていません。10 進値は、BYTE_ARRAY 型の文字列に変換され、UTF8 としてエンコードされます。
DOUBLE	DOUBLE		
FLOAT	DOUBLE		
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		

出典データ型	Parquet プリミティブ型	論理的な型の注釈	変換に関するメモ
NUMERIC	INT32	DECIMAL(p,s)	出典値が 2^{31} 未満の場合は、INT32 として格納されます。
	INT64	DECIMAL(p,s)	出典値が 2^{31} 以上で 2^{63} 未満の場合は、INT64 として格納されます。
	FIXED_LEN_ARRAY(N)	DECIMAL(p,s)	出典値が 2^{63} 以上の場合は、FIXED_LEN_BYTE_ARRAY(N) として格納されます。
	BYTE_ARRAY	STRING	Parquet は、38 を超える数値精度をサポートしていません。この数値は、BYTE_ARRAY 型の文字列に変換され、UTF8 としてエンコードされます。
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
文字列データ型			
BINARY	BYTE_ARRAY		

出典データ型	Parquet プリミティブ型	論理的な型の注釈	変換に関するメモ
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
日付と時刻のデータ型			
DATE	BYTE_ARRAY	STRING	日付は BYTE_ARRAY 型の文字列に変換され、UTF8 としてエンコードされます。

出典データ型	Parquet プリミティブ型	論理的な型の注釈	変換に関するメモ
DATETIME	INT64	TIMESTAMP_MICROS	
TIME	BYTE_ARRAY	STRING	TIME 型は BYTE_ARRAY の文字列に変換され、UTF8 としてエンコードされます。
TIMESTAMP	INT64	TIMESTAMP_MICROS	
YEAR	INT32		
ジオメトリデータ型			
GEOMETRY	BYTE_ARRAY		
GEOMETRYCOLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		
JSON データ型			
JSON	BYTE_ARRAY	STRING	

PostgreSQL データ型の Parquet へのマッピング

次の表は、データが変換されて Amazon S3 にエクスポートされる際の PostgreSQL データ型から Parquet データ型へのマッピングを示しています。

PostgreSQL のデータ型	Parquet プリミティブ型	論理的な型の注釈	マッピングに関するメモ
数値データ型			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	DECIMAL 型は BYTE_ARRAY 型の文字列に変換され、UTF8 としてエンコードされます。 この変換は、データ精度や、数値ではないデータ値 (NaN) に伴う複雑さを回避するためのものです。
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
文字列および関連データ型			
ARRAY	BYTE_ARRAY	STRING	配列は文字列に変換され、BINARY

PostgreSQL のデータ型	Parquet プリミティブ型	論理的な型の注釈	マッピングに関するメモ
			(UTF8) としてエンコードされます。 この変換は、データ精度、数値ではないデータ値 (NaN)、および時間データ値に伴う複雑さを回避するためのものです。
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
XML	BYTE_ARRAY	STRING	
日付と時刻のデータ型			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	

PostgreSQL のデータ型	Parquet プリミティブ型	論理的な型の注釈	マッピングに関するメモ
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
ジオメトリデータ型			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
JSON データ型			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
その他のデータ型			
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	ネットワークデータ型

PostgreSQL のデータ型	Parquet プリミティブ型	論理的な型の注釈	マッピングに関するメモ
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	ネットワークデータ型
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	該当なし		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Amazon RDS の自動バックアップの管理に AWS Backup を使用する

AWS Backup はフルマネージド型のバックアップサービスであり、クラウド内およびオンプレミスの AWS のサービス間でデータのバックアップを簡単に一元化および自動化できます。AWS Backup で Amazon RDS データベースのバックアップを管理できます。

> **Note**
>
> AWS Backup によって管理されるバックアップは手動 DB スナップショットと見なされますが、RDS の DB スナップショットクォータにはカウントされません。AWS Backup で作成されたバックアップの名前は `awsbackup:backup-job-number` で終わります。

AWS Backup の詳細については、AWS Backup デベロッパーガイドを参照してください。

AWS Backup によって管理されているバックアップを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. [バックアップサービス] タブを選択します。

   AWS Backup バックアップは、[バックアップサービスのスナップショット] の下に一覧表示されます。

Amazon RDS インスタンスでのメトリクスのモニタリング

以下のセクションで、Amazon RDS のモニタリングの概要とメトリクスへのアクセス方法の説明を示します。イベント、ログ、およびデータベースアクティビティストリームをモニタリングする方法については、「Amazon RDS DB インスタンスでの、イベント、ログ、およびストリーミングのモニタリング」を参照してください。

トピック

- モニタリング計画
- パフォーマンスのベースライン
- パフォーマンスガイドライン
- Amazon RDS のモニタリングツール
- インスタンスのステータスの表示
- Amazon RDS の推奨事項
- Amazon RDS コンソールでのメトリクスの表示
- Performance Insights ダッシュボードを使用して組み合わせたメトリクスを表示する
- Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング
- CloudWatch Database Insights による Amazon RDS データベースのモニタリング
- Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング
- Amazon DevOps Guru for Amazon RDS でパフォーマンスの異常を分析する
- 拡張モニタリングを使用した OS メトリクスのモニタリング
- Amazon RDS のメトリクスリファレンス

モニタリング計画

Amazon RDS のモニタリングをスタートする前に、モニタリングプランを作成します。この計画で、以下の質問に答えるようにします。

- どのような目的でモニタリングしますか?
- どのリソースをモニタリングしますか?
- どのくらいの頻度でこれらのリソースをモニタリングしますか?

- どのモニタリングツールを使用しますか?
- 誰がモニタリングタスクを実行しますか?
- 問題が発生したときに誰に通知しますか?

パフォーマンスのベースライン

モニタリング目標を達成するには、ベースラインを確立する必要があります。これを行うには、Amazon RDS 環境で負荷条件と時期をさまざまに変えてパフォーマンスを測定します。次のようなメトリクスをモニタリングできます。

- ネットワークスループット
- クライアント接続
- 読み取り、書き込み、メタデータのいずれかのオペレーションの I/O
- DB インスタンスのバーストクレジットバランス

Amazon RDS の履歴パフォーマンスデータを保存することをお勧めします。保存したデータを使用して、現在のパフォーマンスを過去の傾向と比較できます。また、正常なパフォーマンスパターンを異常から区別し、問題に対処するための方法を考案することもできます。

パフォーマンスガイドライン

一般的に、パフォーマンスメトリクスの許容値は、ベースラインに対してアプリケーションの現在の動作によって異なります。ベースラインからの一貫した差異またはトレンドになっている差異を調べます。多くの場合、次のメトリクスがパフォーマンスの問題の原因を示しています。

- CPU または RAM の高消費量 - CPU または RAM の消費量が大きい値になっていても、それは妥当である場合があります。ただし、アプリケーションの目標 (スループット、同時実行数など) に沿った想定値であることが前提です。
- ディスクスペースの消費量 - 使用されているディスクスペースが一貫して合計ディスクスペースの 85% 以上である場合は、ディスクスペースの消費量を調べます。インスタンスからデータを削除するか、別のシステムにデータをアーカイブして、スペースを解放できるかどうかを確認します。
- ネットワークトラフィック - ネットワークトラフィックについてシステム管理者に問い合わせて、ドメインネットワークとインターネット接続に対する想定スループットを把握します。スループットが一貫して想定よりも低い場合は、ネットワークトラフィックを調べます。

- データベース接続数 - ユーザー接続数が多いことが、インスタンスのパフォーマンスが下がっていること、応答時間が長くなっていることに関連しているとわかった場合、データベース接続数を制限することを検討します。DB インスタンスの最適なユーザー接続数は、インスタンスのクラスと実行中のオペレーションの複雑さによって異なります。データベース接続数を確認するには、`User Connections` パラメータが 0 (無制限) 以外の値に設定されているパラメータグループと DB インスタンスを関連付けます。既存のパラメータグループを使用するか、新しいパラメータグループを作成できます。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
- IOPS メトリクス - IOPS メトリクスの想定値はディスクの仕様とサーバーの設定によって異なるため、ベースラインを使用して一般的な値を把握します。値とベースラインとの差が一貫しているかどうかを調べます。最適な IOPS パフォーマンスを得るには、読み取りおよび書き込みオペレーションが最小限になるように、一般的な作業セットがメモリに収まることを確認してください。

確立したベースラインをパフォーマンスが下回ると、場合によって、ワークロードに対してデータベースの可用性を最適化するために変更を加える必要があります。例えば、DB インスタンスのインスタンスクラスの変更が必要になる場合があります。または、クライアントで使用できる DB インスタンスとリードレプリカの数の変更が必要になる場合があります。

Amazon RDS のモニタリングツール

モニタリングは、Amazon RDS およびその他の AWS ソリューションの信頼性、可用性、およびパフォーマンスを維持する上で重要な部分です。AWS には、Amazon RDS を監視したり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするためのモニタリングツールが用意されています。

トピック

- 自動モニタリングツール
- 手動モニタリングツール

自動モニタリングツール

モニタリングタスクをできるだけ自動化することをお勧めします。

トピック

- Amazon RDS インスタンスステータスと推奨事項
- Amazon RDS の Amazon CloudWatch メトリクス
- Amazon RDS Performance Insights とオペレーティングシステムのモニタリング
- 統合サービス

Amazon RDS インスタンスステータスと推奨事項

以下の自動化されたツールを使用して、Amazon RDS をモニタリングし、問題が発生したときにレポートできます。

- Amazon RDS インスタンスステータス – Amazon RDS コンソール、AWS CLI、または RDS API を使用して、インスタンスの現在のステータスに関する詳細を表示します。
- Amazon RDS 推奨事項 — DB インスタンス、、リードレプリカ、DB パラメータグループなどのデータベースリソースに関する推奨事項が自動的に表示されます。詳細については、「Amazon RDS の推奨事項」を参照してください。

Amazon RDS の Amazon CloudWatch メトリクス

Amazon RDS が Amazon CloudWatch と統合し、追加のモニタリング機能が利用できるようになりました。

- Amazon CloudWatch - このサービスは AWS で実行されている AWS リソースやアプリケーションをリアルタイムにモニタリングします。次の Amazon CloudWatch 機能を Amazon RDS で使用できます。
  - Amazon CloudWatch メトリクス - Amazon RDS は、アクティブな各データベースのメトリクスを 1 分ごとに CloudWatch に自動送信します。CloudWatch の Amazon RDS メトリクスに対する追加料金は発生しません。詳細については、Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリングを参照してください。
  - Amazon CloudWatch アラーム - 特定の期間にわたって 1 つの Amazon RDS メトリクスをモニタリングできます。そのため、設定したしきい値に関連するメトリクスの値に基づいて、1 つ以上のアクションを実行できます。詳細については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

Amazon RDS Performance Insights とオペレーティングシステムのモニタリング

Amazon RDS のパフォーマンスをモニタリングするには、次の自動化されたツールを使用します。

- Amazon RDS Performance Insights – データベース負荷を評価し、いつどこに措置を講じたらよいかを判断します。詳しくは、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。
- Amazon RDS 拡張モニタリング – オペレーティングシステムのメトリクスをリアルタイムで参照します。詳しくは、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

統合サービス

次の AWS のサービスが Amazon RDS と統合されます。

- Amazon EventBridge は、アプリケーションをさまざまなソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。詳しくは、「Amazon RDS イベントのモニタリング」を参照してください。
- Amazon CloudWatch Logs を使用すると、Amazon RDS インスタンス、CloudTrail、およびその他のソースからのログファイルをモニタリングして保存し、それらにアクセスすることができます。詳しくは、「Amazon RDS ログファイルのモニタリング」を参照してください。
- AWS CloudTrail は、AWS アカウント により、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。詳しくは、「AWS CloudTrail での Amazon RDS API コールのモニタリング」を参照してください。

- データベースアクティビティストリーミングは、Amazon RDS の一機能であり、Oracle DB インスタンス内のアクティビティのストリーミングをほぼリアルタイムで提供します。詳しくは、「データベースアクティビティストリームを使用した Amazon RDS のモニタリング」を参照してください。

手動モニタリングツール

CloudWatch アラームがカバーしない項目については、手動でモニタリングする必要があります。Amazon RDS、CloudWatch、AWS Trusted Advisor などの AWS コンソールダッシュボードには、AWS 環境の状態が一目でわかるビューが表示されます。また、DB インスタンスのログファイルを確認することをお勧めします。

- Amazon RDS コンソールから、リソースに関する以下の項目をモニタリングできます。
  - DB インスタンスへの接続の数
  - DB インスタンスへの読み書きオペレーションの量
  - DB インスタンスが現在使用しているストレージの量
  - DB インスタンスに使用されているメモリと CPU の量
  - DB インスタンスとの間で送受信されるネットワークトラフィックの量
- Trusted Advisor ダッシュボードから、以下のコスト最適化、セキュリティ、対障害性、パフォーマンス向上のチェックを確認できます。
  - Amazon RDS アイドル DB インスタンス
  - Amazon RDS セキュリティグループのアクセスリスク
  - Amazon RDS バックアップ
  - Amazon RDS Multi-AZ

  これらのチェックの詳細については、「Trusted Advisor のベストプラクティス (チェック)」を参照してください。
- CloudWatch ホームページには、次の内容が表示されます。
  - 現在のアラームとステータス
  - アラームとリソースのグラフ
  - サービスのヘルスステータス

  また、CloudWatch を使用して、次のことが可能です。
  - 重視するサービスをモニタリングするためのカスタマイズしたダッシュボードを作成する。

- メトリクスデータをグラフ化して、問題をトラブルシューティングして、傾向を確認する。
- AWS リソースのすべてのメトリクスを検索およびブラウズする。
- 問題があることを通知するアラームを作成/編集する。

インスタンスのステータスの表示

Amazon RDS コンソールを使用すると、DB インスタンスのステータスにすばやくアクセスできます。

トピック

- Amazon RDS DB インスタンスのステータスの表示

Amazon RDS DB インスタンスのステータスの表示

の DB インスタンスのステータスは、現在の運用状態を示します。次の手順で、Amazon RDS コンソール、AWS CLI コマンド、または API オペレーションで DB インスタンスのステータスを表示できます。

> **Note**
>
> Amazon RDS では、メンテナンスのステータスと呼ばれる別のステータスも使用します。これは、Amazon RDS コンソールの [メンテナンス] 列に表示されます。この値は、DB インスタンスに適用する必要があるメンテナンスパッチのステータスを示します。メンテナンスのステータスは、DB インスタンスのステータスから独立しています。メンテナンスのステータスの詳細については、「DB インスタンスへの更新の適用」を参照してください。

DB インスタンスの考えられるステータス値を以下の表に示します。また、この表は、DB インスタンスとストレージが請求されるか、ストレージのみ請求されるか、または請求されないかを示します。DB インスタンスのすべてのステータスで、バックアップの使用は常に請求されます。

DB インスタンスのステータス	請求される	説明
使用可能	請求対象	DB インスタンスは変更可能です。
バックアップ中	請求される	DB インスタンスを現在バックアップ中です。
拡張モニタリングを設定中	請求される	この DB インスタンスに対して拡張モニタリングを有効または無効にしています。
iam データベース認証を設定中	請求される	AWS Identity and Access Managementこの DB インスタンスに対して (IAM) データベース認証を有効または無効にしています。
ログエクスポートを設定中	請求される	この DB インスタンスに対して Amazon CloudWatch Logs へのログファイルの発行を有効または無効にしています。

DB インスタンスのステータス	請求される	説明
vpc に変換中	請求される	DB インスタンスを、Amazon Virtual Private Cloud (Amazon VPC) 外の DB インスタンスから Amazon VPC 内の DB インスタンスに変換中です。
作成中	請求されない (PITR 以外) 請求される (PITR のみ)	DB インスタンスを作成中です。作成中の DB インスタンスにはアクセスできません。 ポイントインタイムリカバリ (PITR) 中にデータベースを復元する場合、データベースが作成中状態の間は課金されます。これは、作成中状態で料金が発生する唯一のシナリオです。
削除の事前チェック	非請求対象	Amazon RDS は、リードレプリカを削除しても安全かどうかを検証しています。
削除中	課金されない	DB インスタンスを削除しています。
「失敗」	課金されない	DB インスタンスでエラーが発生し、Amazon RDS では復旧できません。DB インスタンスの復元可能な直近の時間までポイントインタイムリカバリを実行し、データを復旧してください。
暗号化認証情報にアクセスできません	非請求対象	DB インスタンスの暗号化または復号に使用する AWS KMS key にアクセスしたり、それを復元したりすることはできません。
inaccessible-encryption-credentials-recoverable	ストレージが請求対象	DB インスタンスの暗号化または復号に使用する KMS キーにアクセスできません。ただし、KMS キーがアクティブな場合は、DB インスタンスを再起動すると復元できます。 詳細については、「DB インスタンスの暗号化」を参照してください。

DB インスタンスのステータス	請求される	説明
互換性がない、作成	非請求対象	Amazon RDS は DB インスタンスを作成しようとしていますが、リソースが DB インスタンスと互換性がないため、作成できません。このステータスは、例えば DB インスタンスのインスタンスプロファイルに適切なアクセス許可がない場合に発生する可能性があります。
互換性のないネットワーク	課金されない	Amazon RDS は、DB インスタンスに対して復旧アクションを実行しようとしていますが、VPC がアクションを完了できない状態にあるため実行できません。このステータスは、例えば、サブネット内の使用可能なすべての IP アドレスが使用中で、Amazon RDS が DB インスタンスの IP アドレスを取得できない場合などに発生する可能性があります。
互換性のないオプショングループ	請求される	Amazon RDS がオプショングループの変更を適用しようとしましたが、適用できませんでした。また、Amazon RDS はオプショングループの前の状態にロールバックできませんでした。詳細については、DB インスタンスの [最近のイベント] 一覧を参照してください。このステータスは、例えば、オプショングループに TDE などのオプションが含まれており、DB インスタンスに暗号化情報が含まれていない場合などに発生する可能性があります。
互換性のないパラメータ	請求される	DB インスタンスの DB パラメータグループに指定されたパラメータが DB インスタンスと互換性がないため、Amazon RDS は DB インスタンスを起動できません。パラメータの変更を元に戻すか、パラメータを DB インスタンスと互換させて、DB インスタンスへのアクセスを回復してください。互換性のないパラメータの詳細については、DB インスタンスの [最近のイベント] 一覧を参照してください。

DB インスタンスのステータス	請求される	説明
互換性のない復元	課金されない	Amazon RDS は、特定の時点への復旧を行うことはできません。この状況の一般的な原因としては、temp テーブルの使用、MySQL での MyISAM テーブルの使用、または MariaDB での Aria テーブルの使用が考えられます。
insufficient-capacity	非請求対象	十分な容量が現在利用できないため、Amazon RDS はインスタンスを作成できません。同じ AZ に、同じインスタンスタイプで DB インスタンスを作成するには、DB インスタンスを削除し、数時間待ってから、もう一度作成を試みます。または、別のインスタンスクラスまたは AZ を使用して新しいインスタンスを作成します。
メンテナンス	請求される	Amazon RDS は、DB インスタンスにメンテナンス更新を適用しています。このステータスは、RDS が事前に十分スケジュールしたインスタンスレベルのメンテナンスに使用されます。
変更中	請求される	ユーザーからの DB インスタンスの変更リクエストに応じて、DB インスタンスを変更中です。
vpc に移動中	請求される	DB インスタンスを新しい Amazon Virtual Private Cloud (Amazon VPC) に移動中です。
再起動	請求される	DB インスタンスの再起動を要求するユーザーのリクエストまたは Amazon RDS プロセスに応じて、インスタンスを再起動中です。
マスター認証をリセット中	請求される	ユーザーからのリセットのリクエストに応じて、DB インスタンスのマスター認証情報をリセット中です。
名前変更中	請求される	ユーザーからの名前変更のリクエストに応じて、DB インスタンスの名前を変更中です。

DB インスタンスのステータス	請求される	説明
復元エラー	請求される	特定時点への復元またはスナップショットからの復元を実行しようとした際に、DB インスタンスでエラーが発生しました。
スタート中	ストレージが請求対象	DB インスタンスを起動中です。
停止済み	ストレージが請求対象	DB インスタンスは停止済みです。
停止	ストレージが請求対象	DB インスタンスを停止中です。
ストレージ設定をアップグレード	請求対象	DB インスタンスのストレージファイルシステム設定をアップグレード中です。このステータスはブルー/グリーンデプロイ内のグリーンデータベース、または DB インスタンスのリードレプリカにのみ適用されます。
ストレージ不足	請求される	DB インスタンスが、ストレージ容量の割り当て分に達しました。これは非常に重要なステータスで、この問題はすぐに修正することをお勧めします。これを行うには、DB インスタンスを変更してストレージを拡張します。このような状況を回避するために、ストレージ容量が減少したときに警告を生成する Amazon CloudWatch アラームを設定します。

DB インスタンスのステータス	請求される	説明
ストレージの初期化	請求対象	DB インスタンスは、スナップショットから復元された後のボリュームパフォーマンスを最適化するために、Amazon S3 からデータブロックをロードしています。オペレーションで引き続き使用できますが、初期化が完了するまでパフォーマンスが最大にならない可能性があります。 このステータスは、ブルー/グリーンデプロイのグリーン環境に対してのみ設定されます。Amazon S3 からボリュームをハイドレートする他のワークフロー (ポイントインタイムリカバリ、スナップショット復元、リードレプリカの作成、シングル AZ からマルチ AZ への変換など) では、初期化がバックグラウンドで実行されている間も DB インスタンスのステータスは [利用可能] のままになります。すべてのケースでボリュームごとの初期化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。初期化中、`StorageOperationStatus` は `Initializing` に設定されます。

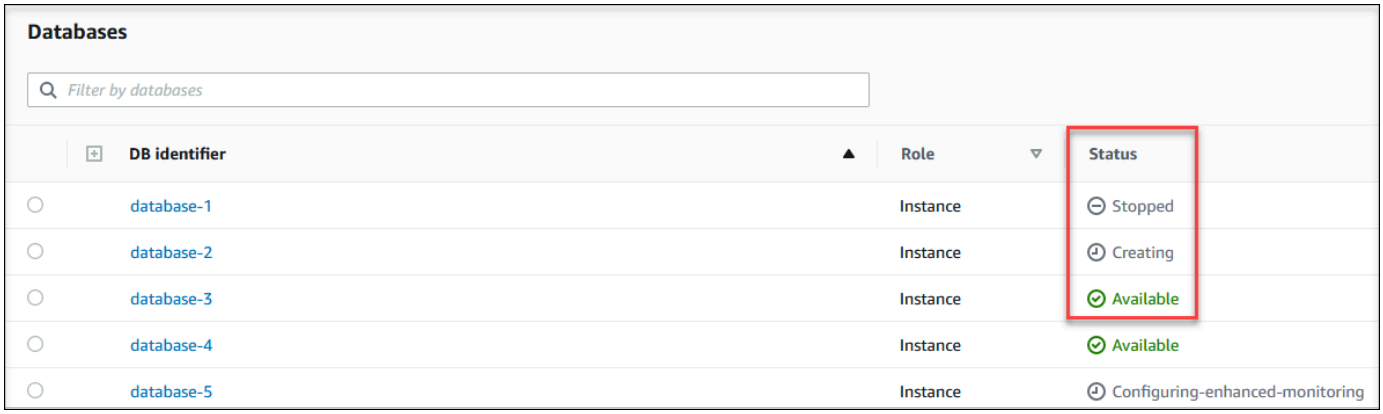
DB インスタンスのステータス	請求される	説明
ストレージの最適化	請求対象	Amazon RDS が、DB インスタンスのストレージを最適化しています。ストレージ最適化プロセスは通常短時間で終了しますが、場合によっては 24 時間以上かかることもあります。 ストレージの最適化中、DB インスタンスは引き続き使用できます。ストレージの最適化は、インスタンスの可用性に影響を与えないバックグラウンドプロセスです。 ボリュームごとの最適化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを使用します。最適化中、`StorageOperationStatus` は `Optimizing` に設定されます。
アップグレード	請求対象	データベースエンジンまたはオペレーティングシステムのバージョンをアップグレード中です。
アップグレード失敗	非請求対象	DB インスタンスをサポートされているバージョンにアップグレードできませんでした。Aurora は、プレフィックス `rds-final` を持つ最終スナップショットを作成します。

コンソール

DB インスタンスのステータスを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。

   データベースページが DB インスタンスのリストとともに表示されます。 DB クラスターごとに、ステータス値が表示されます。



CLI

AWS CLI を使用して DB インスタンスとそのステータス情報を表示するには、describe-db-instances コマンドを使用します。例えば、次の AWS CLI コマンドは、すべての DB インスタンス情報を一覧表示します。

```
aws rds describe-db-instances
```

特定の DB インスタンスとそのステータスを表示するには、次のオプションを指定して describe-db-instances コマンドを呼び出します。

- `DBInstanceIdentifier` - DB インスタンスの名前です。

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

DB インスタンスのステータスだけを表示するには、AWS CLI で次のクエリを使用します。

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

API

Amazon RDS API を使用して DB インスタンスのステータスを表示するには、DescribeDBInstances オペレーションを呼び出します。

Amazon RDS の推奨事項

Amazon RDS では、DB インスタンス、リードレプリカ、DB パラメータグループなどのデータベースリソースについての推奨事項が自動で表示されます。これらの推奨事項は、、DB インスタンス構成、使用状況、パフォーマンスデータを分析して、ベストプラクティスガイダンスを提供します。

Amazon RDS Performance Insights は特定のメトリクスを監視し、特定のリソースで潜在的に問題であると見なされるレベルを分析することで自動的にしきい値を作成します。新しいメトリクス値が事前定義されたしきい値を一定期間にわたって超えた場合に、Performance Insights は事前対応型推奨事項を生成します。この推奨事項は、将来のデータベースパフォーマンスへの影響を防ぐのに役立ちます。例えば、データベースに接続されているセッションがアクティブな作業を行っていないが、データベースリソースがブロックされている可能性がある場合、RDS for PostgreSQL インスタンスに対して「トランザクションでのアイドル状態」という推奨事項が生成されます。事前対応型推奨事項を受け取るには、有料利用枠の保持期間を使用して Performance Insights を有効にする必要があります。Performance Insights を有効にする方法については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。Performance Insights の料金とデータ保持については、「Performance Insights の料金とデータ保持」を参照してください。

DevOps Guru for RDS は特定のメトリクスを監視して、メトリクスの動作が非常に異常または異常になったことを検出します。これらの異常は、推奨事項を含む事後対応型インサイトとして報告されます。例えば、DevOps Guru for RDS は、CPU 容量の増加を検討したり、DB ロードに寄与している待機イベントを調査するよう推奨することがあります。DevOps Guru for RDS は、しきい値ベースの事前対応型推奨事項も提供します。これらの推奨事項を受け取るには、DevOps Guru for RDS を有効にする必要があります。DevOps Guru for RDS を有効にする方法については、「DevOps Guru をオンにしてリソースカバレッジを指定する」を参照してください。

推奨事項のステータスは、アクティブ、却下、保留中、解決済みのいずれかになります。解決済みの推奨事項は 365 日間利用できます。

推奨事項は、表示または却下できます。設定ベースのアクティブな推奨事項をすぐに適用したり、次のメンテナンスウィンドウでスケジュールしたり、却下したりできます。しきい値ベースの事前対応型推奨事項と機械学習ベースの事後対応型推奨事項については、推奨される問題の原因を確認し、推奨アクションを実行して問題を解決する必要があります。

推奨事項は、以下の AWS リージョンでサポートされています。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)

- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (ストックホルム)
- 南米 (サンパウロ)

Amazon RDS の推奨事項を表示、適用、却下、および変更する方法については、以下のセクションを参照してください。。

トピック

- Amazon RDS の推奨事項の表示
- Amazon RDS の推奨事項の適用
- Amazon RDS の推奨事項の却下
- 却下された Amazon RDS 推奨事項のアクティブな推奨事項への変更
- Amazon RDS からの推奨事項のリファレンス

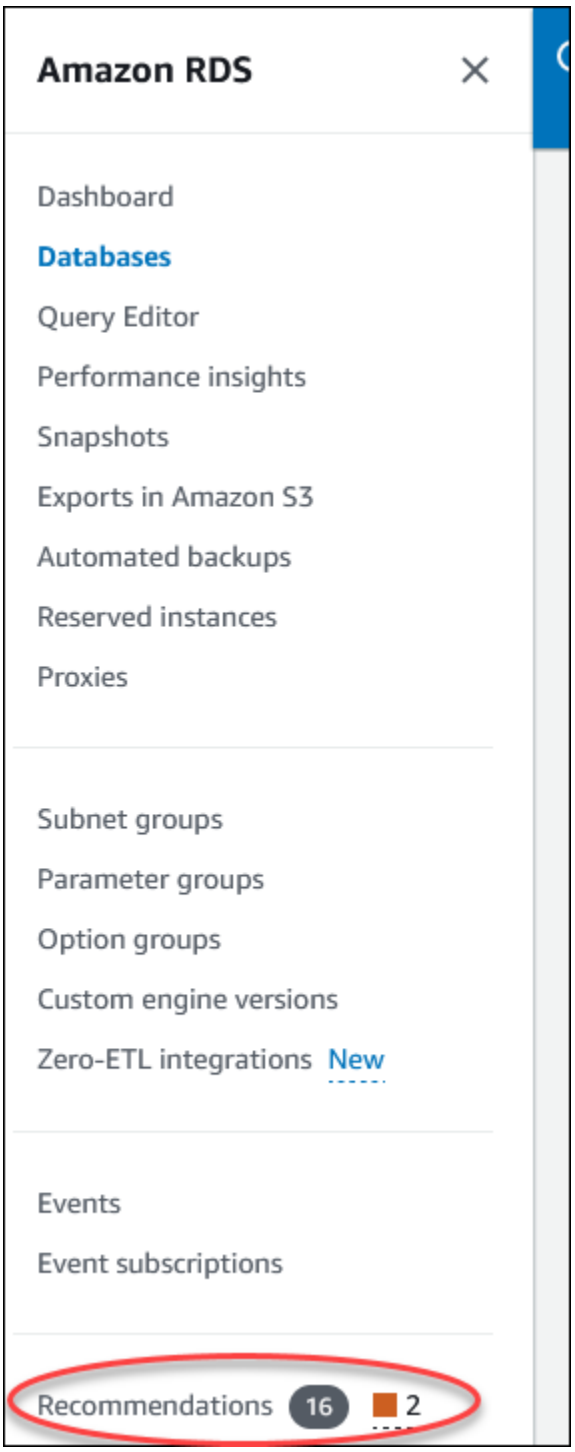
Amazon RDS の推奨事項の表示

Amazon RDS コンソールを使用して、データベースリソースに関する Amazon RDS の推奨事項を表示できます。

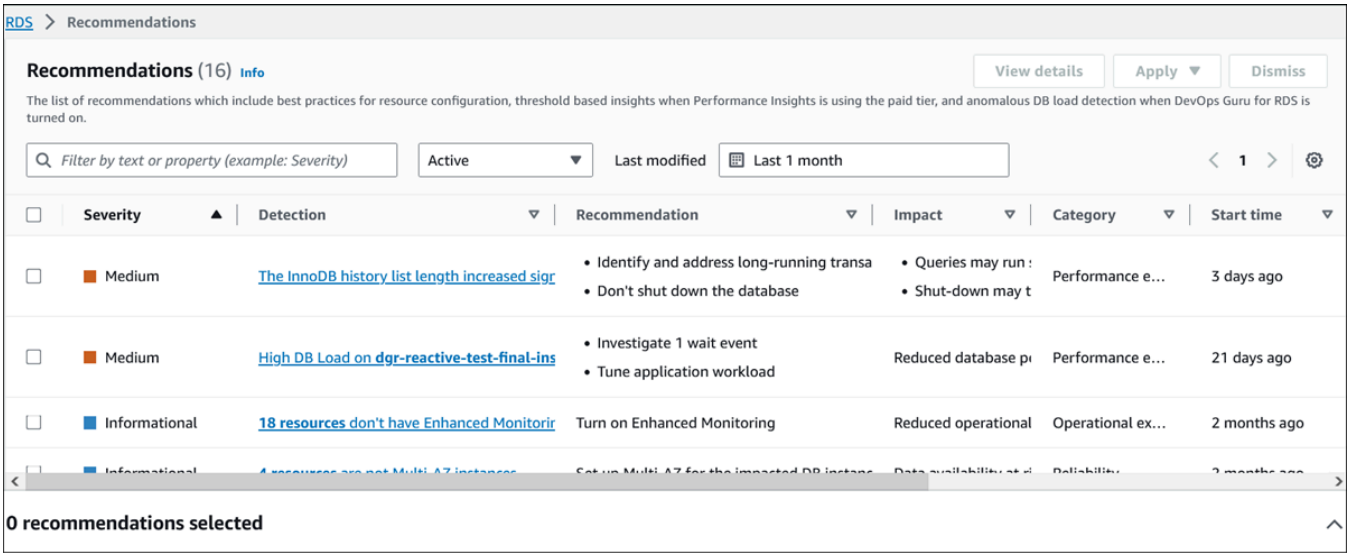
コンソール

Amazon RDS の推奨事項を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、次のいずれかを実行します。
   - [レコメンデーション] を選択します。リソースに対するアクティブな推奨事項の数と、前の月に生成された重要度が最も高い推奨事項の数は、[レコメンデーション] の横に表示されます。重要度ごとにアクティブな推奨事項の数を確認するには、最も重要度が高い番号を選択します。

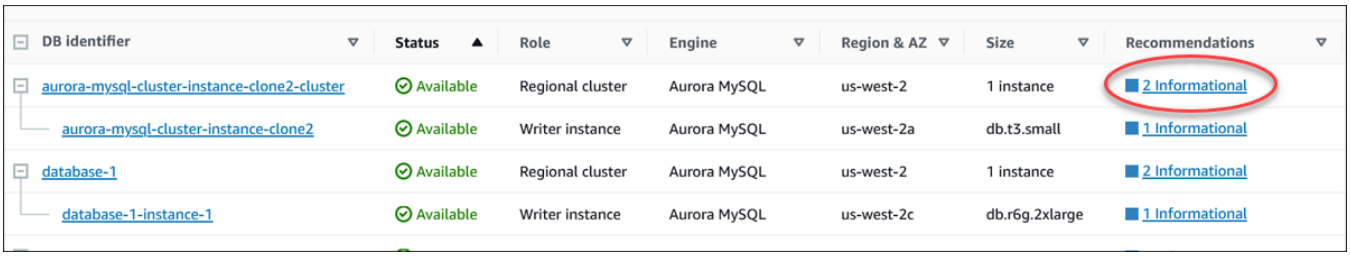


デフォルトでは、[レコメンデーション] ページには前の月の新しい推奨事項のリストが表示されます。Amazon RDS は、アカウント内のすべてのリソースの推奨事項を提供し、重要度でソートします。

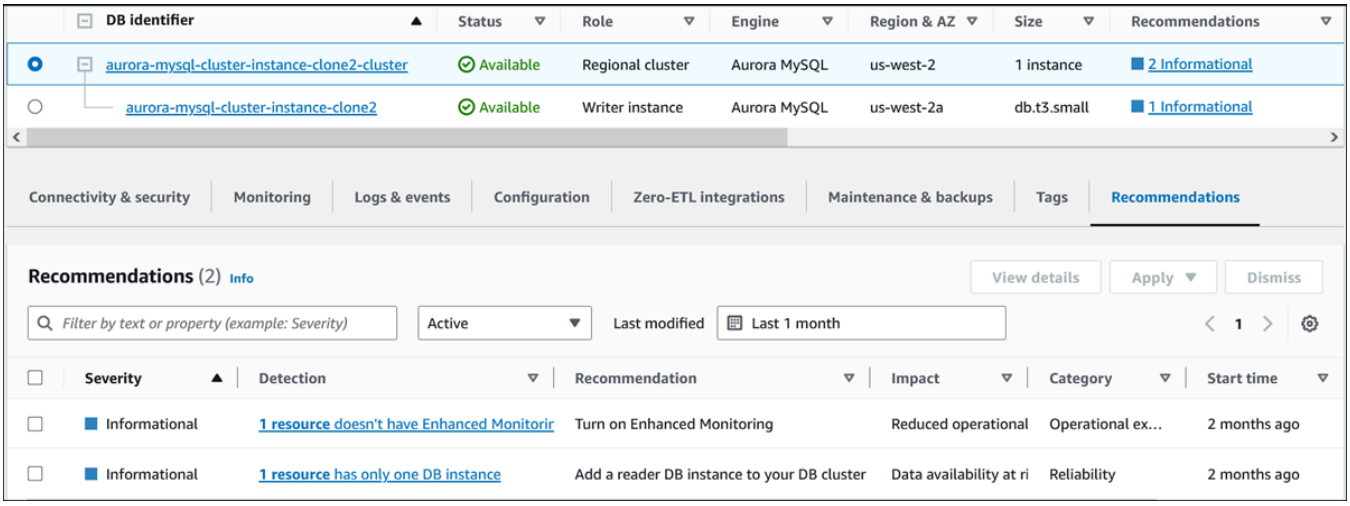


レコメンデーションを選択すると、影響を受けるリソースとレコメンデーションを適用する方法の詳細が表示されます。

- [データベース] ページで、リソースの [レコメンデーション] を選択します。



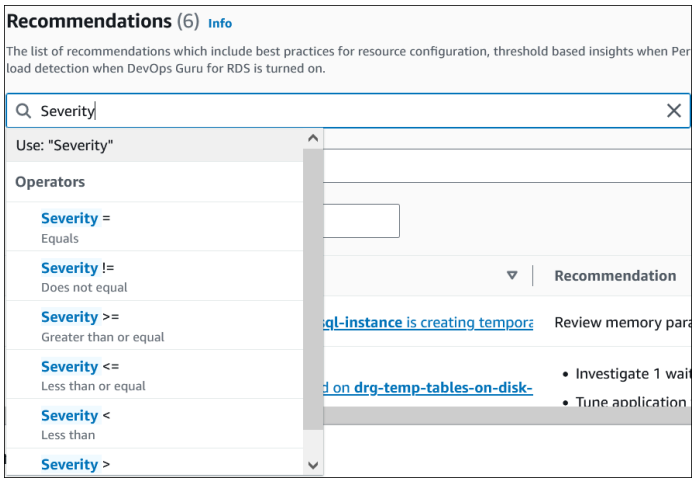
[レコメンデーション] タブには、選択したリソースの推奨事項とその詳細が表示されます。



推奨事項には以下の詳細があります。

- [重要度] — 問題の意味するレベル。重要度レベルは、[高]、[中]、[低]、および [情報] です。
- [検出] — 影響を受けるリソースの数と問題の簡単な説明。このリンクを選択すると、推奨事項と分析の詳細が表示されます。
- [レコメンデーション] — 適用する推奨事項アクションの簡単な説明。
- [影響] — 推奨事項が適用されない場合に発生する可能性のある影響の簡単な説明。
- [カテゴリ] – 推奨事項のタイプ。カテゴリは、[パフォーマンス効率]、[セキュリティ]、[信頼性]、[コスト最適化]、[運用上の優秀性]、[持続可能性]です。
- [ステータス] – ターゲットエンジンの推奨事項のステータス 指定できるステータスは、[すべて]、[アクティブ]、[却下済み]、[解決済み]、[保留中] です。
- [開始時刻] – 問題が開始された時刻。例えば、18 時間前と指定します。
- [最終更新日] – [重要度] の変更により推奨事項がシステムによって最後に更新された時刻、または推奨事項に応答した時刻。例えば、10 時間前と指定します。
- [終了時刻] – 問題が終了した時刻。時間には、継続中の問題は表示されません。
- [リソース識別子] – 1 つ以上のリソースの名前。

3. (オプション) フィールドで [重要度] または [カテゴリ] 演算子を選択して、推奨事項のリストをフィルタリングします。

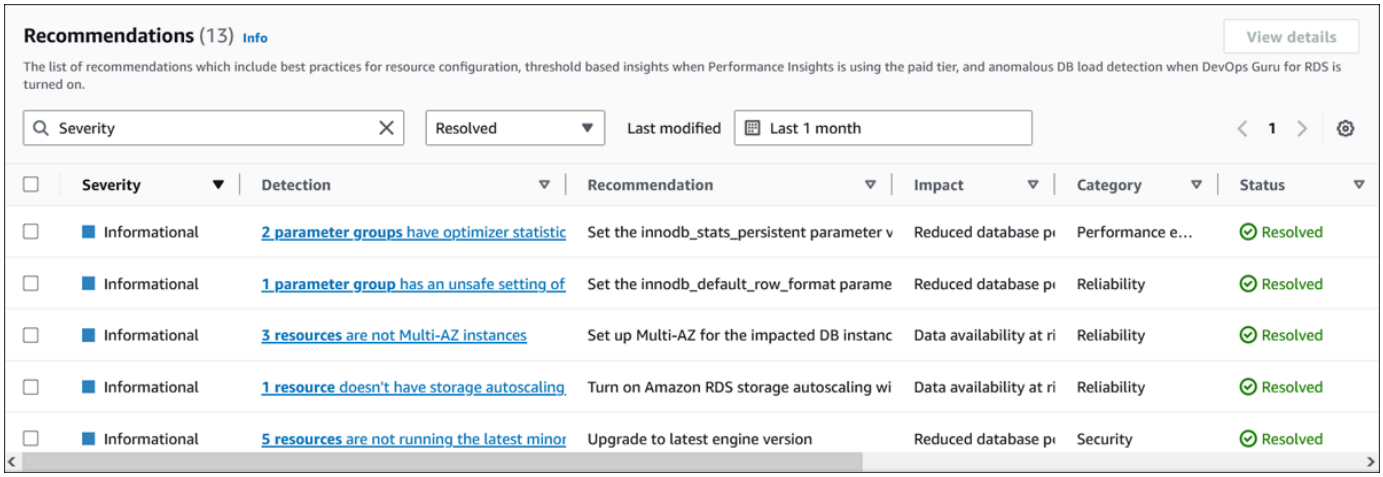


選択したオペレーションの推奨事項が表示されます。

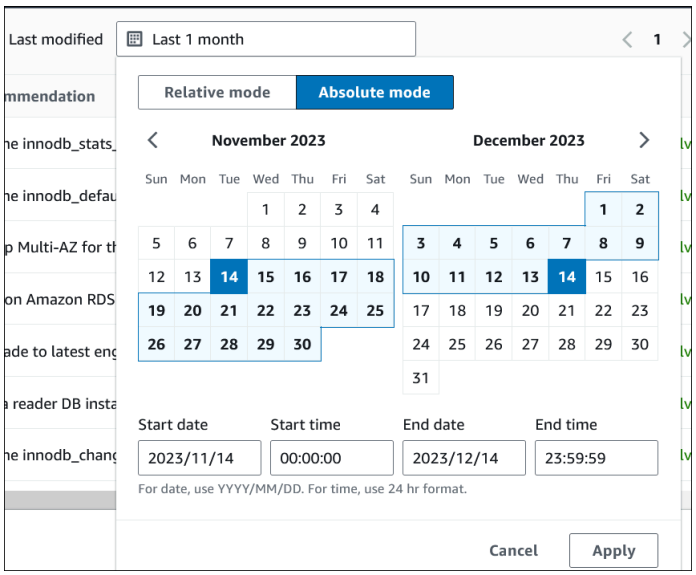
4. (オプション) 次のいずれかの推奨事項ステータスを選択します。

- [アクティブ] (デフォルト) - 次のメンテナンスウィンドウで適用、スケジュール設定できるか、または却下できる現在の推奨事項が表示されます。
- [すべて] – 現在のステータスのすべての推奨事項を表示します。
- [却下済み] – 却下された推奨事項を表示します。

- [解決済み] — 解決済みの推奨事項を表示します。
- [保留中] — 推奨アクションが進行中であるか、次のメンテナンスウィンドウにスケジュールされている推奨事項を表示します。



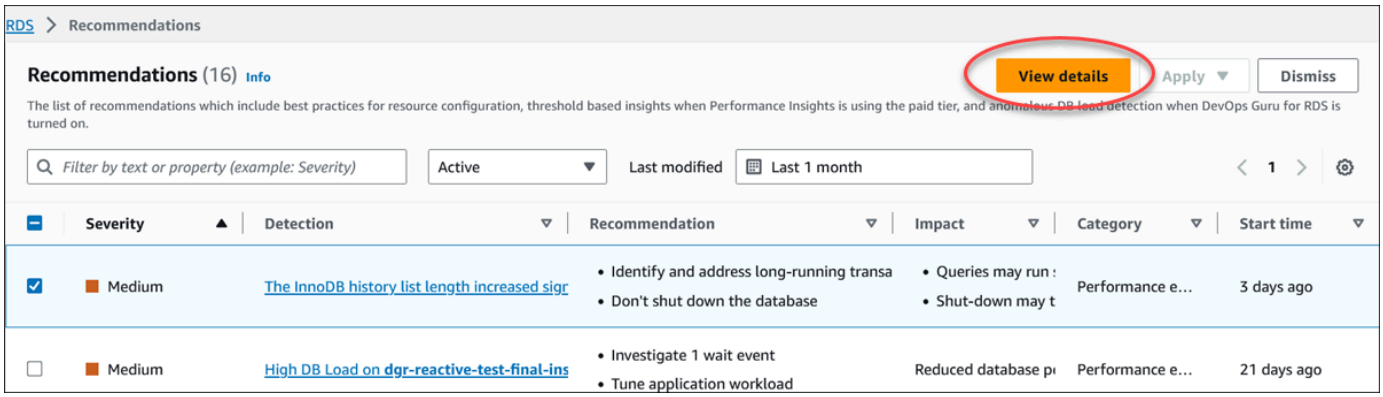
5. (オプション) [最終更新日] の [相対モード] または [絶対モード] を選択して期間を変更します。[レコメンデーション] ページには、期間中に生成された推奨事項が表示されます。デフォルトの期間は過去 1 か月です。[絶対モード] では、期間を選択するか、[開始日] フィールドと [終了日] フィールドに時刻を入力できます。



設定された期間表示の推奨事項。

範囲を [すべて] に設定することで、アカウント内のリソースに関するすべての推奨事項を表示できます。

6. (オプション) 右側の [詳細設定] を選択して、表示する詳細をカスタマイズします。ページサイズを選択し、テキストの行を折り返して、列を許可または非表示にすることができます。
7. (オプション) 推奨事項を選択し、[詳細を表示] を選択します。



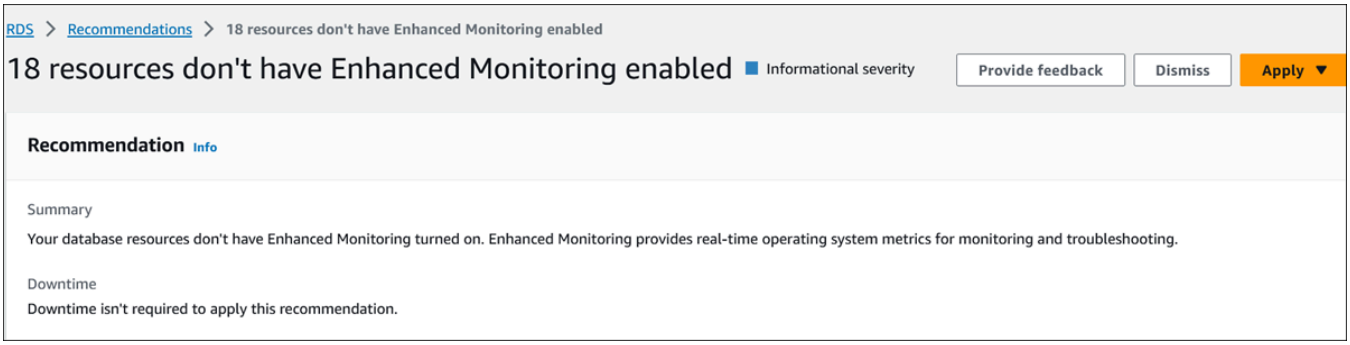
推奨事項の詳細ページが表示されます。タイトルには、問題が検出されたリソースの総数と重要度が表示されます。

異常ベースの事後推奨事項の詳細ページにあるコンポーネントについては、Amazon DevOps Guru ユーザーガイド 「事後対応型異常を表示する」を参照してください。

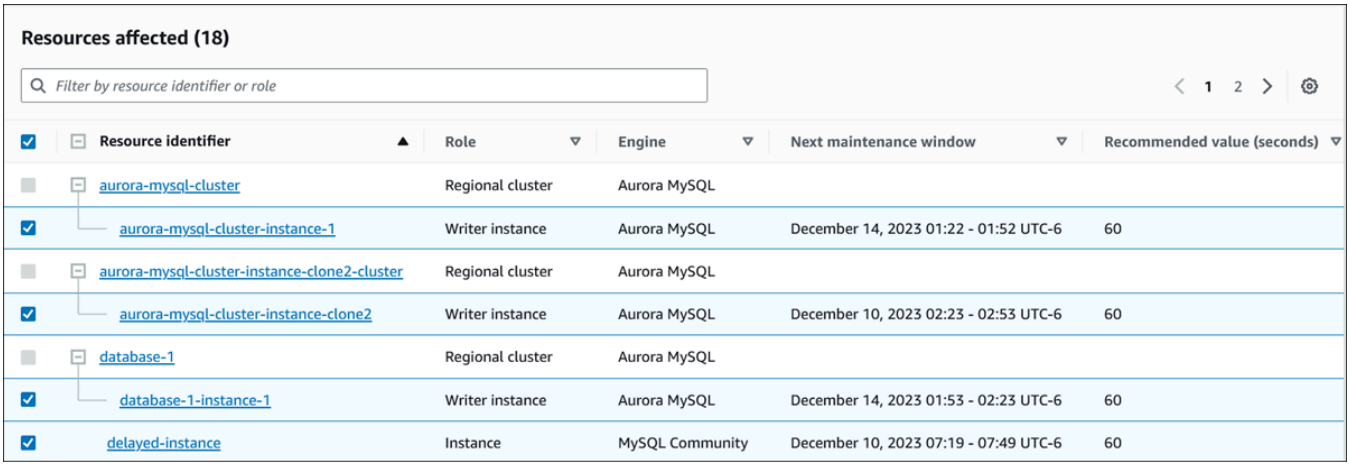
しきい値ベースのプロアクティブ推奨事項の詳細ページのコンポーネントについては、「」を参照してくださいPerformance Insights のプロアクティブ推奨事項の表示。

その他の自動推奨事項では、推奨事項の詳細ページに次のコンポーネントが表示されます。

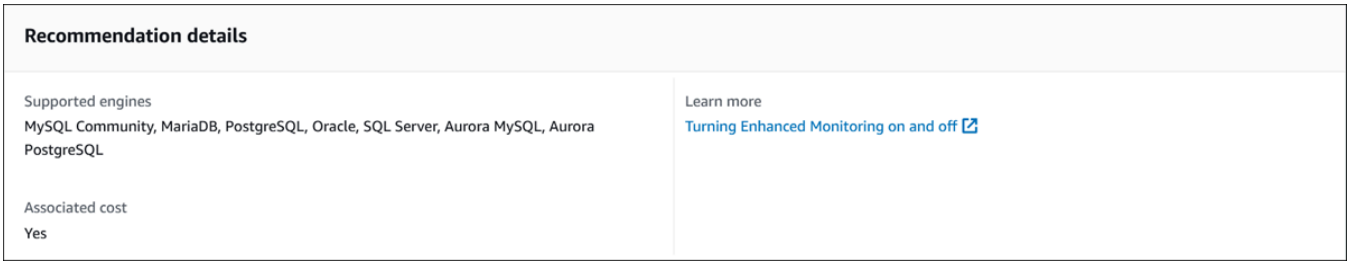
- 推奨事項 — 推奨事項の概要と、推奨事項を適用するためにダウンタイムが必要かどうか。



- [影響を受けるリソース] — 影響を受けるリソースの詳細。



- 推奨事項の詳細 – サポートされるエンジン情報、推奨事項を適用するために必要な関連コスト、および詳細情報を確認するためのドキュメントリンク。



CLI

DB インスタンスの Amazon RDS の推奨事項を表示するには、AWS CLI で次のコマンドを使用します。

```
aws rds describe-db-recommendations
```

RDS API

Amazon RDS API を使用して Amazon RDS の推奨事項を表示するには、DescribeDBRecommendations オペレーションを使用します。

Amazon RDS の推奨事項の適用

Amazon RDS コンソールを使用して Amazon RDS 推奨事項を適用するには、詳細ページで設定ベースの推奨事項または影響を受けるリソースを選択します。次に、推奨事項をすぐに適用するか、次のメンテナンスウィンドウにスケジュールするかを選択します。(これらの変更を有効にするには、クラスターの再起動が必要な場合があります。) いくつかの DB パラメータグループの推奨事項については、リソースの再起動が必要になる場合があります。

しきい値ベースの事前対応型推奨事項または異常ベースの事後対応型推奨事項には適用オプションがないため、追加のレビューが必要になる場合があります。

コンソール

設定ベースの推奨事項を適用するには

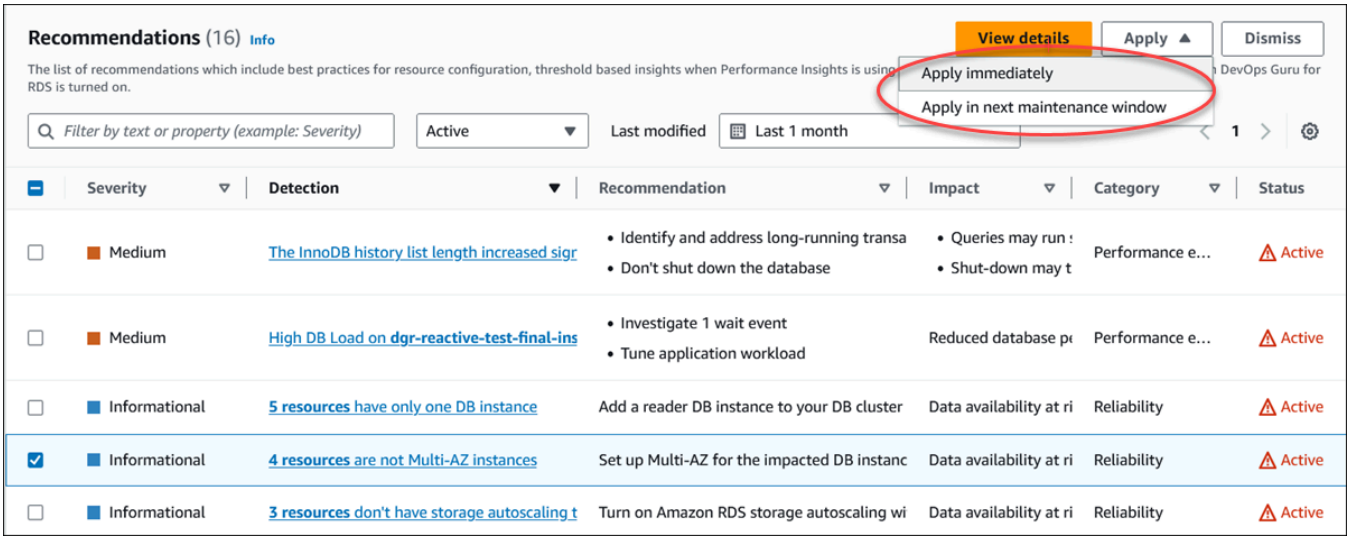
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、次のいずれかを実行します。
   - [レコメンデーション] を選択します。

     [レコメンデーション] ページには、すべての推奨事項のリストが表示されます。
   - [データベース] を選択し、データベースページでリソースの [レコメンデーション] を選択します。

     詳細は、選択した推奨事項の [レコメンデーション] タブに表示されます。
   - [レコメンデーション] ページでアクティブな推奨事項の [検出]、または [データベース] ページの [レコメンデーション] タブを選択します。

     推奨事項の詳細ページが表示されます。
3. 推奨事項、または推奨事項の詳細ページで影響を受ける 1 つ以上のリソースを選択し、次のいずれかを実行します。
   - [適用] を選択し、[今すぐ適用] を選択して、すぐに推奨事項を適用します。
   - [適用] を選択し、次に [次のメンテナンスウィンドウで適用] を選択して、次のメンテナンスウィンドウ中にスケジュール設定します。

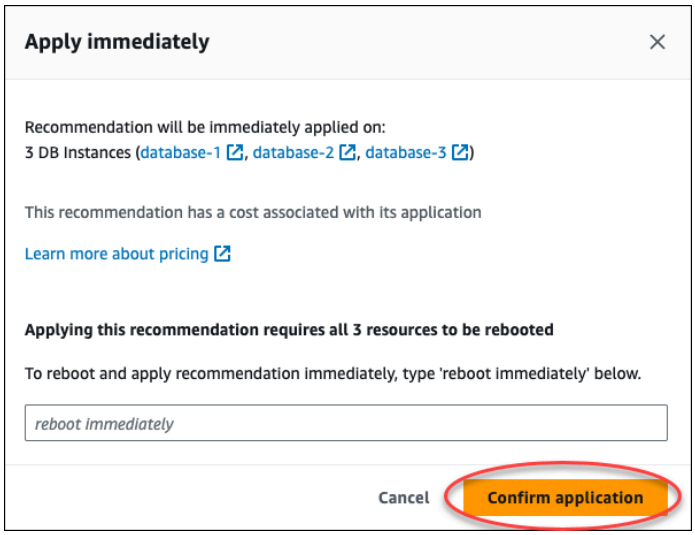
     選択した推奨事項ステータスは、次のメンテナンスウィンドウまで保留に更新されます。



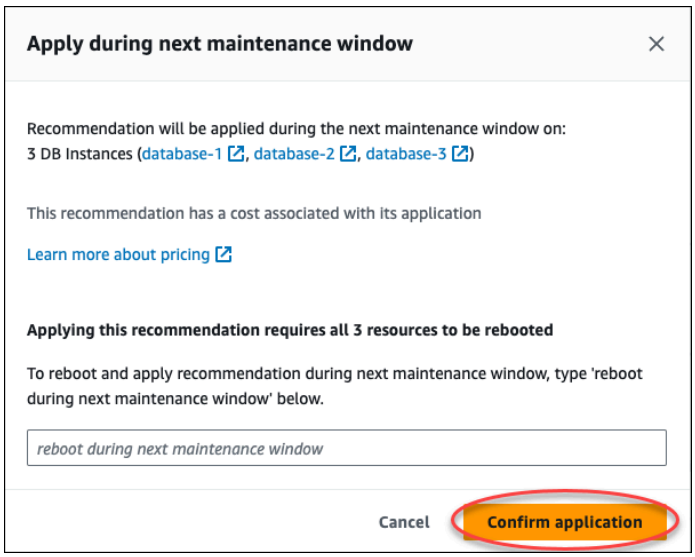
確認ウィンドウが表示されます。

4. [アプリケーションの確認] を選択して、推奨事項を適用します。このウィンドウは、変更を有効にするためにリソースを自動再起動する必要があるか手動で再起動する必要があるかを確認します。

次の例は、推奨事項をすぐに適用するための確認ウィンドウを示しています。



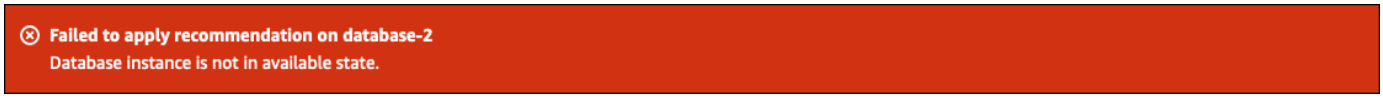
次の例は、次のメンテナンスウィンドウで推奨事項の適用をスケジュールする確認ウィンドウを示しています。



バナーには、適用された推奨事項が成功または失敗したときにメッセージが表示されます。

次の例は、成功メッセージを含むバナーを示しています。

次の例は、失敗メッセージを含むバナーを示しています。



RDS API

Amazon RDS API を使用して設定ベースの RDS 推奨事項を適用するには

1. DescribeDBRecommendations オペレーションを使用します。出力の `RecommendedActions` には、1 つ以上の推奨アクションを含めることができます。
2. ステップ 1 の推奨アクションごとに RecommendedAction オブジェクトを使用します。出力には `Operation` と `Parameters` が含まれます。

   次の例は、1 つの推奨アクションを含む出力を示しています。

```
"RecommendedActions": [
    {
        "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
```

```
            "Title": "Turn on auto backup", // localized
            "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
            "Operation": "ModifyDbInstance",
            "Parameters": [
                {
                    "Key": "DbInstanceIdentifier",
                    "Value": "my-mysql-instance-1"
                },
                {
                    "Key": "BackupRetentionPeriod",
                    "Value": "7"
                }
            ],
            "ApplyModes": ["immediately", "next-maintenance-window"],
            "Status": "applied"
        },
        ... // several others
    ],
```

3. ステップ 2 の出力からの推奨アクションごとに `operation` を使用し、`Parameters` 値を入力します。
4. ステップ 2 の オペレーションが成功したら、ModifyDBRecommendation オペレーションを使用して推奨事項のステータスを変更します。

Amazon RDS の推奨事項の却下

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、1 つ以上の Amazon RDS 推奨事項を却下できます。

コンソール

1 つまたは複数の推奨事項を却下するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、次のいずれかを実行します。
   - [レコメンデーション] を選択します。

[レコメンデーション] ページには、すべての推奨事項のリストが表示されます。

- [データベース] を選択し、データベースページでリソースの [レコメンデーション] を選択します。

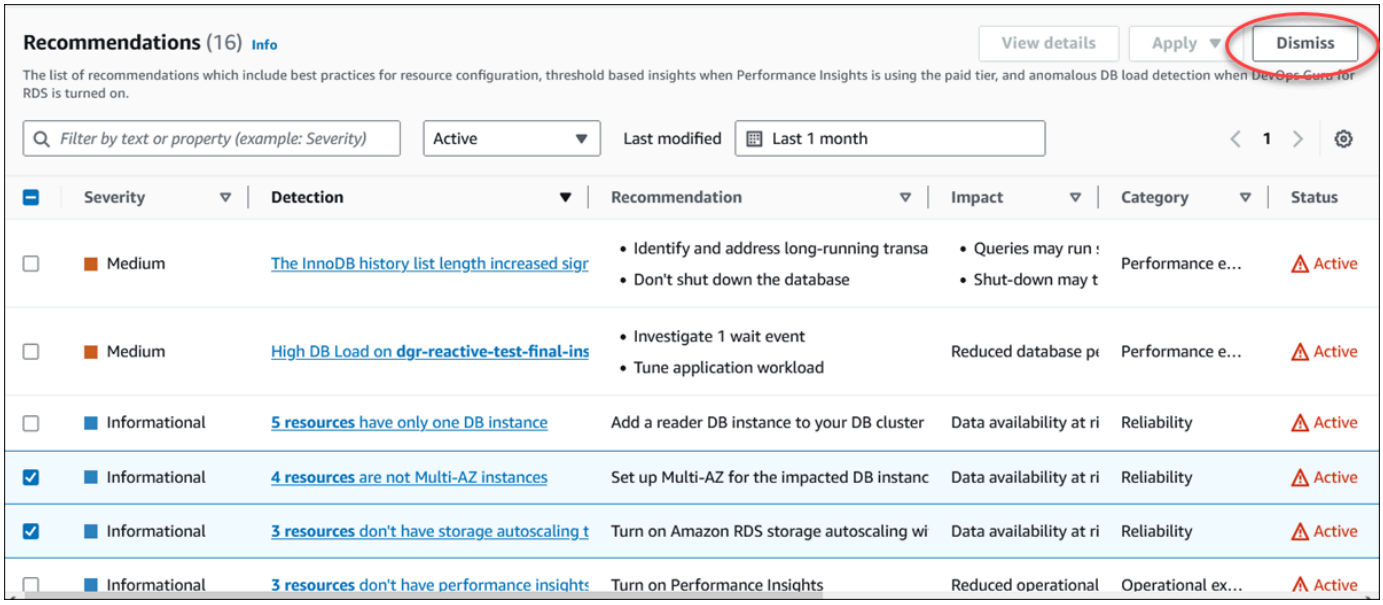
  詳細は、選択した推奨事項の [レコメンデーション] タブに表示されます。

- [レコメンデーション] ページでアクティブな推奨事項の [検出]、または [データベース] ページの [レコメンデーション] タブを選択します。

  推奨事項の詳細ページには、影響を受けるリソースのリストが表示されます。

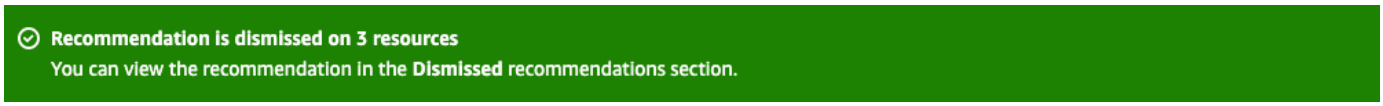
3. 1 つ以上の推奨事項を選択するか、推奨事項の詳細ページで影響を受けるリソースを 1 つ以上選択し、[閉じる] を選択します。

次の例は、却下するアクティブな推奨事項が複数選択されている [レコメンデーション] ページを示しています。



選択した 1 つ以上の推奨事項が却下されると、バナーにメッセージが表示されます。

次の例は、成功メッセージを含むバナーを示しています。



次の例は、失敗メッセージを含むバナーを示しています。

CLI

AWS CLI を使用して RDS 推奨事項を却下するには

1. コマンドを実行します。`aws rds describe-db-recommendations --filters "Name=status,Values=active"`

   出力には、`active` ステータスの推奨事項のリストが表示されます。
2. ステップ 1 で却下する推奨事項の `recommendationId` を見つけます。
3. ステップ 2 の `recommendationId` で `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` コマンドを実行して、推奨事項を却下します。

RDS API

Amazon RDS API を使用して RDS 推奨事項を却下するには、ModifyDBRecommendation オペレーションを使用します。

却下された Amazon RDS 推奨事項のアクティブな推奨事項への変更

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、1 つ以上の却下された Amazon RDS 推奨事項をアクティブな推奨事項に移動できます。

コンソール

却下された推奨事項をアクティブな推奨事項に移動するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、次のいずれかを実行します。

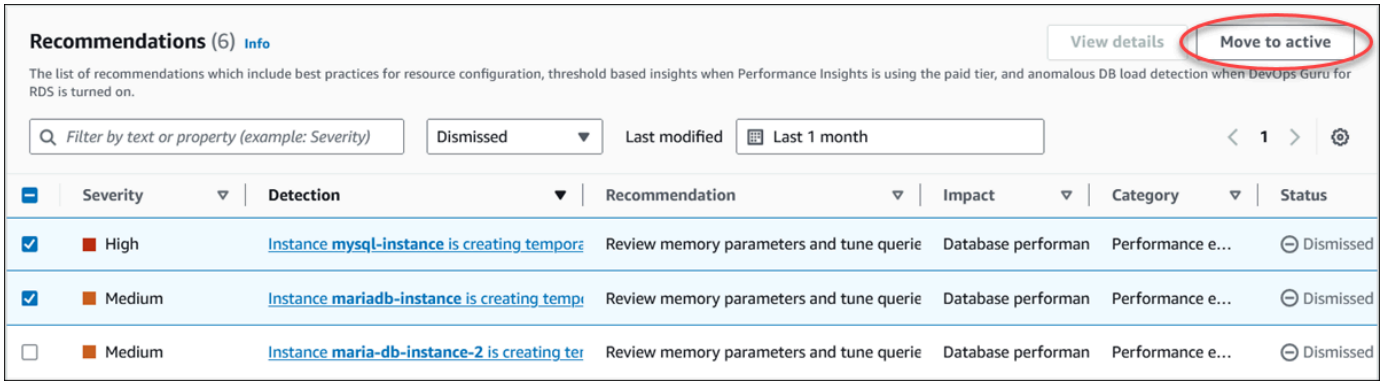
   - [レコメンデーション] を選択します。

     [レコメンデーション] ページには、アカウント内のすべてのリソースの重要度でソートされた推奨事項のリストが表示されます。

- [データベース] を選択し、データベースページでリソースの [レコメンデーション] を選択します。

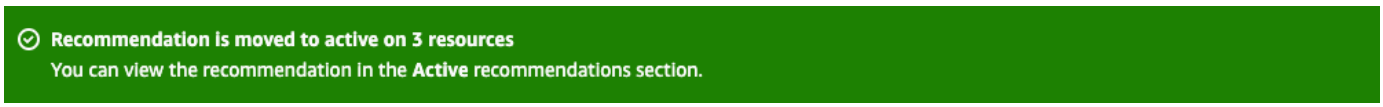
  [レコメンデーション] タブには、選択したリソースの推奨事項とその詳細が表示されます。

3. リストから却下された推奨事項を 1 つ以上選択し、[アクティブに移動] を選択します。



選択した推奨事項を却下からアクティブなステータスに移行すると、バナーに成功または失敗のメッセージが表示されます。

次の例は、成功メッセージを含むバナーを示しています。



次の例は、失敗メッセージを含むバナーを示しています。

CLI

AWS CLI を使用して却下された RDS 推奨事項をアクティブな推奨事項に変更するには

1. コマンドを実行します。`aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`

   出力には、`dismissed` ステータスの推奨事項のリストが表示されます。

2. ステップ 1 でステータスを変更する推奨事項の `recommendationId` を見つけます。

3. ステップ 2 の recommendationId で >aws rds modify-db-recommendation --status active --recommendationId <ID> コマンドを実行して、アクティブな推奨事項を変更します。

RDS API

Amazon RDS API を使用して却下された RDS 推奨事項を変更するには、ModifyDBRecommendation オペレーションを使用します。

Amazon RDS からの推奨事項のリファレンス

Amazon RDS では、リソースが作成または変更されると、リソースの推奨事項が生成されます。次の表に、Amazon RDS からの推奨事項の例を示します。

型	説明	推奨事項	ダウンタイムが必要	追加情報
マグネティックボリュームが使用中です	DB インスタンスはマグネティックストレージを使用しています。ほとんどの DB インスタンスには、マグネティックストレージは推奨されません。別のストレージタイプとして、汎用 (SSD) またはプロビジョンド IOPS を選択してください。	別のストレージタイプとして、汎用 (SSD) またはプロビジョンド IOPS を選択してください。	はい	Amazon EC2 ドキュメントの旧世代のボリューム。
リソースの自動バックアップは無効になっています	自動バックアップは DB インスタンスに対して有効ではありません。DB インスタ	最大 14 日間の保存期間で自動バックアップを有効にします。	はい	自動バックアップの有効化 AWS データベースブログの「 Amazon

型	説明	推奨事項	ダウンタイムが必要	追加情報
	ンスのポイントインタイムリカバリを可能にするため、自動バックアップが推奨されます。			RDS バックアップストレージコストの説明」
エンジンのマイナーバージョンアップグレードが必要です	データベースリソースで最新のマイナー DB エンジンバージョンが実行されていません。最新のマイナーバージョンには、最新のセキュリティ修正プログラムやその他の改善が含まれています。	最新のエンジンバージョンにアップグレードします。	はい	DB インスタンスのエンジンバージョンのアップグレード
拡張モニタリングは無効になっています	データベースリソースでは拡張モニタリングが有効になっていません。拡張モニタリングにより、モニタリングとトラブルシューティングのためのリアルタイムのオペレーティングシステムメトリクスが提供されます。	Enhanced monitoring] を有効にします。	いいえ	拡張モニタリングを使用した OS メトリクスのモニタリング

型	説明	推奨事項	ダウンタイムが必要	追加情報
ストレージの暗号化は無効になっています。	Amazon RDS では、AWS Key Management Service (AWS KMS) で管理しているキーを使用して、すべてのデータベースエンジンの保存時の暗号化をサポートしています。Amazon RDS 暗号化を使用するアクティブな DB インスタンスでは、ストレージに保存されているデータは、自動バックアップ、リードレプリカ、スナップショットのように暗号化されます。 DB インスタンスの作成時に暗号化が有効になっていない場合は、暗号化を有効にする前に、DB インスタンスの復号化されたスナップショットの暗号化されたコピーを作成および復	DB インスタンスの保管中のデータの暗号化を有効にします。	はい	Amazon RDS でのセキュリティ Amazon RDS の DB スナップショットのコピー

型	説明	推奨事項	ダウンタイムが必要	追加情報
	元する必要があります。			
Performance Insights は無効になっている	Performance Insights では、DB インスタンスの負荷をモニタリングし、データベースパフォーマンスの問題の分析と解決をサポートします。Performance Insights を有効にすることをお勧めします。	Performance Insights をオンにします。	いいえ	Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング
DB インスタンスのストレージの自動スケーリングが無効になっている	DB インスタンスのストレージ自動スケーリングが有効になっていません。RDS ストレージの自動スケーリングは、データベースのワークロードが増加したときに、ダウンタイムなしでストレージ容量を自動的にスケーリングします。	指定した最大ストレージしきい値で Amazon RDS ストレージ自動スケーリングを有効にします	いいえ	Amazon RDS ストレージの自動スケーリングによる容量の自動管理

型	説明	推奨事項	ダウンタイムが必要	追加情報
RDS リソースのメジャーバージョンの更新が必須	DB エンジンの、現行メジャーバージョンのデータベースはサポートされません。新しい機能や拡張機能を含む最新のメジャーバージョンにアップグレードすることをお勧めします。	DB エンジンを最新のメジャーバージョンにアップグレードします。	はい	DB インスタンスのエンジンバージョンのアップグレード データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する
RDS リソースのインスタンスクラスの更新が必須	DB インスタンスは、旧世代の DB インスタンスクラスで実行されています。旧世代の DB インスタンスクラスは、コスト、パフォーマンス、またはその両方が向上した DB インスタンスクラスに置き換えられました。DB インスタンスには、新しい世代の DB インスタンスクラスを使用して実行することをお勧めします。	DB インスタンスクラスをアップグレードします。	はい	DB インスタンスクラスでサポートされている DB エンジン

型	説明	推奨事項	ダウンタイムが必要	追加情報
ライセンス付きのサポート終了エンジンエディションを使用する RDS リソース	現在のライセンスサポートを継続するには、メジャーバージョンを Amazon RDS がサポートする最新のエンジンバージョンにアップグレードすることをお勧めします。データベースのエンジンバージョンは、現在のライセンスではサポートされません。	ライセンスモデルを引き続き使用するには、データベースを Amazon RDS でサポートされている最新バージョンにアップグレードすることをお勧めします。	はい	Oracle のメジャーバージョンのアップグレード

型	説明	推奨事項	ダウンタイムが必要	追加情報
DB インスタンスがマルチ AZ 配置を使用していない	マルチ AZ 配置を使用することをお勧めします。マルチ AZ 配置により、DB インスタンスの可用性と耐久性が向上します。	影響を受ける DB インスタンスにマルチ AZ を設定します。	いいえ この変更時にダウンタイムは発生しません。ただし、パフォーマンスに影響する可能性があります。詳細については、An RDS	Amazon RDS マルチ AZ の料金

型	説明	推奨事項	ダウンタイムが必要	追加情報
			のDBインスタンスをマルチAZ配置に変換するを参照してください。	

型	説明	推奨事項	ダウンタイムが必要	追加情報
DB のメモリパラメータがデフォルトと異なる	DB インスタンスのメモリパラメータがデフォルト値と大きく異なります。これらの設定はパフォーマンスに影響が及び、エラーの原因となる可能性があります。 DB インスタンスのカスタムメモリパラメータを、DB パラメータグループのデフォルト値に再設定することをお勧めします。	メモリパラメータをデフォルト値にリセットします。	いいえ	Amazon RDS for MySQL のパラメータを設定するためのベストプラクティス (AWS データベースブログ)

型	説明	推奨事項	ダウンタイムが必要	追加情報
最適値未満を使用する `InnoDB_Change_Buffering` パラメータ	変更バッファリングでは、MySQL DB インスタンスは、セカンダリインデックスを維持するために必要ないくつかの書き込みを延期することができます。この機能は、低速ディスクを使用する環境で有効でした。バッファリング設定を変更することで DB のパフォーマンスはわずかに向上しましたが、クラッシュリカバリの遅延やアップグレード中のシャットダウン時間の増加の原因となりました。MySQL バージョン 8.4 では、デフォルトで `OFF` に設定されます。	DB パラメータグループの `InnoDB_Change_Buffering` パラメータを `1` に設定します。	いいえ	Amazon RDS for MySQL のパラメータを設定するためのベストプラクティス (AWS データベースブログ)

型	説明	推奨事項	ダウンタイムが必要	追加情報
Amazon RDS クエリキャッシュパラメータは有効になっている	変更によってクエリキャッシュの削除が必要になった場合、DB インスタンスは停止しているように見えます。通常ワークロードでは、クエリキャッシュのメリットは得られません。クエリキャッシュは、MySQL 8.0 以降のバージョンから削除されました。query_cache_type パラメータを 0 に設定することをお勧めします。	DB パラメータグループの `query_cache_type` パラメータを `0` に設定します。	はい	Amazon RDS for MySQL のパラメータを設定するためのベストプラクティス (AWS データベースブログ)

型	説明	推奨事項	ダウンタイムが必要	追加情報
log_output パラメータが table に設定されている	log_output が TABLE に設定されている場合、log_output が FILE に設定されている場合よりも多くのストレージが使用されます。ストレージサイズの制限に達しないように、パラメーターを FILE に設定することをお勧めします。MySQL 8.4 以降のバージョンでは、デフォルトで FILE に設定されます。	DB パラメータグループの log_output パラメータを FILE に設定します。	いいえ	MySQL データベースのログファイル

型	説明	推奨事項	ダウンタイムが必要	追加情報
パラメータグループで huge pages が使用されない	Large pages はデータベースのスケーラビリティを高めることができますが、DB インスタンスは Large pages を使用していません。DB インスタンスの DB パラメータグループで、use_large_pages パラメータを ONLY に設定することをお勧めします。	DB パラメータグループの use_large_pages パラメータを ONLY に設定します。	はい	サポートされている RDS for Oracle インスタンスで HugePages をオンにする
autovacuum パラメータがオフになっている	DB インスタンスの自動バキュームパラメータは無効になっています。自動バキュームを無効にすると、テーブルとインデックスが肥大化し、パフォーマンスに影響します。 DB パラメータグループの自動バキュームを有効にすることをお勧めします。	DB パラメータグループの自動バキュームパラメータを有効にしてください。	いいえ	AWS データベースブログの「Amazon RDS for PostgreSQL 環境における自動バキュームについて」

型	説明	推奨事項	ダウンタイムが必要	追加情報
synchronous_commit パラメータがオフになっている	synchronous_commit パラメータを無効にすると、データベースのクラッシュでデータが失われる可能性があります。データベースの耐久性が危険にさらされます。 synchronous_commit パラメータをオンにすることをお勧めします。	DB パラメータグループの synchronous_commit パラメータを有効にします。	はい	AWS データベースブログの「Amazon Aurora PostgreSQL パラメータ: データベースブログの「レプリケーション、セキュリティ、ログ記録」
track_counts パラメータがオフになっている	track_counts パラメータが無効の場合、データベースはデータベースアクティビティ統計を収集しません。自動バキュームでは、これらの統計が正しく機能する必要があります。 track_counts パラメータを 1 に設定することをお勧めします。	track_counts パラメータを 1 に設定します。	いいえ	PostgreSQL のランタイム統計

型	説明	推奨事項	ダウンタイムが必要	追加情報
enable_indexonlyscan パラメータがオフになっている	クエリプランナーまたはオプティマイザーは、インデックスのみのスキャン計画タイプが無効になっている場合は使用できません。 enable_indexonlyscan パラメータ値を 1 に設定することをお勧めします。	enable_indexonlyscan パラメータ値を 1 に設定します。	いいえ	PostgreSQL のプランナーメソッド設定
enable_indexscan パラメータがオフになっている	クエリプランナーまたはオプティマイザーは、インデックスのみのスキャン計画タイプが無効になっている場合は使用できません。 enable_indexscan 値を 1 に設定することをお勧めします。	enable_indexscan パラメータ値を 1 に設定します。	いいえ	PostgreSQL のプランナーメソッド設定

型	説明	推奨事項	ダウンタイムが必要	追加情報
innodb_flush_log_at_trx パラメータがオフになっている	DB インスタンスの innodb_flush_log_at_trx パラメータの値は安全ではありません。このパラメータは、ディスクへのコミット操作の持続性を制御します。 innodb_flush_log_at_trx パラメータを 1 に設定することをお勧めします。	innodb_flush_log_at_trx パラメータ値を 1 に設定します。	いいえ	Amazon RDS for MySQL のパラメータを設定するためのベストプラクティス (AWS データベースブログ)
sync_binlog パラメータがオフになっている	DB インスタンスでトランザクションのコミットが確認される前には、バイナリログのディスクへの同期は実行されません。 sync_binlog パラメータ値を 1 に設定することをお勧めします。	sync_binlog パラメータ値を 1 に設定します。	いいえ	AWS データベースブログのAmazon RDS for MySQL のレプリケーションパラメータを設定するためのベストプラクティス

型	説明	推奨事項	ダウンタイムが必要	追加情報
innodb_stats_persistent パラメータがオフになっている	DB インスタンスは、InnoDB 統計をディスクに保持するように設定されていません。統計が保存されていない場合は、インスタンスが再起動してテーブルにアクセスするたびに再計算されます。これにより、クエリ実行プランにばらつきが生じます。このグローバルパラメータの値はテーブルレベルで変更できます。 innodb_stats_persistent パラメータ値を ON に設定することをお勧めします。	innodb_stats_persistent パラメータ値を ON に設定します。	いいえ	Amazon RDS for MySQL のパラメータを設定するためのベストプラクティス (AWS データベースブログ)

型	説明	推奨事項	ダウンタイムが必要	追加情報
innodb_open_files パラメータが低い	innodb_open_files パラメータは、InnoDB が一度に開くことができるファイル数を制御します。InnoDB は、mysqld の実行時にすべてのログファイルとシステムテーブルスペースファイルを開きます。 お使いの DB インスタンスは、InnoDB が一度に開くことができる最大ファイル数の値が低くなっています。innodb_open_files パラメータを少なくとも 65 に設定することをお勧めします。	innodb_open_files パラメータを最小値の 65 に設定します。	はい	MySQL 用の InnoDB オープンファイル

型	説明	推奨事項	ダウンタイムが必要	追加情報
max_user_connections パラメータが低い	DB インスタンスは、各データベースアカウントの最大同時接続数の値が低くなっています。 max_user_connections パラメータを 5 より大きい数に設定することをお勧めします。	max_user_connections パラメータの値を 5 より大きい数にします。	はい	MySQL のアカウントリソース制限の設定
リードレプリカは書き込み可能モードで開かれている	DB インスタンスには書き込み可能モードのリードレプリカがあり、クライアントからの更新が可能です。 リードレプリカが書き込み可能モードにならないように、read_only パラメータを TrueIfReplica に設定することをお勧めします。	read_only パラメータ値を TrueIfReplica に設定します。	いいえ	AWS データベースブログのAmazon RDS for MySQL のレプリケーションパラメータを設定するためのベストプラクティス

型	説明	推奨事項	ダウンタイムが必要	追加情報
innodb_default_row_format パラメータ設定が安全ではない	DB インスタンスで既知の問題が発生しました: MySQL バージョン 8.0.26 よりも前のバージョンで、row_format を COMPACT または REDUNDANT に設定して作成されたテーブルは、インデックスが 767 バイトを超えるとアクセスできなくなり、回復できなくなります。 innodb_default_row_format パラメータ値を DYNAMIC に設定することをお勧めします。	innodb_default_row_format パラメータ値を DYNAMIC に設定します。	いいえ	MySQL 8.0.26 での変更

型	説明	推奨事項	ダウンタイムが必要	追加情報
general_logging パラメータがオンになっている	DB インスタンスの一般ログ記録が有効になっています。この設定は、データベースの問題のトラブルシューティングに役立ちます。しかし、一般ログ記録を有効にすると、入出力操作の量と割り当てられるストレージ容量が増え、競合やパフォーマンスの低下につながる可能性があります。 一般ログ記録の使用状況の要件を確認してください。general_logging パラメータ値を 0 に設定することをお勧めします。	一般ログ記録の使用状況の要件を確認してください。必須ではない場合は、general_logging パラメーターの値を 0 に設定することをお勧めします。	いいえ	RDS for MySQL データベースログの概要

型	説明	推奨事項	ダウンタイムが必要	追加情報
システムメモリ容量のプロビジョニングが不十分な RDS インスタンス	メモリの使用量を減らすか、メモリの割り当て量の多い DB インスタンスタイプを使用するようにクエリを調整することをお勧めします。インスタンスのメモリが不足すると、データベースのパフォーマンスに影響を及ぼします。	メモリ容量のより高い DB インスタンスを使用する	はい	AWS データベースブログの「Amazon RDS インスタンスの垂直スケーリングと水平スケーリング」 Amazon RDS インスタンスタイプ Amazon RDS の価格設定
システム CPU 容量のプロビジョニングが不十分な RDS インスタンス	より少ないメモリを使用するようにクエリを調整するか、v CPU の割り当て量がより多い DB インスタンスを使用するように DB インスタンスを変更することをお勧めします。DB インスタンスの CPU が少なくなると、データベースのパフォーマンスが低下する可能性があります。	CPU 容量がより多い DB インスタンスを使用する	はい	AWS データベースブログの「Amazon RDS インスタンスの垂直スケーリングと水平スケーリング」 Amazon RDS インスタンスタイプ Amazon RDS の価格設定

型	説明	推奨事項	ダウンタイムが必要	追加情報
RDS リソースは接続プールを正しく利用していません。	Amazon RDS Proxy を有効にして、既存のデータベース接続を効率的にプールして共有することをお勧めします。データベースで既にプロキシを使用している場合は、複数の DB インスタンス間の接続プールと負荷分散を改善するようにプロキシを正しく設定します。RDS Proxy は、接続の枯渇やダウンタイムのリスクを軽減すると同時に、可用性とスケーラビリティを向上させるのに役立ちます。	RDS プロキシを有効にするか、既存のプロキシ設定を変更する	いいえ	AWS データベースブログの「Amazon RDS インスタンスの垂直スケーリングと水平スケーリング」 Amazon RDS Proxy Amazon RDS Proxy の料金

型	説明	推奨事項	ダウンタイムが必要	追加情報
RDS インスタンスが過剰な一時オブジェクトを作成している	ワークロードを調整して過剰な一時オブジェクトが作成されないようにするか、最適化された読み取りをサポートする RDS インスタンスクラスに切り替えることをお勧めします。RDS Optimized Reads は、多数の一時オブジェクトや大きな一時オブジェクトを含むワークロードのデータベースパフォーマンスを向上させます。ワークロードを評価し、インスタンスで RDS Optimized Reads を使用することで、データベースワークロードが改善されるかどうかを判断します。	RDS Optimized Reads で DB インスタンスタイプを使用する	はい	Amazon RDS インスタンスタイプ Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上 Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上 Amazon RDS Optimized Reads による RDS for PostgreSQL のクエリパフォーマンスの向上

型	説明	推奨事項	ダウンタイムが必要	追加情報
システム IOPS 容量のプロビジョニングが不十分な RDS インスタンス	データベースワークロードを調整して IOPS を減らすか、デフォルトの IOPS 制限が高いタイプに DB インスタンスをスケールアップすることをお勧めします。現在の DB インスタンスではプロビジョンド IOPS をサポートできないか、データベースワークロードの IOPS 使用率が高くなっています。	より高い IOPS デフォルト制限を持つ DB インスタンスタイプを使用する	はい	Amazon RDS インスタンスタイプ Amazon RDS DB インスタンスストレージ データベース負荷
RDS インスタンスにプロビジョニング不足の Amazon EBS ボリュームがある	データベースワークロードを調整して IOPS を減らすか、データベースのプロビジョンド IOPS を増やすことをお勧めします。IOPS 使用率がプロビジョンド IOPS に近づくと、データベースのパフォーマンスが低下する可能性があります。	DB インスタンスにより多くの IOPS をプロビジョニングする	はい	Amazon RDS インスタンスタイプ Amazon RDS DB インスタンスストレージ データベース負荷

型	説明	推奨事項	ダウンタイムが必要	追加情報
RDS インスタンスのスループットキャパシティがプロビジョニング不足	データベースワークロードを調整してスループットを減らすか、データベースのプロビジョンドスループットを増やすことをお勧めします。スループット使用率がプロビジョニングされたスループットに近づくと、データベースのパフォーマンスに影響する可能性があります。	DB インスタンスにより多くのスループットをプロビジョニングする	はい	Amazon RDS インスタンスタイプ Amazon RDS DB インスタンスストレージ データベース負荷

型	説明	推奨事項	ダウンタイムが必要	追加情報
RDS インスタンスは EBS I/O のプロビジョニング不足	データベースワークロードを調整して I/O 操作を減らすか、Amazon RDS io2 Block Express ボリュームを使用するように DB インスタンスを変更することをお勧めします。このボリュームは、高パフォーマンス、高スループット、および低レイテンシーを必要とするデータベースワークロード用に設計されています。現在のワークロードでは、データベースは必要な速度で I/O 操作を処理できない可能性があり、パフォーマンスが低下する可能性があります。	RDS インスタンスに Amazon RDS io2 Block Express ボリュームを使用する	いいえ	Amazon RDS DB インスタンスストレージ Amazon RDS の Amazon CloudWatch メトリクス 「Amazon EBS ユーザーガイド」の「プロビジョンド IOPS SSD ボリューム」

Amazon RDS コンソールでのメトリクスの表示

Amazon RDS は Amazon Cloudwatch と統合し、さまざまな RDS DB インスタンスメトリクスを RDS コンソールで表示できるようになりました。の詳細については、「Amazon RDS のメトリクスリファレンス」を参照してください。

DB インスタンス については、次のカテゴリのメトリクスがモニタリングされます。

- CloudWatch – RDS コンソールからアクセスできる RDS の Amazon CloudWatch メトリクスを表示します。これらのメトリクスには、CloudWatch コンソールからアクセスすることもできます。各メトリクスには、特定の期間にわたってモニタリングされたメトリクスを示すグラフが含まれます。CloudWatch メトリクスのリストについては、「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。
- [Enhanced monitoring] (拡張モニタリング) – RDS DB インスタンスで、拡張モニタリングがオンにされたときのオペレーティングシステムメトリクスの概要を表示します。RDS は、拡張モニタリングのメトリクスを Amazon CloudWatch Logs アカウントに配信します。各 OS メトリクスには、特定の期間にわたってモニタリングされたメトリクスを示すグラフが含まれます。概要については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。拡張モニタリングメトリクスのリストについては、「拡張モニタリングの OS メトリクス」を参照してください。
- [OS Process list] (OS プロセスリスト) – DB インスタンスで実行中の各プロセスの詳細を表示します。
- [Performance Insights] – DB インスタンスで Amazon RDS Performance Insights ダッシュボードを開きます。Performance Insights の概要については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。Performance Insights メトリクスのリストについては、「Amazon RDS Performance Insights の Amazon CloudWatch メトリクス」を参照してください。

Amazon RDS の Performance Insights ダッシュボードで、Performance Insights と CloudWatch メトリクスの統合ビューが提供されるようになりました。このビューを使用するには、DB インスタンスの Performance Insights がオンになっている必要があります。[モニタリング] タブまたは、ナビゲーションペインの [Performance Insights] で新しいモニタリングビューを選択できます。このビューを選択する手順については、「Performance Insights ダッシュボードを使用して組み合わせたメトリクスを表示する」を参照してください。

Performance Insights ダッシュボードを使用して組み合わせたメトリクスを表示する

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
> Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
> アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
> CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Amazon RDS の Performance Insights ダッシュボードでは、DB インスタンスに対する Performance Insights メトリクスと CloudWatch メトリクスの統合ビューが提供されます。事前設定されたダッシュボードを使用するか、カスタムダッシュボードを作成できます。事前設定されたダッシュボードには、データベースエンジンのパフォーマンス問題の診断に役立つ最も一般的に使用されるメトリク

スが表示されます。また、分析要件を満たすデータベースエンジンのメトリダッシュボードを作成することもできます。次に、このダッシュボードを AWS アカウント内のそのデータベースエンジンタイプのすべての DB インスタンスに使用します。

[モニタリング] タブまたは、ナビゲーションペインの [Performance Insights] でモニタリングビューを選択できます。

Performance Insights ダッシュボードに組み合わせたメトリクスを表示するには、 DB インスタンス に対して Performance Insights がオンになっている必要があります。Performance Insights をオンにする方法の詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。

以下のセクションでは、Performance Insights メトリクスと CloudWatch メトリクスを表示する方法について説明します。

トピック

- [モニタリング] タブで新しいモニタリングビューを選択する
- Performance Insights ページから新しいモニタリングビューを選択する
- Performance Insights が表示されたカスタムダッシュボードの作成
- Performance Insights で、事前設定されたダッシュボードを選択する

[モニタリング] タブで新しいモニタリングビューを選択する

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までにニーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロッ

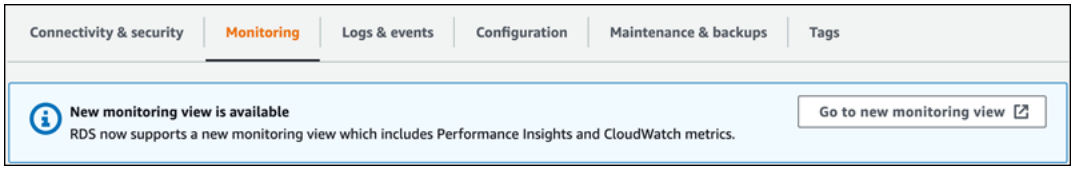
ク診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Amazon RDS コンソールから、新しいモニタリングビューを選択すると、DB インスタンスのパフォーマンスインサイトと CloudWatch メトリクスを表示できます。

[モニタリング] タブで新しいモニタリングビューを選択するには

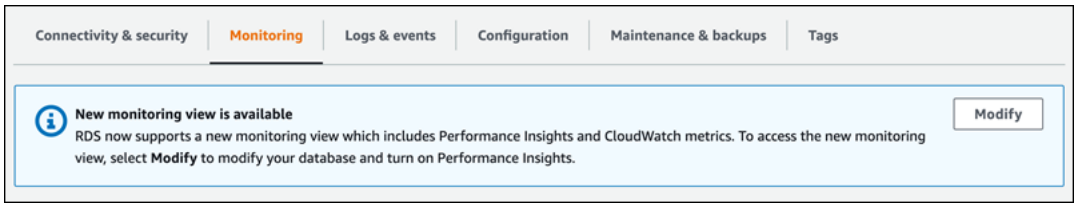
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 左のナビゲーションペインの [データベース] を選択します。
3. モニタリングする DB インスタンスの名前を選択します。
4. 下にスクロールし、[モニタリング] タブを選択します。

   新しいモニタリングビューを選択するオプションを含むバナーが表示されます。次の例は、新しいモニタリングビューを選択するためのバナーを示しています。



5. [新しいモニタリングビューに移動] を選択すると、に対して、Performance Insights と CloudWatch メトリクスが表示された Performance Insights ダッシュボードが開きます。
6. (オプション) DB インスタンスの Performance Insights がオフになっている場合、DB クラスターを変更して Performance Insights をオンにするオプションを含むバナーが表示されます。

次の例は、[モニタリング] タブの DB クラスターを変更するためのバナーを示しています。



[変更] を選択して DB クラスターを変更し、Performance Insights をオンにします。Performance Insights をオンにする方法の詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。

Performance Insights ページから新しいモニタリングビューを選択する

Important

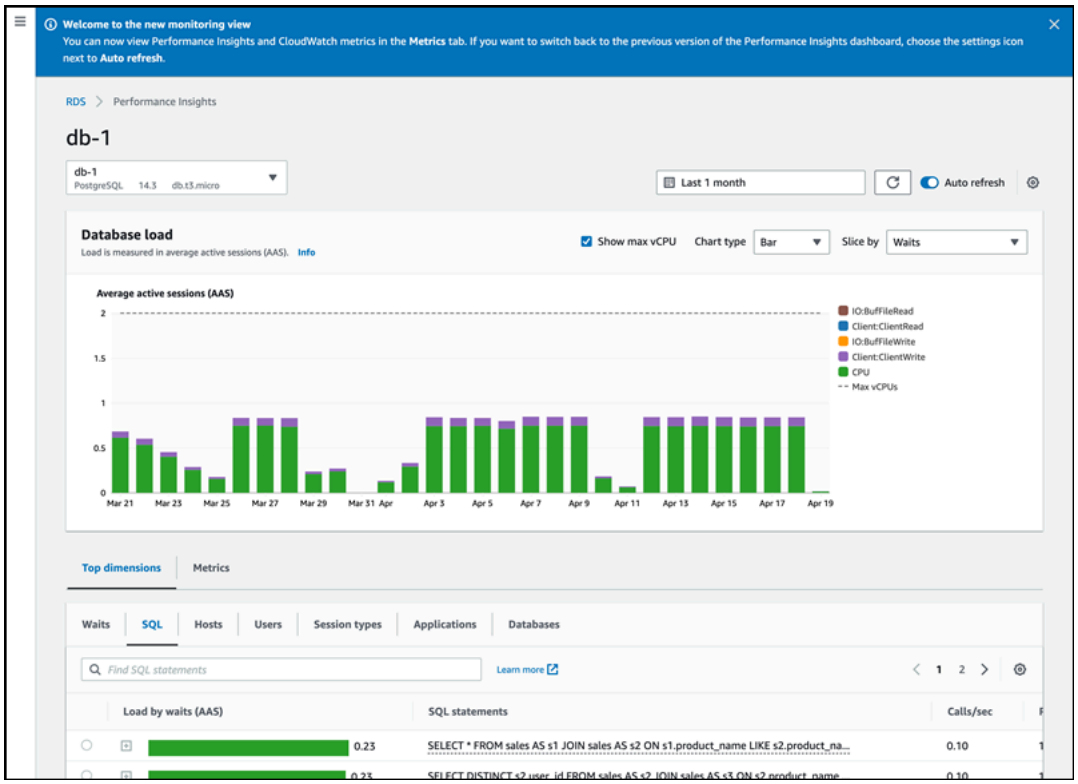
AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うこ

とができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Amazon RDS コンソールから、新しいモニタリングビューを選択すると、DB インスタンスのパフォーマンスインサイトと CloudWatch メトリクスを表示できます。

ナビゲーションペインの Performance Insights を使用して新しいモニタリングビューを選択するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択すると、DB インスタンスの Performance Insights と CloudWatch メトリクスの両方が表示された Performance Insights ダッシュボードが表示されます。



Performance Insights が表示されたカスタムダッシュボードの作成

Important

AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

新しいモニタリングビューでは、分析要件を満たすために必要なメトリクスを含むカスタムダッシュボードを作成できます。

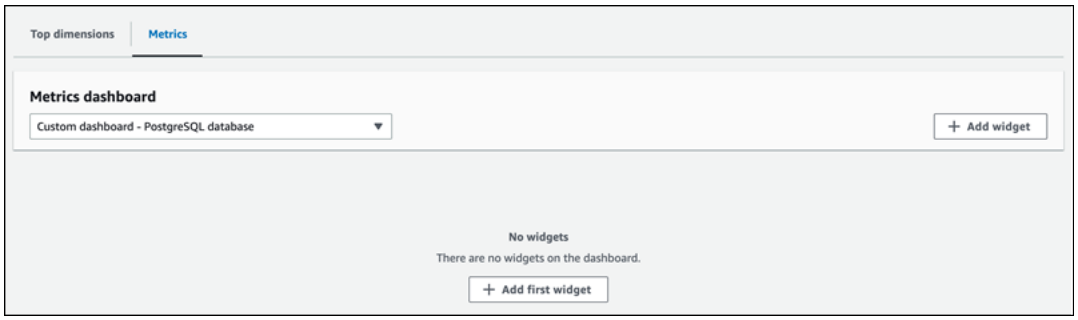
DB インスタンスに対して Performance Insights と CloudWatch メトリクスを選択することで、カスタムダッシュボードを作成できます。次に、このカスタムダッシュボードを AWS アカウント内の同じデータベースエンジンタイプのすべての DB インスタンスに使用します。

> **Note**
>
> カスタマイズされたダッシュボードは、最大 50 のメトリクスをサポートします。

ウィジェット設定メニューを使用して、ダッシュボードを編集または削除したり、ウィジェットウィンドウを移動したり、サイズを変更したりできます。

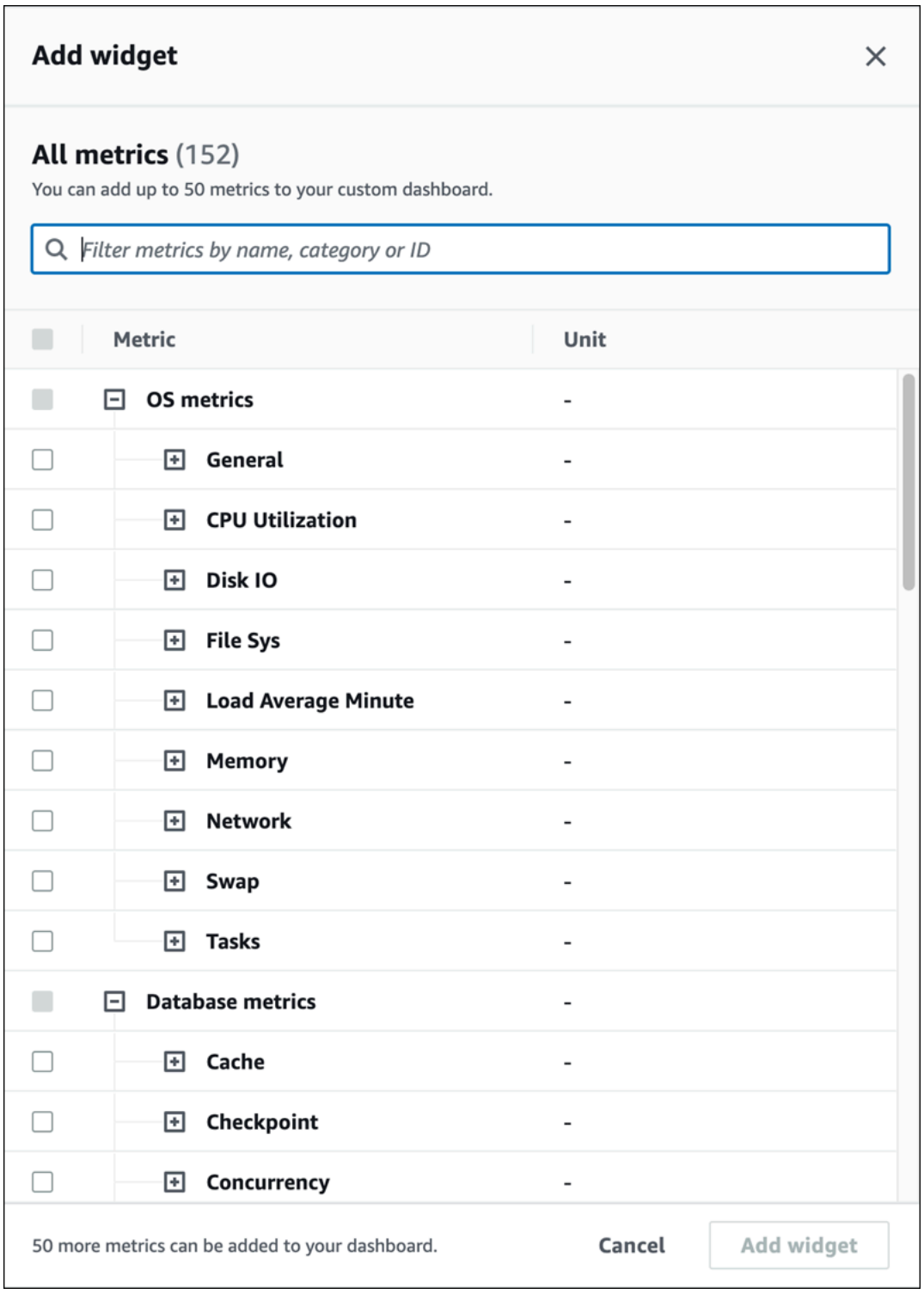
ナビゲーションペインに Performance Insights が表示されたカスタムダッシュボードを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。
4. ウィンドウの [メトリクス] タブが表示されるまで下にスクロールします。
5. ドロップダウンリストから、カスタムダッシュボードを選択します。次の例は、カスタムダッシュボードの作成を示しています。



6. [ウィジェットの追加] を選択して、[ウィジェットの追加] ウィンドウを開きます。使用可能なオペレーティングシステム (OS) メトリクス、データベースメトリクス、および CloudWatch メトリックスをウィンドウで開いて表示できます。

   次の例は、メトリクスが表示された[ウィジェットの追加] ウィンドウを示しています。



7. ダッシュボードで表示するメトリクスを選択してから、[ウィジェットの追加] を選択します。検索フィールドを使用して、特定のメトリクスを検索できます。

   選択したメトリクスがダッシュボードに表示されます。

8. (オプション) ダッシュボードを変更または削除する場合は、ウィジェットの設定アイコンを選択し、メニューで次のいずれかのアクションを選択します。

   - 編集 — ウィンドウ内のメトリクスリストを変更します。ダッシュボードのメトリクスを選択したら、[ウィジェットの更新] を選択します。
   - 削除 — ウィジェットを削除します。確認ウィンドウで、[削除] を選択します。

Performance Insights で、事前設定されたダッシュボードを選択する

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
> Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
> アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
> CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

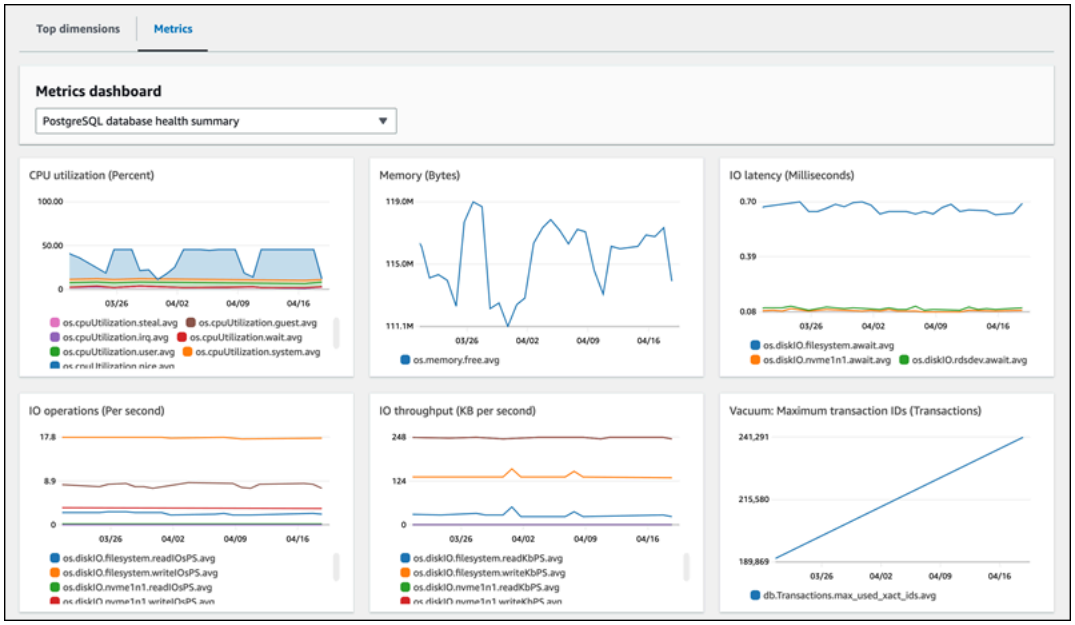
事前設定されたダッシュボードを使用して、最も一般的に使用されるメトリクスを表示できます。このダッシュボードは、データベースエンジンのパフォーマンスの問題を診断し、平均復旧時間を数時間から数分に短縮するのに役立ちます。

> **Note**
>
> このダッシュボードは編集できません。

ナビゲーションペインの Performance Insights で、事前設定されたダッシュボードを選択するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。
4. ウィンドウの [メトリクス] タブが表示されるまで下にスクロールします
5. ドロップダウンリストから、事前設定されたダッシュボードを選択します。

   ダッシュボードで DB インスタンスのメトリクスを表示できます。次の例は、事前設定されたメトリクスダッシュボードを示しています。



Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング

Amazon CloudWatch はメトリクスリポジトリです。リポジトリは、Amazon RDS から raw データを収集し、リアルタイムに近い読み取り可能なメトリクスに加工することができます。CloudWatch に送信される Amazon RDS メトリクスの詳細なリストについては、「Amazon RDS のメトリクスリファレンス」を参照してください。

大規模なデータベースのパフォーマンスを分析およびトラブルシューティングするには、CloudWatch Database InsightsCloudWatch を使用します。

トピック

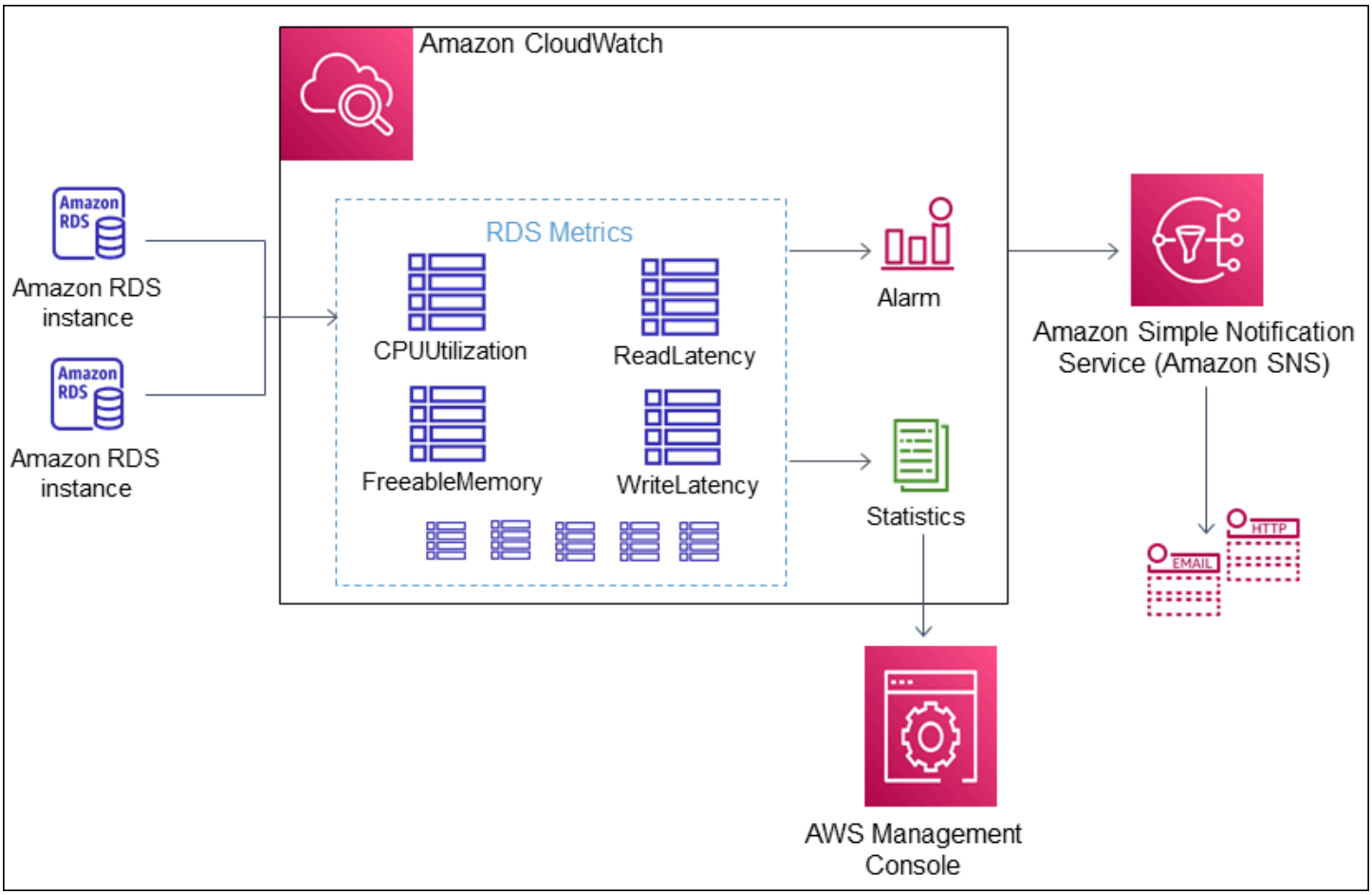
- Amazon RDS および Amazon CloudWatch の概要
- CloudWatch コンソールおよび AWS CLI での DB インスタンスメトリクスの表示
- Performance Insights メトリクスの CloudWatch へのエクスポート
- Amazon RDS をモニタリングするための CloudWatch アラームの作成
- チュートリアル: Amazon RDS 向けマルチ AZ DB クラスターレプリカラグ用の Amazon CloudWatch アラームの作成

Amazon RDS および Amazon CloudWatch の概要

デフォルトでは、Amazon RDS はメトリクスデータを 1 分間隔で CloudWatch に自動的に送信します。例えば、`CPUUtilization` メトリクスは、DB インスタンスの CPU 使用率のパーセンテージを時間の経過とともに記録します。期間が 60 秒 (1 分) のデータポイントは、15 日間使用できます。これにより、履歴情報にアクセスし、ウェブアプリケーションやサービスのパフォーマンスを確認できます。

Performance Insights メトリクスダッシュボードを Amazon RDS から Amazon CloudWatch にエクスポートできるようになりました。事前設定またはカスタマイズされたメトリクスダッシュボードを新しいダッシュボードとしてエクスポートするか、それらを既存の CloudWatch ダッシュボードに追加できます。エクスポートしたメトリクスは CloudWatch コンソールに表示できます。Performance Insights メトリクスダッシュボードを CloudWatch にエクスポートする方法の詳細については、「Performance Insights メトリクスの CloudWatch へのエクスポート」を参照してください。

次の図に示すように、CloudWatch メトリクスのアラームを設定できます。例えば、インスタンスの CPU 使用率が 70% を超えたときに通知するアラームを作成できます。Amazon Simple Notification Service を設定して、しきい値が過ぎたときにメールを送信できます。



Amazon RDS は、次のタイプのメトリクスを Amazon CloudWatch に発行します。

- RDS DB インスタンスのメトリクス

  これらのメトリクスの表については、「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。
- Performance Insights メトリクス

  これらのメトリクスの表については、「Amazon RDS Performance Insights の Amazon CloudWatch メトリクス」および「Performance Insights カウンターメトリクス」を参照してください。
- 拡張モニタリングメトリクス (Amazon CloudWatch Logs に公開)

  これらのメトリクスの表については、「拡張モニタリングの OS メトリクス」を参照してください。
- AWS アカウント の Amazon RDS サービスクォータの使用状況メトリクス

  これらのメトリクスの表については、「Amazon RDS の Amazon CloudWatch 使用状況メトリクス」を参照してください。Amazon RDS のクォータの詳細については、Amazon RDS のクォータと制約 を参照してください。

CloudWatch の詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch とは」を参照してください。CloudWatch メトリクスの保持の詳細については、「メトリクスの保持」を参照してください。

CloudWatch コンソールおよび AWS CLI での DB インスタンスメトリクスの表示

CloudWatch を使用して DB インスタンスのメトリクスを表示する方法の詳細を次に示します。CloudWatch Logs を使用して DB インスタンスのオペレーティングシステムのメトリクスをリアルタイムでモニタリングする方法については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

Amazon RDS リソースを使用する場合、Amazon RDS が 1 分ごとにメトリクスとディメンションを Amazon CloudWatch に送信します。

Performance Insights メトリクスダッシュボードを Amazon RDS から Amazon CloudWatch にエクスポートし、これらのメトリクスを CloudWatch コンソールで表示できるようになりまし

た。Performance Insights メトリクスダッシュボードを CloudWatch にエクスポートする方法の詳細については、「Performance Insights メトリクスの CloudWatch へのエクスポート」を参照してください。

CloudWatch コンソールや CLI で Amazon RDS のメトリクスを表示する場合は、次の手順に従います。

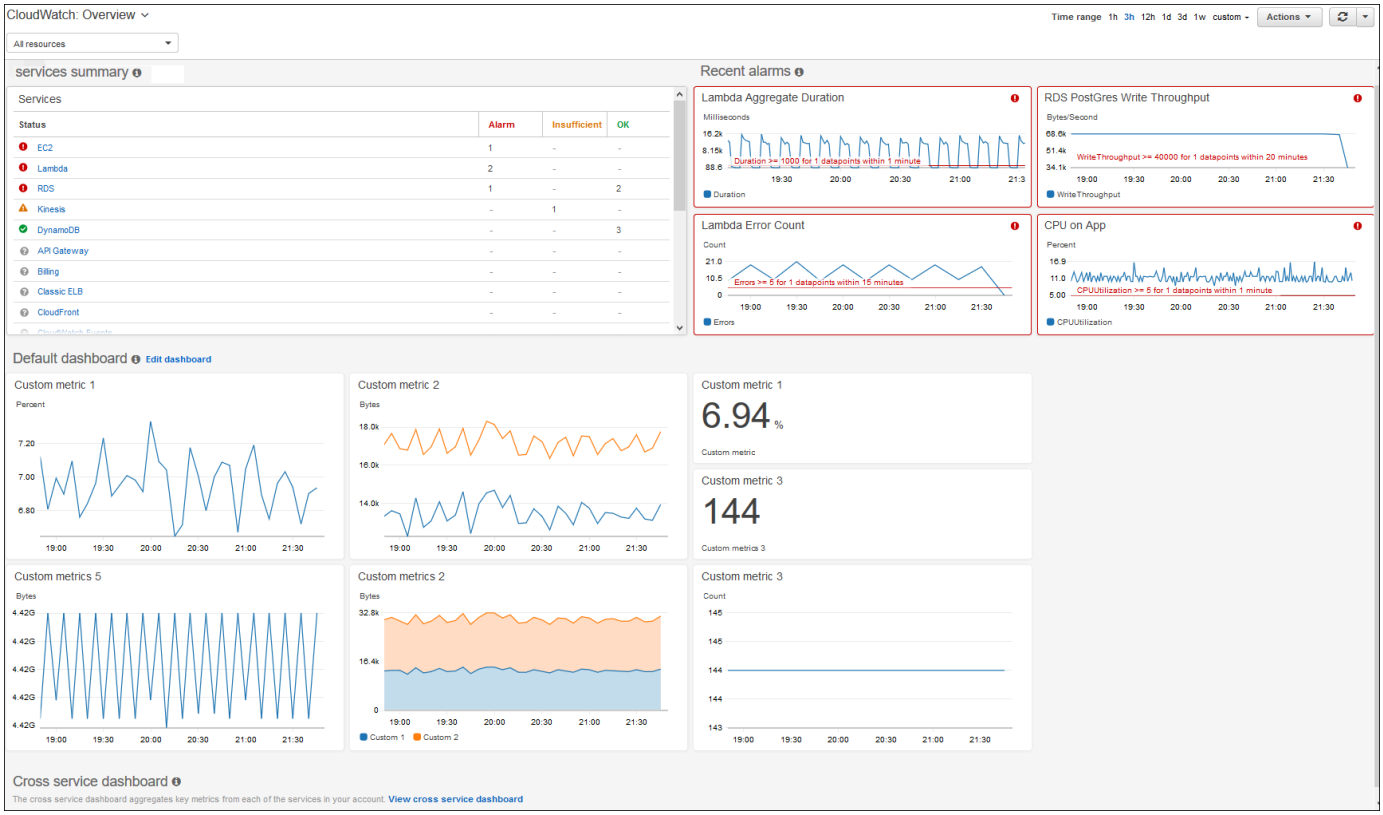
コンソール

Amazon CloudWatch コンソールを使用してメトリクスを表示するには

メトリクスはまずサービスの名前空間ごとにグループ化され、次に各名前空間内のさまざまなディメンションの組み合わせごとにグループ化されます。

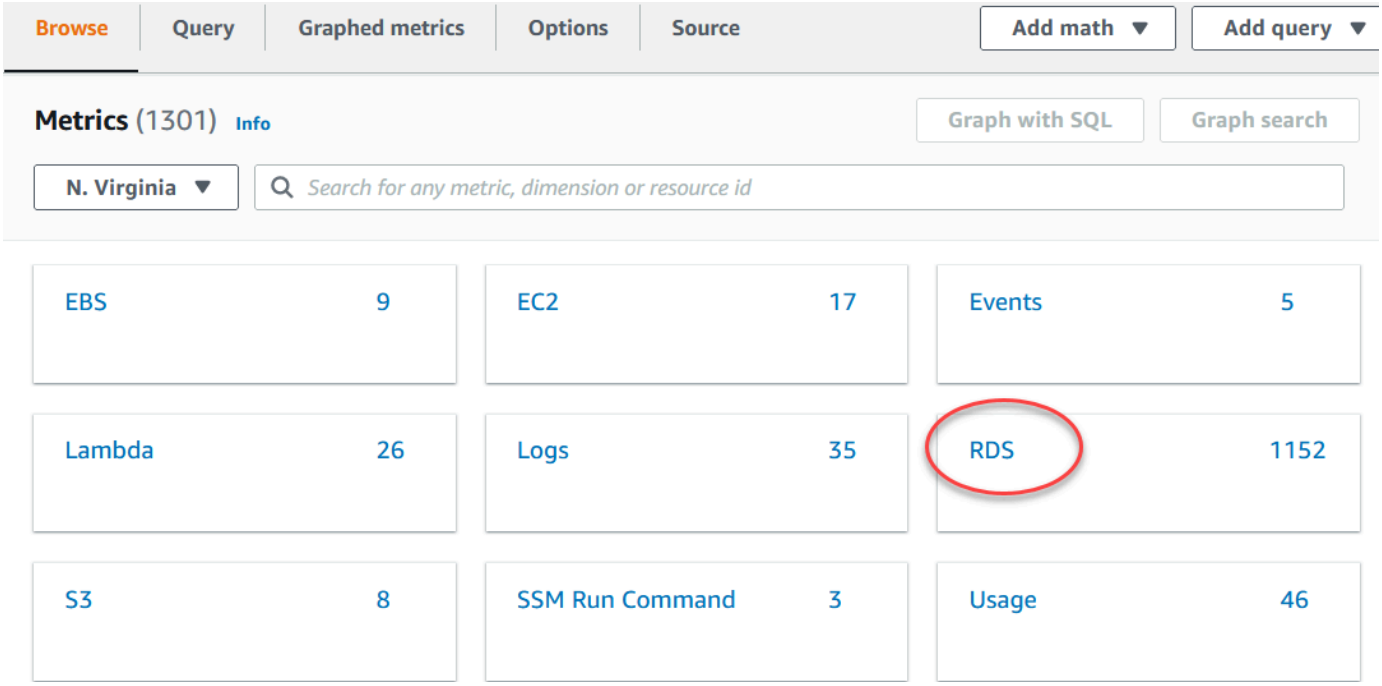
1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。

   CloudWatch 概要のホームページが表示されます。



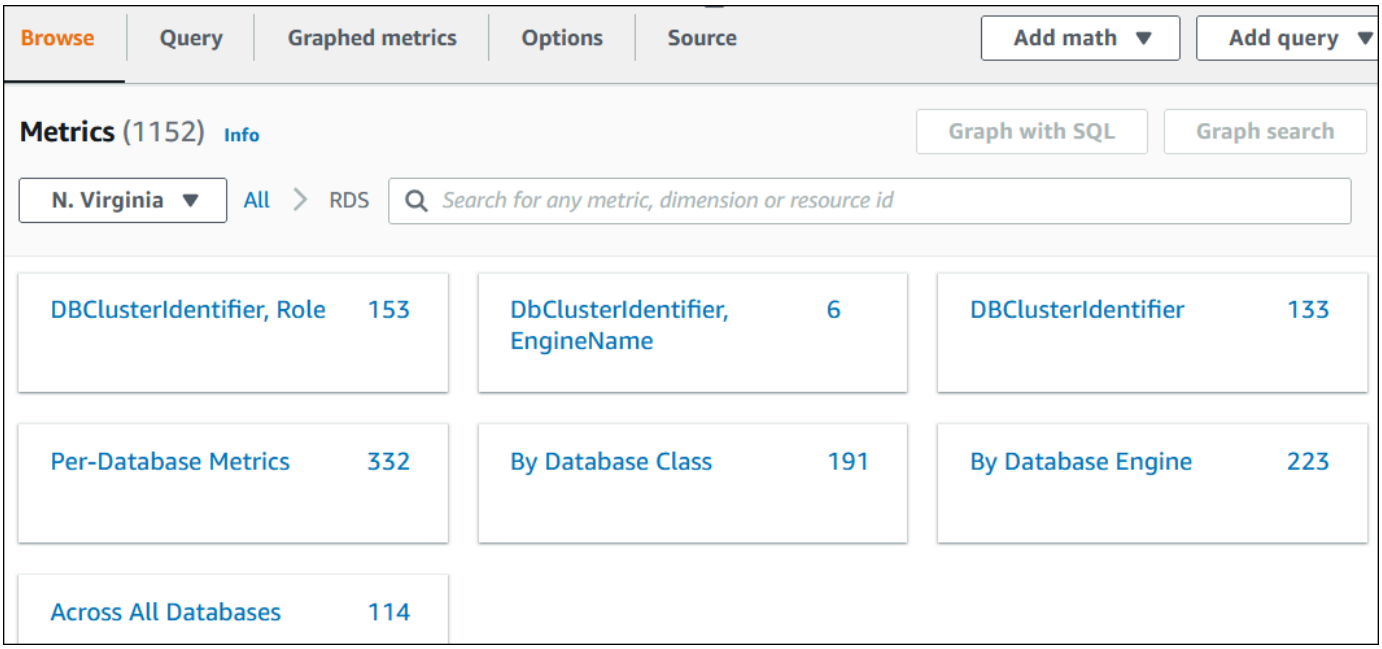
2. 必要に応じて AWS リージョン を変更します。ナビゲーションバーから、AWS リソースがある AWS リージョン リージョンを選択します。詳細については、「リージョンとエンドポイント」を参照してください。

3. ナビゲーションペインで、[Metrics] (メトリクス)、[All metrics] (すべてのメトリクス) の順に選択します。

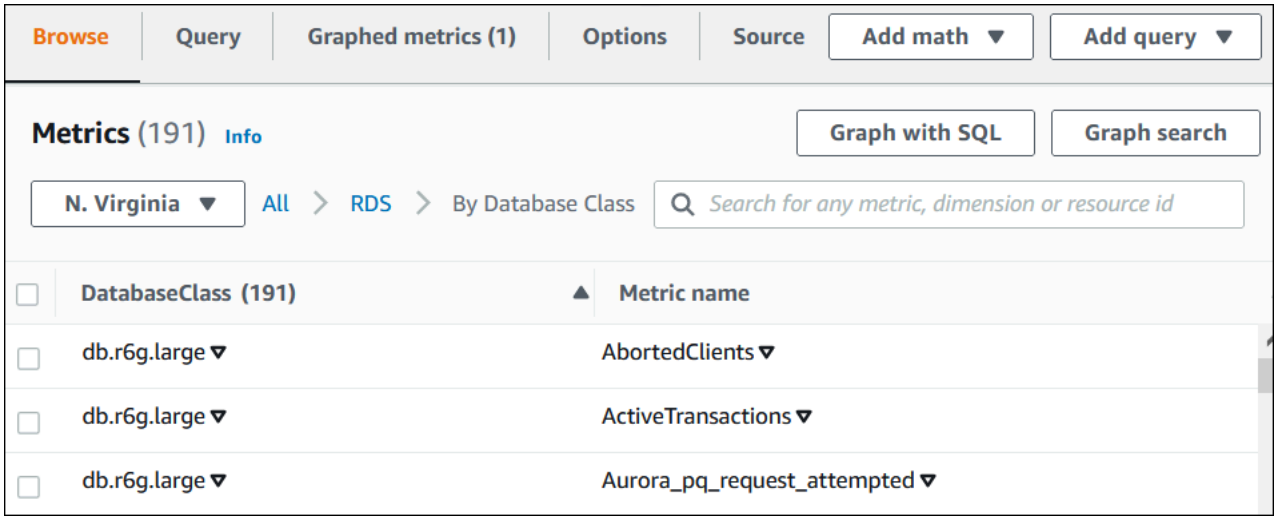


4. 下にスクロールし、RDS メトリクス名前空間を選択します。

   ページに Amazon RDS ディメンションが表示されます。これらのディメンションの詳細については、「Amazon RDS の Amazon CloudWatch ディメンション」を参照してください。



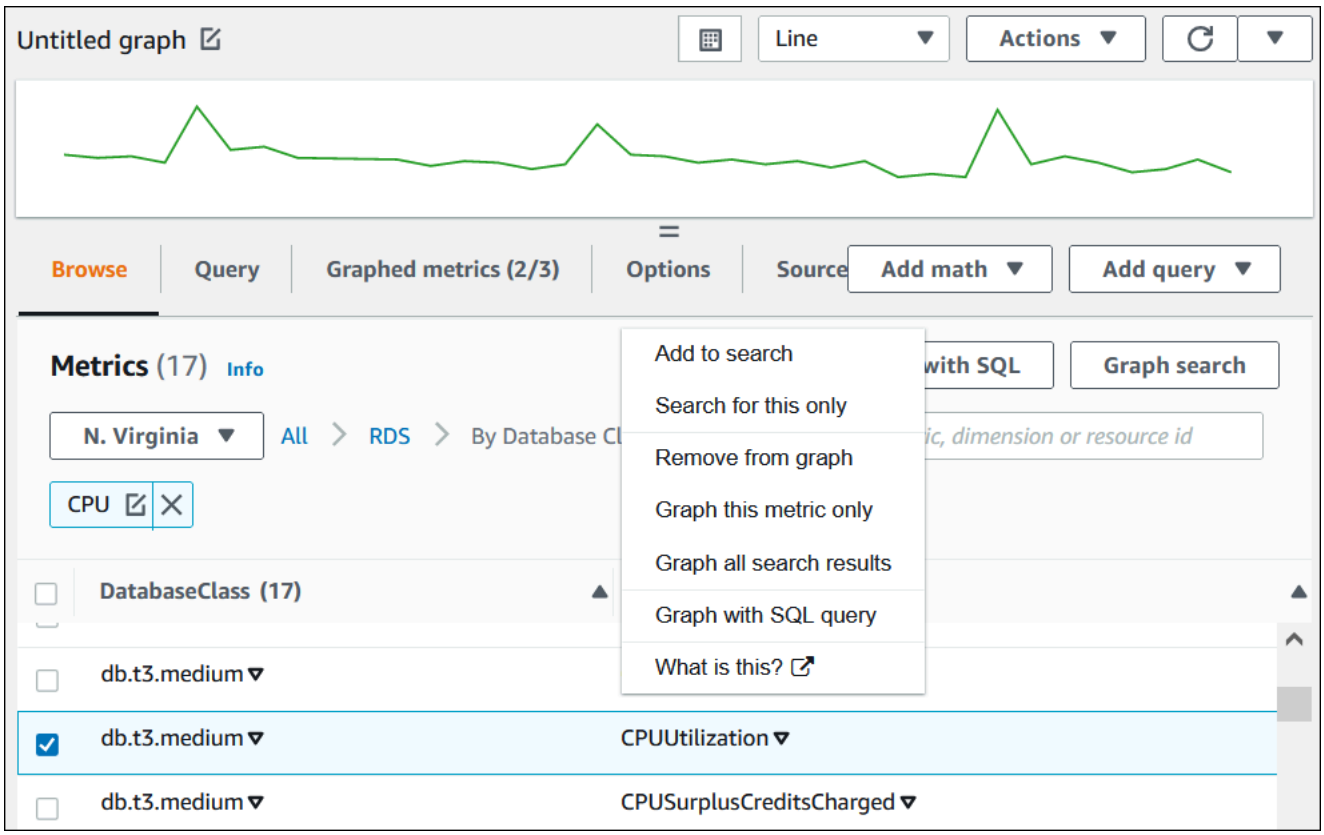
5. メトリクスディメンションを選択します。例えば [データベースクラス別] を選択します。



6. 次のアクションのいずれかを実行します。

   - メトリクスを並べ替えるには、列見出しを使用します。
   - メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。
   - リソースでフィルタするには、リソース ID を選択し、[Add to search] (検索に追加) を選択します。
   - メトリクスでフィルターするには、メトリクス名を選択し、[Add to search] (検索に追加) を選択します。

   次の例では、db.t3.medium クラスをフィルタリングし、CPUUtilization メトリクスをグラフ化します。



AWS CLI

AWS CLI を使用してメトリクスの情報を取得するには、CloudWatch の `list-metrics` コマンドを使用します。次の例では、AWS/RDS 名前空間にすべてのメトリクスがリストされています。

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

メトリクスデータを取得するには、`get-metric-data` コマンドを使用します。

次の例では、特定の 24 時間において 5 分の精度でインスタンス my-instance のCPUUtilization 統計情報を取得します。

次の内容を含む JSON ファイル CPU_metric.json を作成します。

```
{
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
```

```
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Linux、macOS、Unix の場合:

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Windows の場合:

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

出力例は次のとおりです。

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
```

```
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

詳細については、「Amazon CloudWatch ユーザーガイド」の「メトリクスの統計の取得」を参照してください。

Performance Insights メトリクスの CloudWatch へのエクスポート

Performance Insights では、DB インスタンスの事前設定済みまたはカスタムメトリクスダッシュボードを Amazon CloudWatch にエクスポートできます。メトリクスダッシュボードを新しいダッシュボードとしてエクスポートするか、それらを既存の CloudWatch ダッシュボードに追加できます。ダッシュボードを既存の CloudWatch ダッシュボードに追加することを選択した場合、ヘッダーラベルを作成して、メトリクスが CloudWatch ダッシュボードの個別のセクションに表示されるようにすることができます。

エクスポートしたメトリクスダッシュボードは、CloudWatch コンソールで表示できます。Performance Insights メトリクスダッシュボードをエクスポートした後に、新しいメトリクスを追加した場合、CloudWatch コンソールに新しいメトリクスを表示するには、このダッシュボードを再度エクスポートする必要があります。

Performance Insights ダッシュボードでメトリクスウィジェットを選択し、CloudWatch コンソールでメトリクスデータを表示することもできます。

CloudWatch コンソールでメトリクスの表示する詳細については、「CloudWatch コンソールおよび AWS CLI での DB インスタンスメトリクスの表示」を参照してください。

以下のセクションでは、Performance Insights メトリクスを新規または既存のダッシュボードとして CloudWatch にエクスポートし、CloudWatch で Performance Insights メトリクスを表示します。

トピック

- Performance Insights メトリクスを新しいダッシュボードとして CloudWatch にエクスポート
- 既存の CloudWatch ダッシュボードへの Performance Insights メトリクスの追加
- CloudWatch での Performance Insights メトリクスウィジェットの表示

Performance Insights メトリクスを新しいダッシュボードとして CloudWatch にエクスポート

Performance Insights ダッシュボードから事前設定済みまたはカスタムのメトリクスダッシュボードを選択し、新しいダッシュボードとして CloudWatch にエクスポートします。エクスポートしたダッシュボードは、CloudWatch コンソールで表示できます。

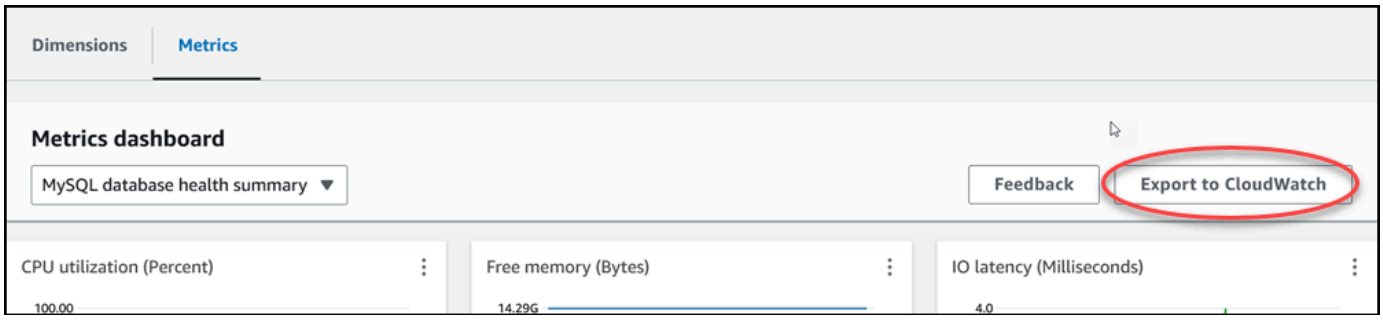
Performance Insights メトリクスダッシュボードを新しいダッシュボードとして CloudWatch にエクスポートするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

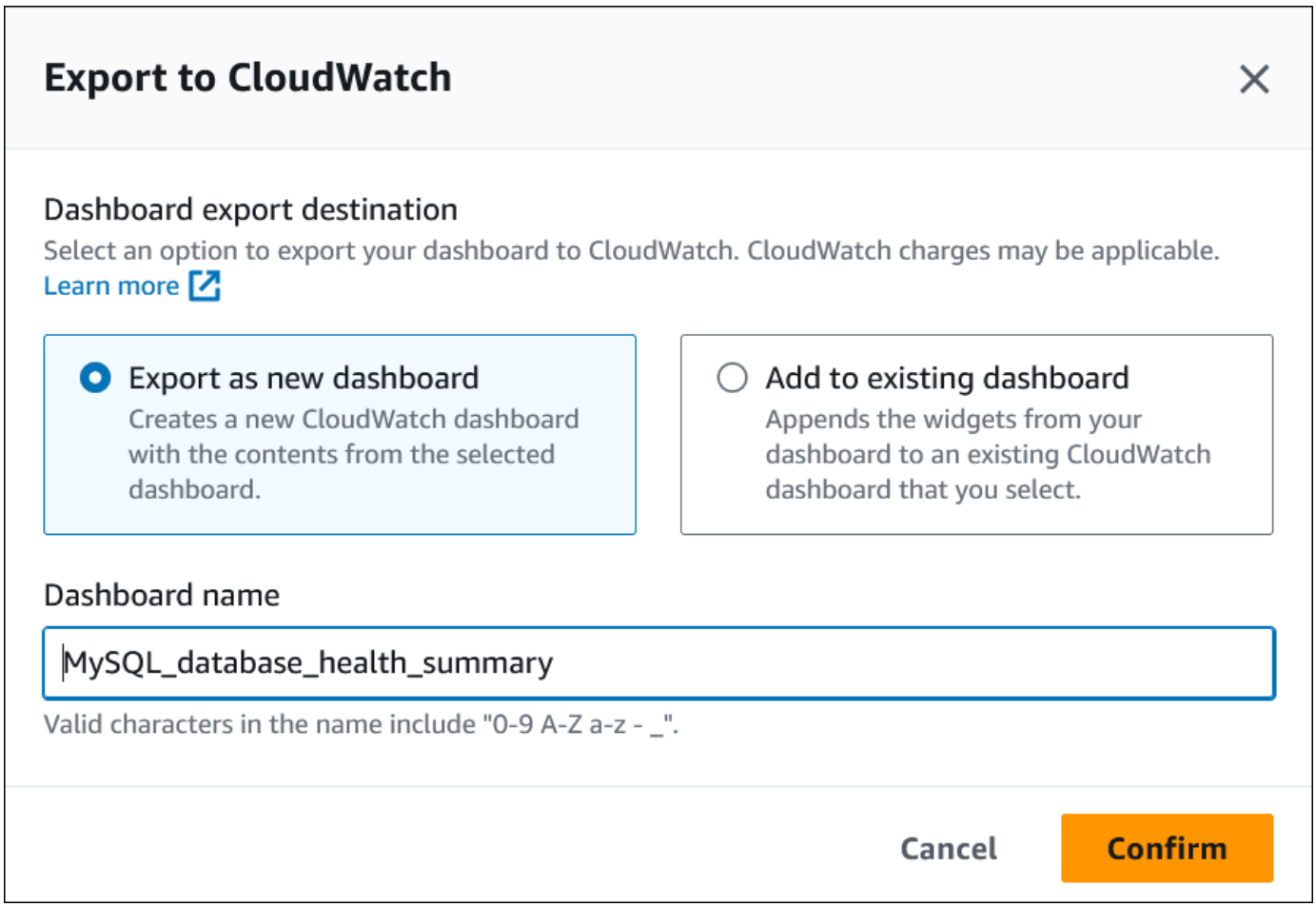
   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. 下にスクロールして [メトリクス] を選択します。

   デフォルトでは、Performance Insights メトリクスで、事前設定されたダッシュボードが表示されます。
5. 事前設定されたダッシュボードまたはカスタムダッシュボードを選択してから、[CloudWatch にエクスポート] を選択します。

   [CloudWatch にエクスポート] ウィンドウが表示されます。

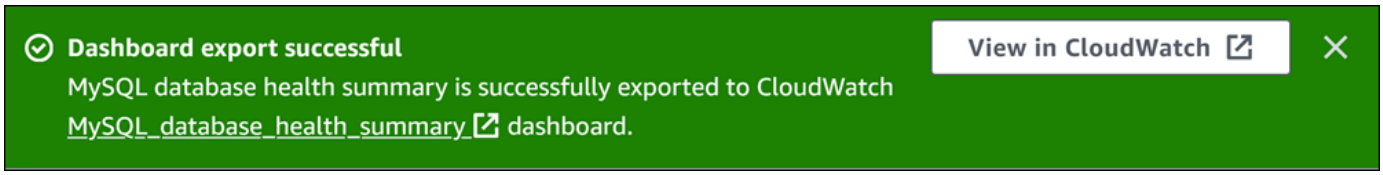


6. [新しいダッシュボードとしてエクスポート] を選択します。



7. [ダッシュボード名] フィールドに新しいダッシュボードの名前を入力し、[確認] を選択します。

   ダッシュボードのエクスポートが成功すると、バナーにメッセージが表示されます。



8. リンクまたはバナーの [CloudWatch で表示] を選択すると、CloudWatch コンソールにメトリクスダッシュボードが表示されます。

既存の CloudWatch ダッシュボードへの Performance Insights メトリクスの追加

事前設定済みまたはカスタムのメトリクスダッシュボードを既存の CloudWatch ダッシュボードに追加します。メトリクスダッシュボードにラベルを追加して、CloudWatch ダッシュボードの個別のセクションに表示することができます。

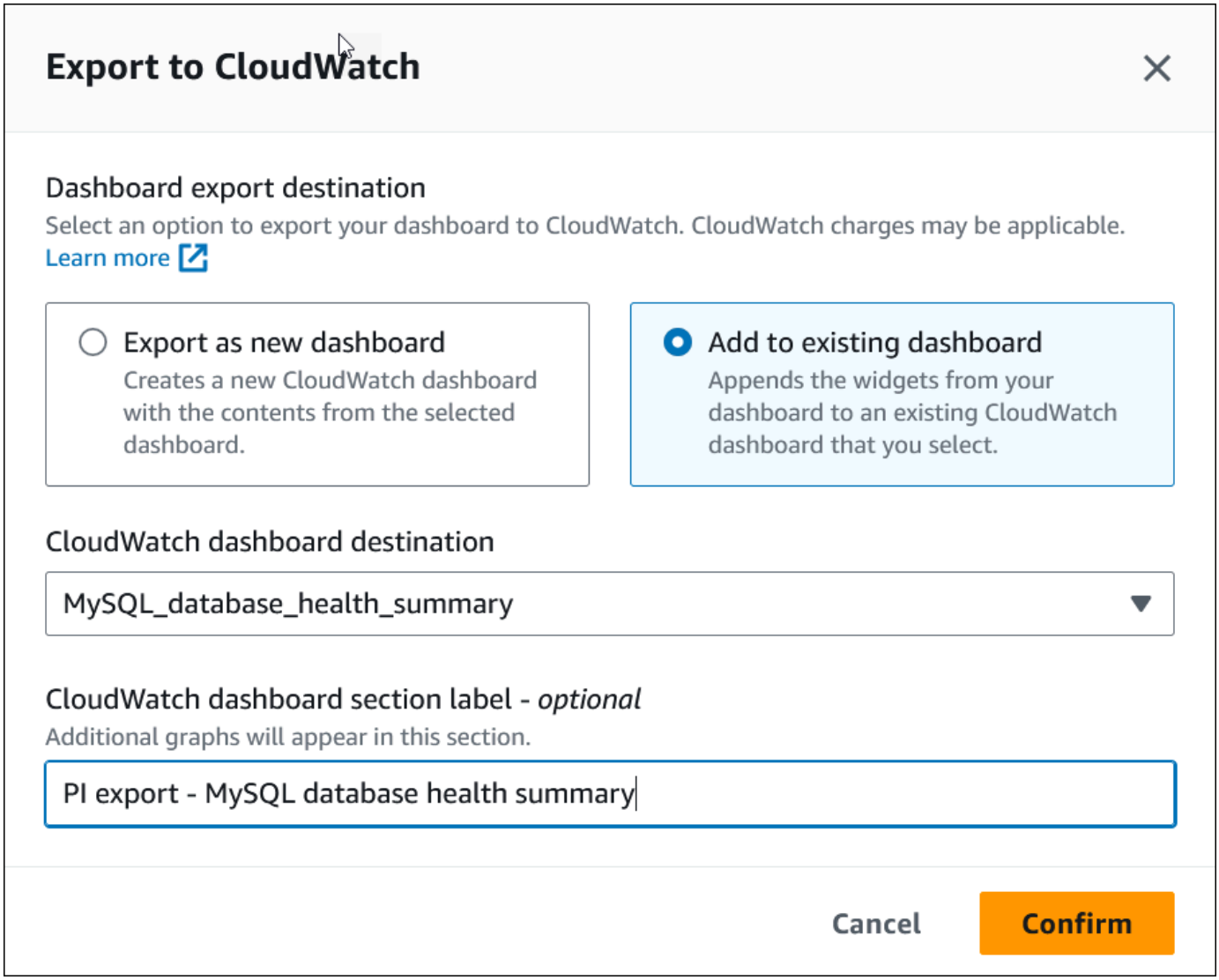
メトリクスを既存の CloudWatch ダッシュボードにエクスポートするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. 下にスクロールして [メトリクス] を選択します。

   デフォルトでは、Performance Insights メトリクスで、事前設定されたダッシュボードが表示されます。
5. 事前設定されたダッシュボードまたはカスタムダッシュボードを選択し、[CloudWatch にエクスポート] を選択します。

   [CloudWatch にエクスポート] ウィンドウが表示されます。
6. [既存のダッシュボードに追加] を選択します。



7. ダッシュボードの送信先とラベルを指定し、[確認] を選択します。

   - [CloudWatch ダッシュボードの送信先] - 既存の CloudWatch ダッシュボードを選択します。
   - [CloudWatch ダッシュボードセクションラベル - オプション] - CloudWatch ダッシュボードのこのセクションに表示する Performance Insights メトリクスの名前を入力します。

   ダッシュボードのエクスポートが成功すると、バナーにメッセージが表示されます。

8. リンクまたはバナーの [CloudWatch で表示] を選択すると、CloudWatch コンソールにメトリクスダッシュボードが表示されます。

CloudWatch での Performance Insights メトリクスウィジェットの表示

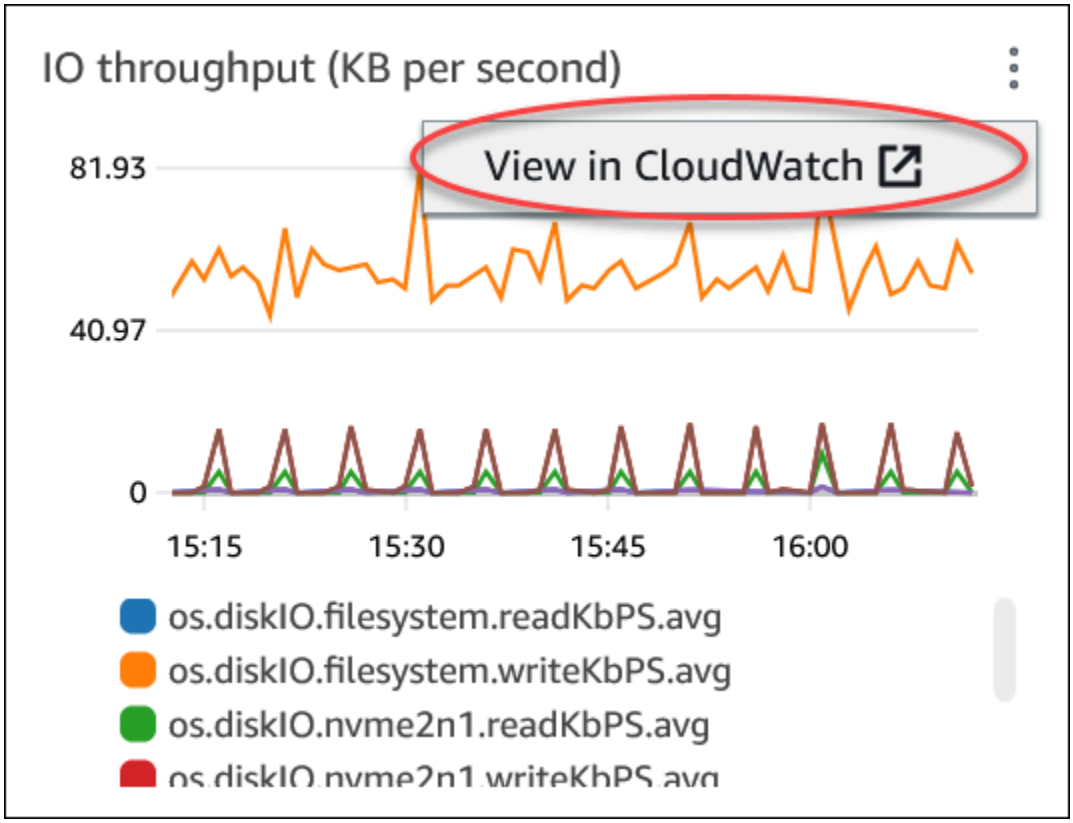
Amazon RDS Performance Insights ダッシュボードで Performance Insights メトリクスウィジェットを選択し、CloudWatch コンソールでメトリクススデータを表示します。

メトリクスウィジェットをエクスポートして CloudWatch コンソールでメトリクスデータを表示するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. [メトリクス] までスクロールします。

   デフォルトでは、Performance Insights メトリクスで、事前設定されたダッシュボードが表示されます。
5. メトリクスウィジェットを選択し、メニューで [CloudWatch で表示] を選択します。



   メトリクスデータは CloudWatch コンソールに表示されます。

Amazon RDS をモニタリングするための CloudWatch アラームの作成

アラームの状態が変わったら、Amazon SNS メッセージを送信する Amazon CloudWatch のアラームを作成することができます。1 つのアラームで、指定した期間中、1 つのメトリクスをモニタリングします。アラームは、指定された複数の期間にわたるしきい値に関連するメトリクスの値に基づいて、1 つ以上のアクションを実行することもできます。アクションは、Amazon SNS トピックまたは Amazon EC2 Auto Scaling ポリシーに送信される通知です。

アラームは、持続している状態変化に対してのみアクションを呼び出します。CloudWatch アラームは、特定の状態にあるというだけの理由ではアクションを呼び出しません。状態が変わって、変わった状態が指定期間にわたって維持される必要があります。

CloudWatch コンソールの DB_PERF_INSIGHTS メトリクス数学関数を使用して Amazon RDS にクエリを実行し、Performance Insights カウンターメトリクスを取得できます。DB_PERF_INSIGHTS 関数には、1 分未満の間隔での DBLoad メトリクスも含まれます。これらのメトリクスに基づいた CloudWatch アラームを設定することができます。

アラームの作成方法の詳細については、「AWS データベースから Performance Insights カウンターメトリクスのアラームを作成する」を参照してください。

AWS CLI を使用してアラームを設定するには

- `put-metric-alarm` を呼び出します。詳細については、「AWS CLI コマンドリファレンス」を参照してください。

CloudWatch API を使用してアラームを設定するには

- を呼び出します。`PutMetricAlarm`詳細については、Amazon CloudWatch API リファレンスを参照してください。

Amazon SNS トピックの設定およびアラームの作成の詳細については、「Amazon CloudWatch アラームの使用」を参照してください。

チュートリアル: Amazon RDS 向けマルチ AZ DB クラスターレプリカラグ用の Amazon CloudWatch アラームの作成

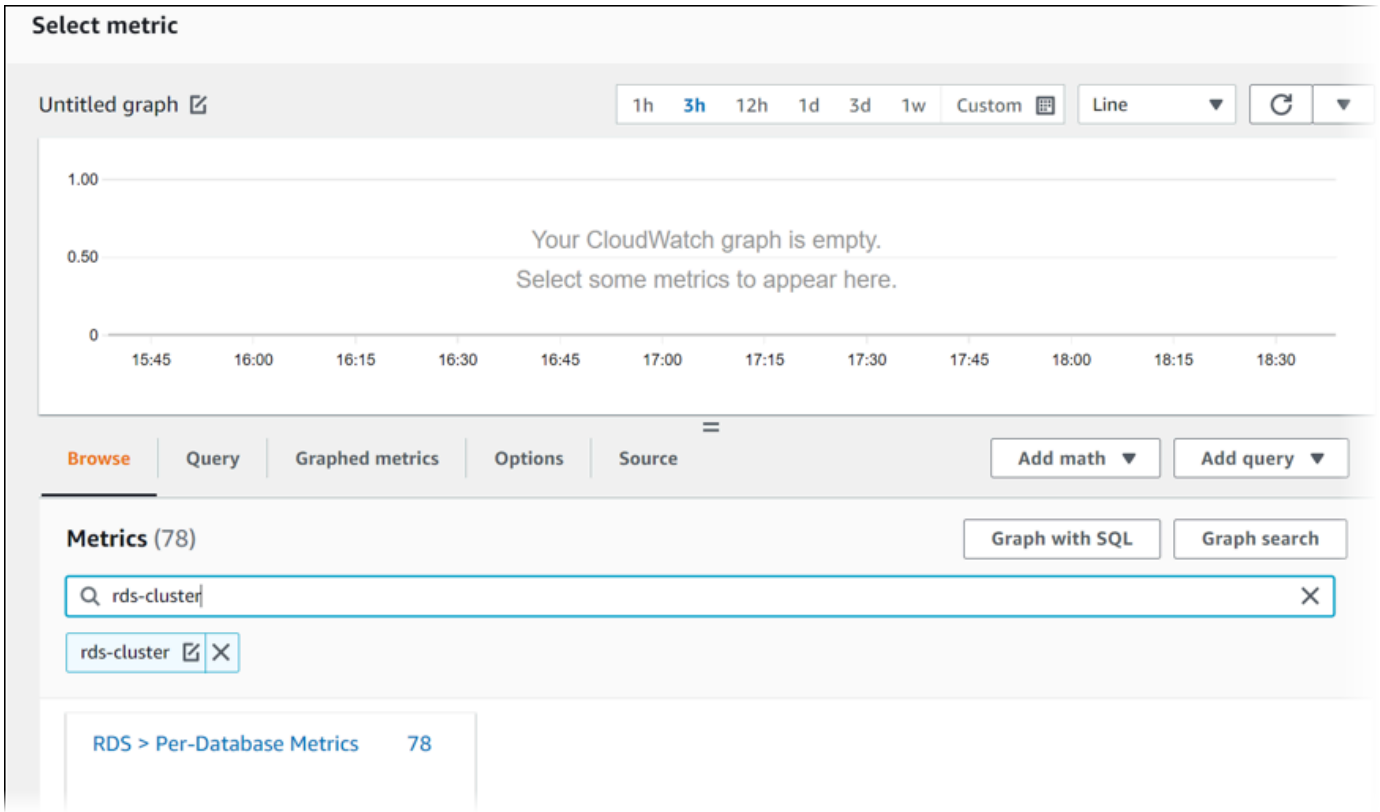
マルチ AZ DB クラスターのレプリカ遅延がしきい値を超えたときに Amazon SNS メッセージを送信する Amazon CloudWatch アラームを作成できます。1 つのアラームで、指定した期間

中、ReplicaLag メトリクスを監視します。アクションは、Amazon SNS トピックまたは Amazon EC2 Auto Scaling ポリシーに送信される通知です。

マルチ AZ DB クラスターレプリカラグ用の CloudWatch アラームを設定するには

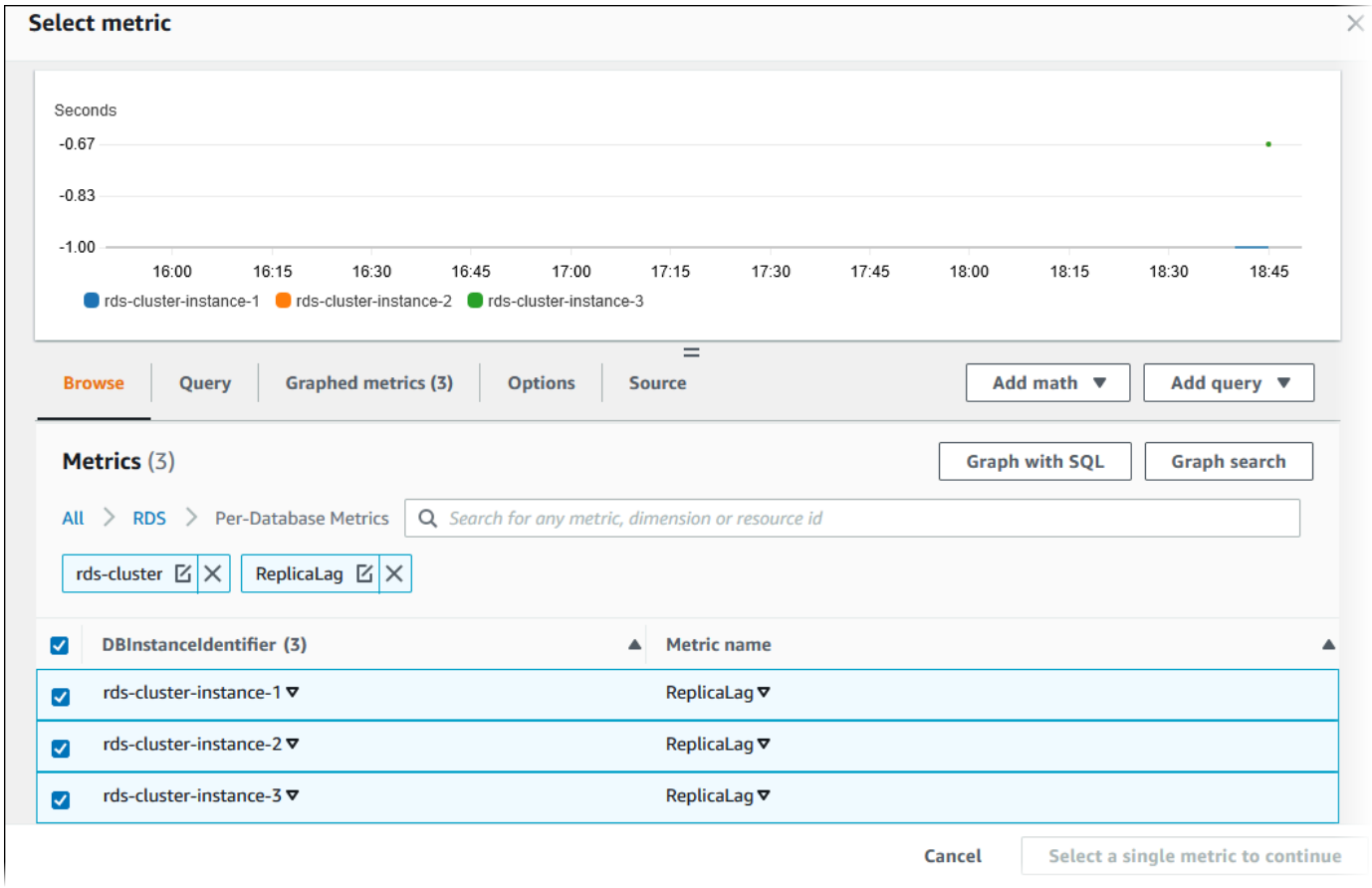
1. AWS マネジメントコンソール にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[アラーム]、[すべてのアラーム] の順に選択します。
3. [アラームの作成] を選択します。
4. [Specify metric and conditions] (メトリクスと条件の指定) ページで、[Select metric] (メトリクスの選択) を選択してください。
5. 検索ボックスに、マルチ AZ DB クラスターの名前を入力し、Enter キーを押します。

   次の図は、rds-cluster という名前のマルチ AZ DB クラスターが入力された [Select metric] (メトリクスの選択) ページを示しています。



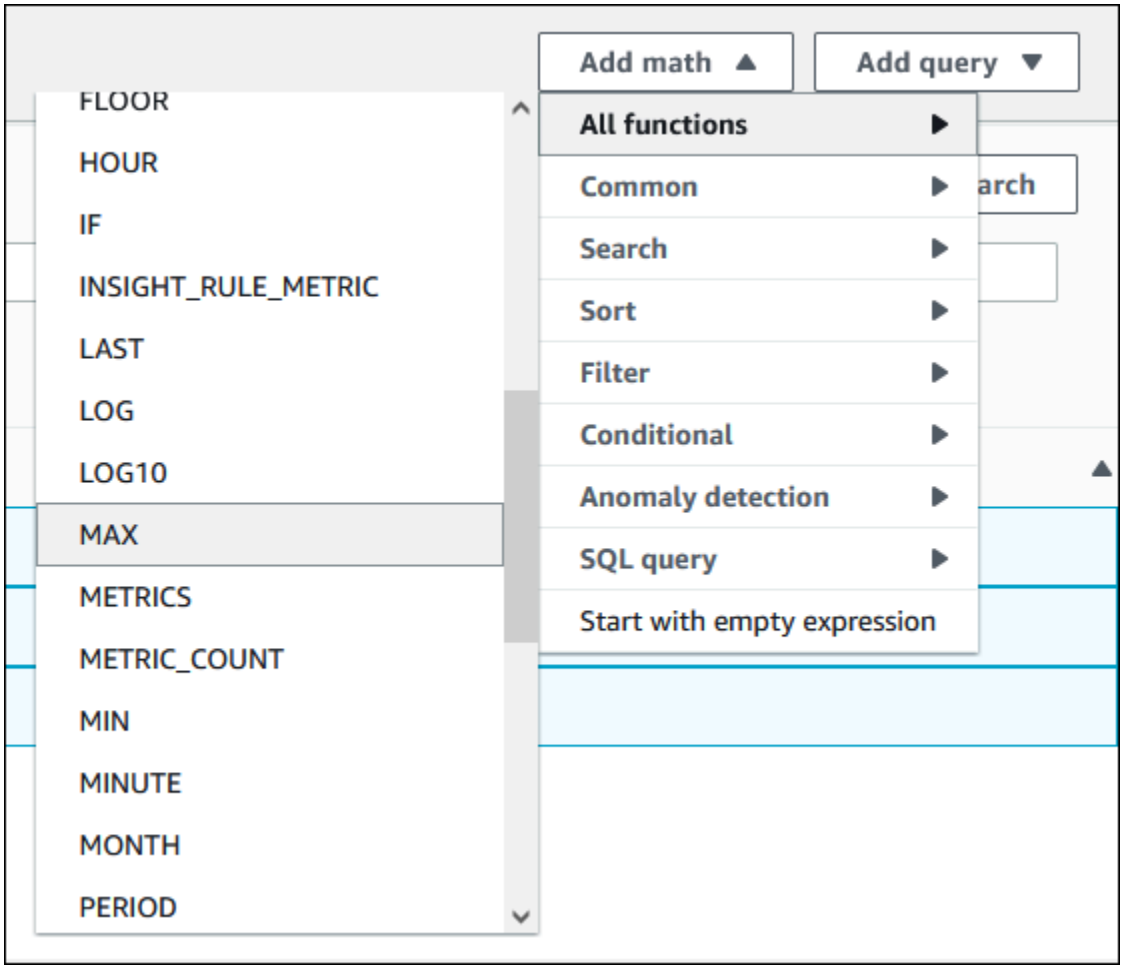
6. [RDS]、[Per-Database Metrics] (データベース別メトリクス) を順に選択します。
7. 検索ボックスに「**ReplicaLag**」と入力して Enter キーを押し、DB クラスター内の各 DB インスタンスを選択します。

次の図は、ReplicaLag メトリクスに DB インスタンスが選択されている [Select metric] (メトリクスの選択) ページを示しています。



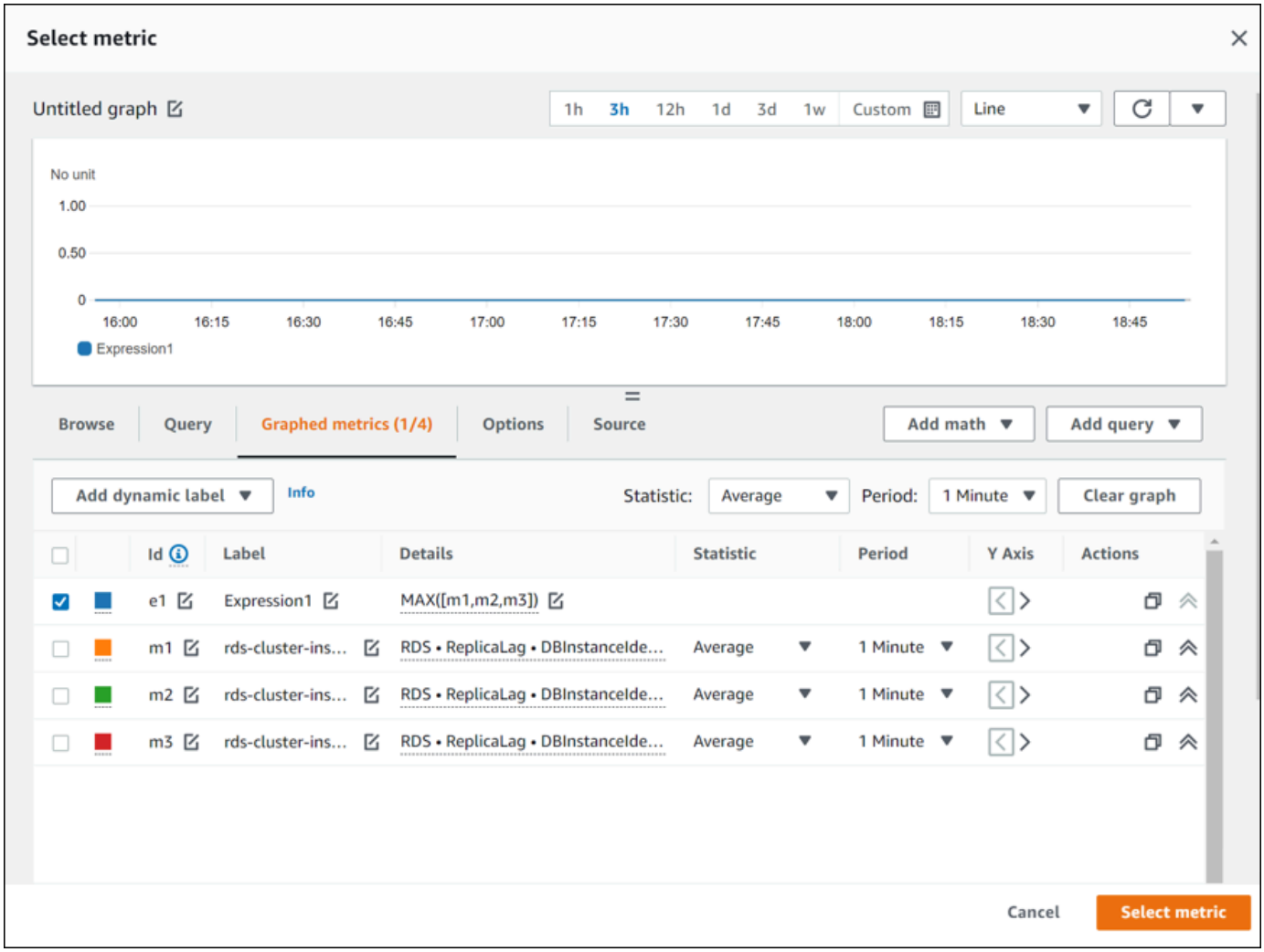
このアラームでは、マルチ AZ DB クラスター内の 3 つの DB インスタンスすべてのレプリカラグが考慮されます。いずれかの DB インスタンスがしきい値を超えると、アラームが応答します。アラームでは、3 つのメトリクスの最大値を返す数式が使用されています。まず、メトリクス名でソートしてから、3 つの ReplicaLag メトリクスすべてを選択します。

8. [Add math] (算術の追加) で、[All functions] (すべての関数)、[MAX] (最大) を順に選択します。



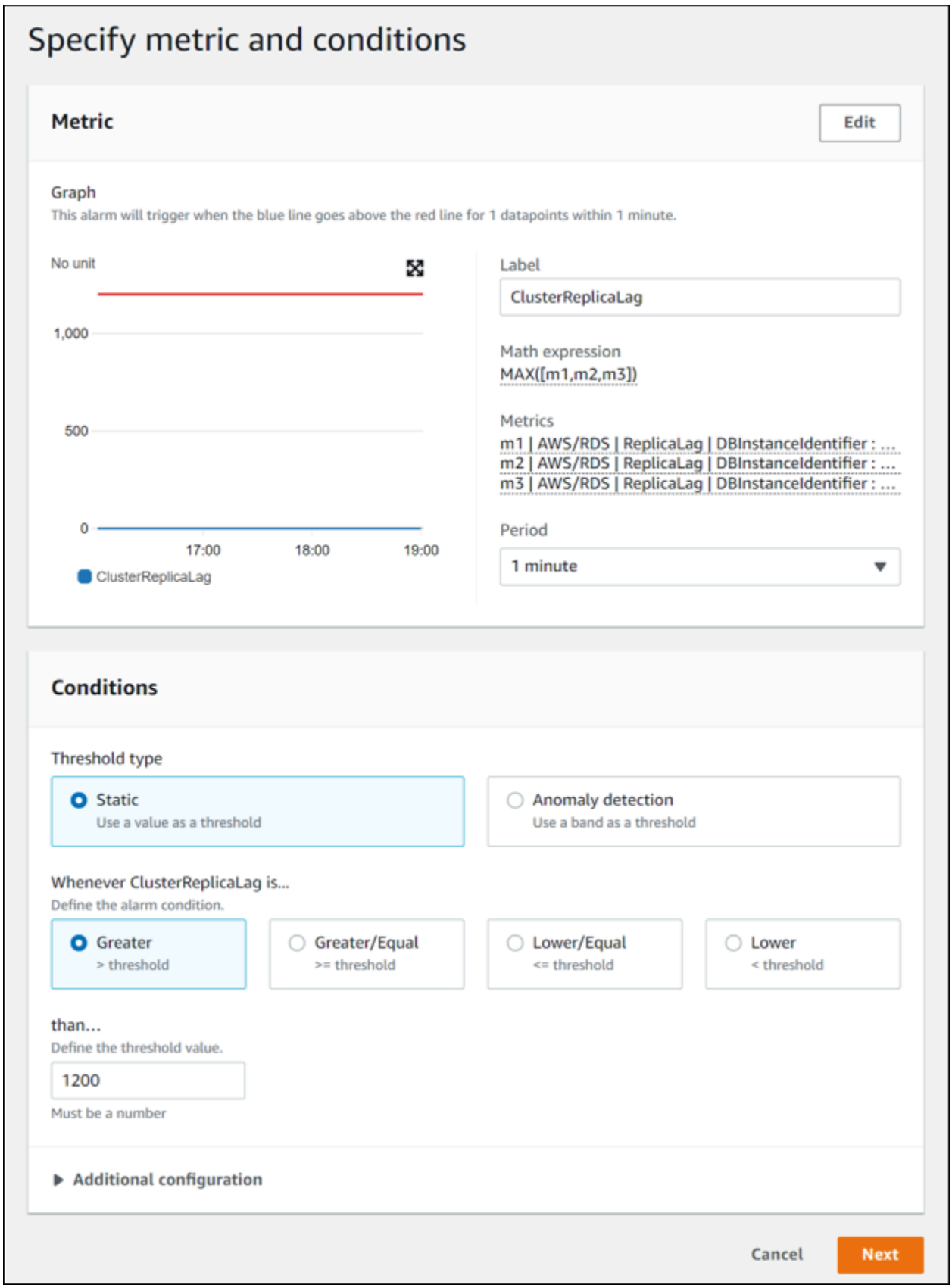
9. [Graphed metrics] (グラフ化したメトリクス) タブを選択し、Expression1 の詳細を **`MAX([m1,m2,m3])`** に編集します。
10. 3 つの ReplicaLag メトリクスすべてについて、[Period] (期間) を 1 分に変更します。
11. Expression1 を除くすべてのメトリクスの選択を解除します。

    [Select metric] (メトリクスの選択) ページは、次の図のようになります。

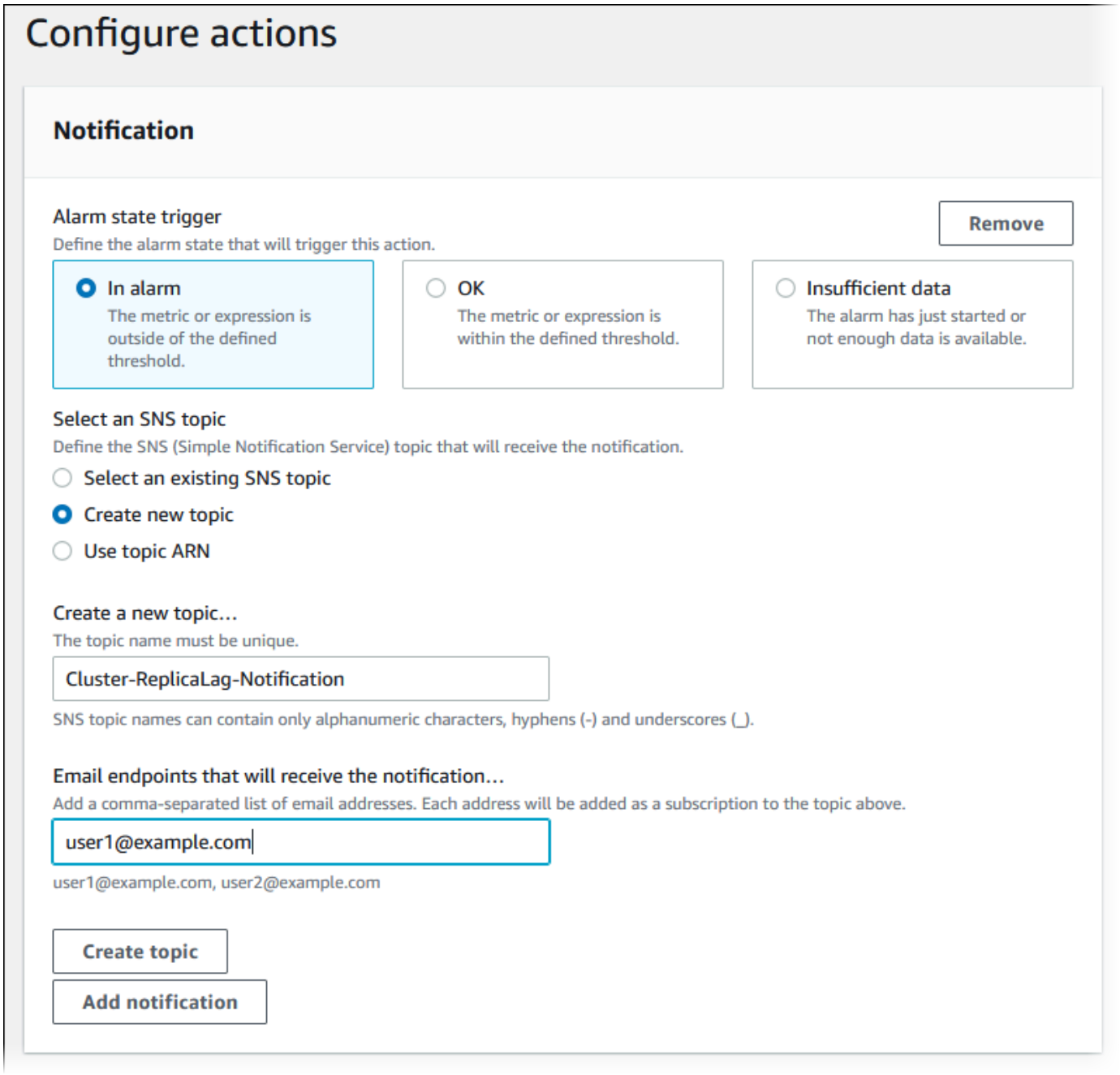


12. [メトリクスの選択] を選択します。
13. [Specify metric and conditions] (メトリクスと条件の指定) ページで、ラベルをわかりやすい名前 (**`ClusterReplicaLag`** など) に変更し、[Define the threshold value] (しきい値の定義) で秒数を入力します。このチュートリアルでは、「**`1200`**」秒 (20 分) と入力します。この値は、ワークロード要件に合わせて調整できます。

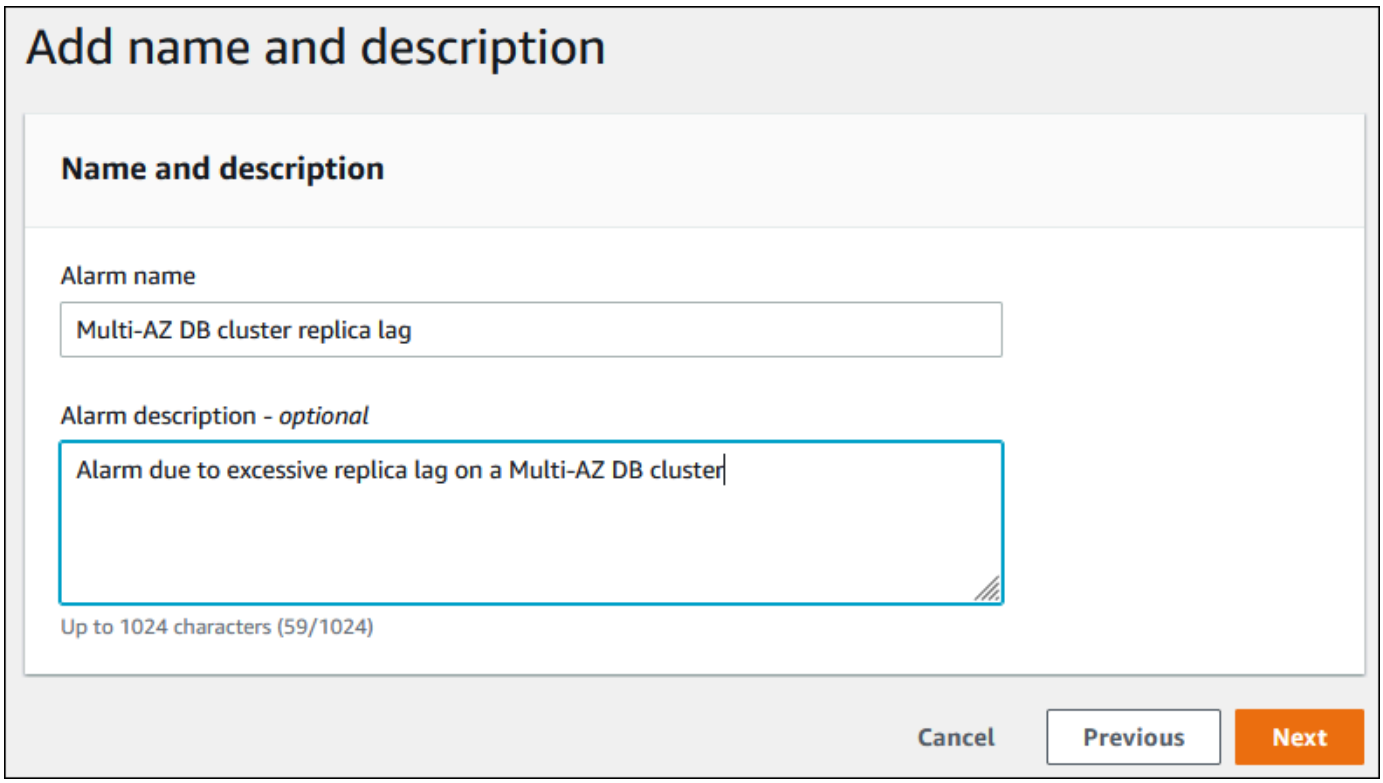
    [Specify metric and conditions] (メトリクスと条件の指定) ページは、次の図のようになります。


Specify metric and conditions
Metric
Edit
Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.
No unit
1,000
500
0
17:00
18:00
19:00
ClusterReplicaLag
Label
ClusterReplicaLag
Math expression
MAX([m1,m2,m3])
Metrics
m1 | AWS/RDS | ReplicaLag | DBInstanceIdentifier : ...
m2 | AWS/RDS | ReplicaLag | DBInstanceIdentifier : ...
m3 | AWS/RDS | ReplicaLag | DBInstanceIdentifier : ...
Period
1 minute
Conditions
Threshold type
Static
Use a value as a threshold
Anomaly detection
Use a band as a threshold
Whenever ClusterReplicaLag is...
Define the alarm condition.
Greater
> threshold
Greater/Equal
>= threshold
Lower/Equal
<= threshold
Lower
< threshold
than...
Define the threshold value.
1200
Must be a number
Additional configuration
Cancel
Next

14. [Next] (次へ) をクリックすると、[Configure actions] (アクションの設定) ページが表示されます。
15. [In alarm] (アラーム状態) を選択したまま、[Create new topic] (新しいトピックの作成) 選択し、トピック名と有効な E メールアドレスを入力します。



16. [Create topic] (トピックの作成) を選択してから、[Next] (次へ) をクリックします。
17. [Add name and description] (名前と説明を追加) ページで、アラーム名とアラームの説明を入力し、[Next] (次へ) をクリックします。



18. [Preview and create] (プレビューと作成) ページで、作成しようとしているアラームをプレビューしてから、[Create alarm] (アラームの作成) を選択します。

CloudWatch Database Insights による Amazon RDS データベースのモニタリング

Amazon RDS DB インスタンスのフリートのデータベース負荷 (DB 負荷) は、Database Insights を使用してモニタリングします。DB 負荷は、データベース内のセッションアクティビティのレベルを測定するものです。Database Insights を使用して、Amazon RDS データベースのパフォーマンスを大規模に分析およびトラブルシューティングできます。

Database Insights を使用してデータベースのフリートの DB 負荷を視覚化したり、負荷を待機、SQL ステートメント、ホスト、ユーザー別にフィルタリングしたりできます。

RDS では、Amazon RDS データベースに対してデフォルトで Database Insights のスタンダードモードが有効になっています。

Amazon CloudWatch コンソールで Database Insights を使用する方法については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch Database Insights」を参照してください。

料金

料金の詳細については、「Amazon CloudWatch の料金」を参照してください。

トピック

- Amazon RDS エンジン、リージョン、インスタンスクラスの Database Insights のサポート
- Amazon RDS で Database Insights のアドバンストモードを有効にする
- Amazon RDS で Database Insights のスタンダードモードを有効にする
- Amazon RDS の Database Insights でスロー SQL クエリをモニタリングするようにデータベースを設定する
- Amazon RDS の Database Insights に関する考慮事項

Amazon RDS エンジン、リージョン、インスタンスクラスの Database Insights のサポート

次の表に、Database Insights をサポートしている Amazon RDS DB エンジンを示します。

Amazon RDS DB エンジン	サポート対象のエンジンバージョンとリージョン	インスタンスクラスに関する制限
Amazon RDS for MariaDB	RDS for MariaDB に関する Database Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	Database Insights は、次のインスタンスクラスではサポートされていません。 • db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
RDS for MySQL	RDS for MySQL で Database Insights が使用できるバージョンとリージョンの詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	Database Insights は、次のインスタンスクラスではサポートされていません。 • db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small

Amazon RDS DB エンジン	サポート対象のエンジンバージョンとリージョン	インスタンスクラスに関する制限
Amazon RDS for Microsoft SQL Server	RDS for SQL Server で Database Insights が使用できるバージョンとリージョンの詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし
Amazon RDS for PostgreSQL	RDS for PostgreSQL で Database Insights が使用できるバージョンとリージョンの詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし
Amazon RDS for Oracle	RDS for Oracle で Database Insights が使用できるバージョンとリージョンの詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし

Amazon RDS DB エンジン、リージョン、インスタンスクラスの Database Insights のサポート

次の表に、Database Insights の機能をサポートしている Amazon RDS DB エンジンを示します。

機能	料金階層	サポートされるリージョン	サポートされるDB エンジン	サポートされるインスタンスクラス
Performance Insights の SQL 統計	すべて	すべて	すべて	すべて
Amazon RDS の Performance Insights ダッシュボードを使用した Oracle 実行プランの分析	すべて	すべて	RDS for Oracle	すべて
一定期間のデータベースパフォーマンスの分析	有料階層のみ	すべて	• RDS for MariaDB • RDS for MySQL • RDS for PostgreSQL	すべて
Performance Insights のプロアクティブ推奨事項の表示	有料階層のみ	• 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アジアパシフィック (ムンバイ)	すべて	すべて

機能	料金階層	サポートされるリージョン	サポートされるDB エンジン	サポートされるインスタンスクラス
		• アジアパシフィック (ソウル) • アジアパシフィック (シンガポール) • アジアパシフィック (シドニー) • アジアパシフィック (東京) • カナダ (中部) • 欧州 (フランクフルト) • 欧州 (アイルランド) • 欧州 (ロンドン) • 欧州 (パリ) • 欧州 (ストックホルム) • 南米 (サンパウロ)		

Amazon RDS が Database Insights をサポートしているリージョン

Amazon RDS は、次の AWS リージョンで Database Insights をサポートしています。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)

- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (マレーシア)
- アジアパシフィック (メルボルン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWSGovCloud(米国西部)

Amazon RDS で Database Insights のアドバンストモードを有効にする

Amazon RDS で Database Insights のアドバンストモードを有効にするには、次の手順を使用します。

DB インスタンスまたはマルチ AZ DB クラスターの作成時に Database Insights のアドバンストモードを有効にする

Amazon RDS のデータベースを作成するときに、Database Insights のアドバンストモードを有効にします。

Console

コンソールで、DB インスタンスまたマルチ AZ DB クラスターを作成するときに Database Insights のアドバンストモードを有効にできます。

コンソールを使用して DB インスタンスまたはマルチ AZ DB クラスターを作成するときに Database Insights のアドバンストモードを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベース] をクリックします。
3. [データベースの作成] を選択します。
4. [Database Insights] セクションで [アドバンストモード] を選択し、次のいずれかのオプションを選択します。
   - 保持期間 - Performance Insights データを保持する期間。Database Insights のアドバンストモードでは、保持期間は 15 か月とする必要があります。
   - AWS KMS key - KMS キーを指定します。Performance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
5. [データベースの作成] を選択します。

AWS CLI

DB インスタンスまたはマルチ AZ DB クラスターの作成時に Database Insights のアドバンストモードを有効にするには、create-db-instance または create-db-cluster AWS CLI コマンドを呼び出し、次の値を指定します。

- `--database-insights-mode advanced` – Database Insights のアドバンストモードを有効にします。
- `--engine` – DB インスタンス のデータベースエンジン。
- `--db-instance-identifier` - DB インスタンスの識別子または `--db-cluster-identifier` - マルチ AZ DB クラスターの識別子。
- `--enable-performance-insights` – Database Insights の Performance Insights を有効にします。
- `--performance-insights-retention-period` – DB インスタンスまたはマルチ AZ DB クラスターのデータの保持期間。Database Insights を有効にするには、保持期間を 465 日以上にする必要があります。

次の例では、DB インスタンスの作成時に Database Insights のアドバンストモードを有効にします。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --database-insights-mode advanced \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Windows の場合:

```
aws rds create-db-instance ^
    --database-insights-mode advanced ^
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

DB インスタンスまたはマルチ AZ DB クラスターの作成時に Database Insights のアドバンストモードを有効にするには、CreateDBInstance または CreateDBCluster Amazon RDS API オペレーションに次のパラメータを指定します。

- `DatabaseInsightsMode` ~ `advanced`
- `EnablePerformanceInsights` ~ `True`
- `PerformanceInsightsRetentionPeriod` – 465 日以上

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Database Insights のアドバンストモードを有効にする

Amazon RDS のデータベースを変更するときに Database Insights を有効にします。DB インスタンスを変更して Database Insights のアドバンストモードを有効にしても、ダウンタイムは発生しません。

> **Note**
>
> Database Insights を有効にするには、マルチ AZ DB クラスター内の各 DB インスタンスに、同じ Performance Insights と拡張モニタリング設定が必要です。

Console

コンソールで、DB インスタンスまたマルチ AZ DB クラスターを変更するときに Database Insights のアドバンストモードを有効にできます。

コンソールを使用して DB インスタンスまたはマルチ AZ DB クラスターを変更するときに Database Insights のアドバンストモードを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベース] をクリックします。
3. DB インスタンスまたはマルチ AZ DB クラスターを選択した上で、[Modify] (変更) を選択します。
4. [Database Insights] セクションで [アドバンストモード] を選択し、次のいずれかのオプションを選択します。

- 保持期間 - Performance Insights データを保持する期間。Database Insights のアドバンストモードでは、保持期間は 15 か月とする必要があります。
- AWS KMS key - KMS キーを指定します。Performance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。

5. [続行] を選択します。
6. [Scheduling of Modifications] で、[Apply immediately] を選択します。[スケジュールされた次回のメンテナンス時間帯に適用] を選択すると、データベースではこの設定が無視され、Database Insights のアドバンストモードが直ちに有効になります。
7. [インスタンスの変更] を選択します。

AWS CLI

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Database Insights のアドバンストモードを有効にするには、modify-db-instance または modify-db-cluster AWS CLI コマンドを呼び出し、次の値を指定します。

- `--database-insights-mode advanced` – Database Insights のアドバンストモードを有効にします。
- `--db-instance-identifier` - DB インスタンスの識別子または `--db-cluster-identifier` - マルチ AZ DB クラスターの識別子。
- `--enable-performance-insights` – Database Insights の Performance Insights を有効にします。
- `--performance-insights-retention-period` – DB インスタンスのデータの保持期間。Database Insights のアドバンストモードを有効にするには、保持期間を 465 日以上にする必要があります。

次の例では、DB インスタンスの変更時に Database Insights のアドバンストモードを有効にします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --database-insights-mode advanced \
```

```
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Windows の場合:

```
aws rds modify-db-instance ^
    --database-insights-mode advanced ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Database Insights のアドバンストモードを有効にするには、ModifyDBInstance または ModifyDBCluster Amazon RDS API オペレーションに次のパラメータを指定します。

- `DatabaseInsightsMode` ~ `advanced`
- `EnablePerformanceInsights` ~ `True`
- `PerformanceInsightsRetentionPeriod` ~ 465 日以上

Amazon RDS で Database Insights のスタンダードモードを有効にする

Amazon RDS で Database Insights のスタンダードモードを有効にするには、次の手順を使用します。

DB インスタンスまたはマルチ AZ DB クラスターの作成時に Database Insights のスタンダードモードを有効にする

Amazon RDS のデータベースを作成するときに、Database Insights のスタンダードモードを有効にします。

Console

コンソールで、DB インスタンスまたはマルチ AZ DB クラスターを作成するときに Database Insights のスタンダードモードを有効にできます。

コンソールを使用して DB インスタンスまたはマルチ AZ DB クラスターを作成するときに Database Insights のスタンダードモードを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベース] をクリックします。
3. [データベースの作成] を選択します。
4. [Database Insights] セクションで [スタンダードモード] を選択し、次のオプションから選択して Performance Insights のオンとオフを切り替えます。
   - Performance Insights を無効にするには、[Performance Insights を有効にする] の選択を解除します。
   - Performance Insights を有効にするには、[Performance Insights を有効にする] を選択します。Performance Insights を設定するには、次のオプションを指定します。
     - 保持期間 - Performance Insights データを保持する期間。保持期間は 7 日以上にする必要があります。
     - AWS KMS key - KMS キーを指定します。Performance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
5. [データベースの作成] を選択します。

AWS CLI

DB インスタンスまたマルチ AZ DB クラスターの作成時に Database Insights のスタンダードモードを有効にするには、create-db-instance または create-db-cluster AWS CLI コマンドを呼び出し、次の値を指定します。

- `--database-insights-mode standard` – Database Insights のスタンダードモードを有効にします。
- `--engine` – DB インスタンス のデータベースエンジン。
- `--db-instance-identifier` - DB インスタンスの識別子または `--db-cluster-identifier` - マルチ AZ DB クラスターの識別子。
- `--enable-performance-insights` または `--no-enable-performance-insights` – Performance Insights を有効または無効にします。`--enable-performance-insights` を指

定する場合は、--performance-insights-retention-period (DB インスタンスのデータの保持期間) も指定する必要があります。保持期間は 7 日以上にする必要があります。

次の例では、DB インスタンスに Database Insights のスタンダードモードと Performance Insights を有効にします。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --database-insights-mode standard \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Windows の場合:

```
aws rds create-db-instance ^
    --database-insights-mode standard ^
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

次の例では、DB インスタンスに Database Insights のスタンダードモードと Performance Insights を有効および無効にします。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --database-insights-mode standard \
    --engine postgresql \
    --db-instance-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Windows の場合:

```
aws rds create-db-instance ^
    --database-insights-mode standard ^
```

```
    --engine postgresql ^
    --db-instance-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

DB インスタンスまたはマルチ AZ DB クラスターの作成時に Database Insights のスタンダードモードを有効にするには、CreateDBInstance または CreateDBCluster Amazon RDS API オペレーションに次のパラメータを指定します。

- `DatabaseInsightsMode` ~ `standard`
- `EnablePerformanceInsights` ~ `True` または `False`。`EnablePerformanceInsights` を `True` に設定する場合は、`PerformanceInsightsRetentionPeriod` を 7 日以上に設定する必要があります。

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Database Insights のスタンダードモードを有効にする

Amazon RDS のデータベースを変更するときに Database Insights のスタンダードモードを有効にします。DB インスタンスを変更して Database Insights の標準モードを有効にしても、ダウンタイムは発生しません。

> **Note**
>
> Database Insights を有効にするには、マルチ AZ DB クラスター内の各 DB インスタンスに、同じ Performance Insights と拡張モニタリング設定が必要です。

Console

コンソールで、DB インスタンスまたはマルチ AZ DB クラスターを作成するときに Database Insights のスタンダードモードを有効にできます。

コンソールを使用して DB インスタンスまたはマルチ AZ DB クラスターを変更するときに Database Insights のスタンダードモードを有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. [データベース] をクリックします。
3. DB インスタンスまたはマルチ AZ DB クラスターを選択した上で、[Modify] (変更) を選択します。
4. [Database Insights] セクションで [スタンダードモード] を選択し、次のオプションから選択します。

   - Performance Insights を無効にするには、[Performance Insights を有効にする] の選択を解除します。
   - Performance Insights を有効にするには、[Performance Insights を有効にする] を選択します。Performance Insights を設定するには、次のオプションを指定します。
     - 保持期間 - Performance Insights データを保持する期間。保持期間は 7 日以上にする必要があります。
     - AWS KMS key - KMS キーを指定します。Performance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
5. [続行] を選択します。
6. [Scheduling of Modifications] で、[Apply immediately] を選択します。[スケジュールされた次回のメンテナンス時間帯に適用] を選択すると、データベースではこの設定が無視され、Database Insights のスタンダードモードが直ちに有効になります。
7. [インスタンスの変更] を選択します。

AWS CLI

DB インスタンスまたマルチ AZ DB クラスターの変更時に Database Insights のスタンダードモードを有効にするには、modify-db-instance または modify-db-cluster AWS CLI コマンドを呼び出し、次の値を指定します。

- `--database-insights-mode standard` – Database Insights のスタンダードモードを有効にします。
- `--db-instance-identifier` - DB インスタンスの識別子または `--db-cluster-identifier` - マルチ AZ DB クラスターの識別子。
- `--enable-performance-insights` または `--no-enable-performance-insights` – Performance Insights を有効または無効にします。`--enable-performance-insights` を指定する場合は、`--performance-insights-retention-period` (DB インスタンスまたは

マルチ AZ DB クラスターのデータの保持期間) も指定する必要があります。保持期間は 7 日以上にする必要があります。

次の例では、DB インスタンスの変更時に Database Insights のスタンダードモードを有効にし、Performance Insights を有効にします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --database-insights-mode standard \
    --db-instance-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Windows の場合:

```
aws rds modify-db-instance ^
    --database-insights-mode standard ^
    --db-instance-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

次の例では、DB インスタンスの変更時に Database Insights のスタンダードモードを有効にし、Performance Insights を無効にします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --database-insights-mode standard \
    --db-instance-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Windows の場合:

```
aws rds modify-db-instance ^
    --database-insights-mode standard ^
    --db-instance-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Database Insights のスタンダードモードを有効にするには、ModifyDBInstance または ModifyDBCluster Amazon RDS API オペレーションに次のパラメータを指定します。

- `DatabaseInsightsMode ~ standard`
- `EnablePerformanceInsights ~ True または False。EnablePerformanceInsights を True に設定する場合は、PerformanceInsightsRetentionPeriod を 7 日以上に設定する必要があります。`

Amazon RDS の Database Insights でスロー SQL クエリをモニタリングするようにデータベースを設定する

データベースのスロー SQL クエリをモニタリングするには、Database Insights ダッシュボードの [スロー SQL クエリ] セクションを使用できます。スロー SQL クエリをモニタリングするようにデータベースを設定する前は、[スロー SQL クエリ] セクションは空白です。

Database Insights ダッシュボードでのスロー SQL クエリのモニタリングの詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch Database Insights のデータベースインスタンスダッシュボードの表示」を参照してください。

Database Insights を使用して、スロー SQL クエリをモニタリングするようにデータベースを設定するには、次の手順を実行します。

1. CloudWatch Logs へのログエクスポートを有効にします。
2. DB インスタンスの DB パラメータグループを作成または変更します。

ログエクスポートの設定方法については、「Amazon RDS ユーザーガイド」の「Amazon CloudWatch Logs へのデータベースログの発行」を参照してください。

DB パラメータグループを作成または変更するには、以下のトピックを参照してください。

- Amazon RDS での DB パラメータグループの作成
- Amazon RDS の DB パラメータグループのパラメータの変更

RDS for MariaDB

スロー SQL クエリをモニタリングするように RDS for MariaDB DB インスタンスを設定するには、次のパラメータを設定します。

- `log_slow_query` - `1` に設定
- `log_slow_query_time` - `1.0` に設定
- `log_output` - `FILE` に設定

RDS for MySQL

スロー SQL クエリをモニタリングするように RDS for MySQL DB インスタンスを設定するには、次のパラメータを設定します。

- `slow_query_log` - `1` に設定
- `long_query_time` - `1.0` に設定
- `log_output` - `FILE` に設定

RDS for PostgreSQL

スロー SQL クエリをモニタリングするように RDS for PostgreSQL DB インスタンスを設定するには、次のパラメータを設定します。これらのパラメータを設定すると、DB インスタンスのパフォーマンスが低下する可能性があることに注意してください。

- `log_min_duration_statement` - `1000` に設定
- `log_statement` - `none` に設定
- `log_destination` - `stderr` に設定

> **Note**
>
> RDS for MySQL では、`long_query_time` パラメータを 1 マイクロ秒の精度で設定できます。例えば、このパラメータを `0.000001` に設定できます。DB インスタンスのクエリの量によっては、`long_query_time` パラメータの値によってパフォーマンスが低下する可能性があります。値 `1.0` から開始し、ワークロードに基づいて調整します。このパラメータを `0` に設定すると、Database Insights はすべてのクエリをログに記録します。

RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL ログの詳細については、以下を参照してください。

- MariaDB データベースのログファイル
- MySQL データベースのログファイル
- RDS for PostgreSQL データベースログファイル

Amazon RDS の Database Insights に関する考慮事項

Amazon RDS の Database Insights に関する考慮事項を次に示します。

- 特定の マルチ AZ DB クラスター内の特定の DB インスタンスで Database Insights を管理することはできません。
- Database Insights のアドバンストモードを有効にするには、Performance Insights を有効にし、Performance Insights の保持期間を少なくとも 465 日 (15 か月) に設定する必要があります。Performance Insights の保持期間を15 か月に設定しても、Database Insights のコスト以外に追加のコストは発生しません。Database Insights の料金については、「Amazon CloudWatch 料金表」を参照してください。
- Database Insights を有効にするには、マルチ AZ DB クラスター内の各 DB インスタンスに、同じ Performance Insights と拡張モニタリング設定が必要です。
- Database Insights のいずれかのモードを有効にするために DB インスタンスを変更しても、ダウンタイムは発生しません。

Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング

Important

AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Performance Insights は、既存の Amazon RDS モニタリング機能を拡張して、データベースのパフォーマンスを明確にし、分析しやすくします。Performance Insights ダッシュボードを使用してAmazon RDS DB インスタンスのロードのデータベースロードを視覚化したり、ロードを待機、SQL ステートメント、ホスト、ユーザー別にフィルタリングしたりできます。Amazon

DocumentDB での Performance Insights 使用については、「Amazon DocumentDB デベロッパーガイド」を参照してください。

トピック

- Amazon RDS での Performance Insights の概要
- Amazon RDS の Performance Insights の有効化と無効化
- Amazon RDS for MariaDB または MySQL における Performance Insights のPerformance Schema の概要
- Performance Insights 用のアクセスポリシーの設定
- Performance Insights ダッシュボードを使用してメトリクスを分析する
- Performance Insights のプロアクティブ推奨事項の表示
- Performance Insights API for Amazon RDS によるメトリクスの取得
- AWS CloudTrail を使用した Performance Insights 呼び出しのログ記録
- Performance Insights API とインターフェイス VPC エンドポイント (AWS PrivateLink)

Amazon RDS での Performance Insights の概要

Important

AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までにニーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に

動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

デフォルトで RDS は、すべての Amazon RDS エンジンのコンソール作成ウィザードで Performance Insights を有効にします。DB インスタンスに複数のデータベースがある場合、Performance Insights はパフォーマンスデータを集計します。

Amazon RDS の Performance Insights の概要は次の動画で確認できます。

Amazon Aurora PostgreSQL のパフォーマンスを分析する Performance Insights を使用する

Important

以下のトピックでは、Aurora 以外の DB エンジンで Amazon RDS Performance Insights を使用する方法について説明します。Amazon Aurora での Amazon RDS Performance Insights の使用については、「Amazon Aurora ユーザーガイド」の「Using Amazon RDS Performance Insights」(Amazon RDS Performance Insights の使用) を参照してください。

トピック

- データベース負荷
- 最大 CPU 容量
- Amazon RDS DB エンジンとインスタンスクラスでサポートされている Performance Insights
- Performance Insights の料金とデータ保持

データベース負荷

データベース負荷 (DB 負荷) は、データベース内のセッションアクティビティのレベルを測定します。DBLoad は Performance Insights の主要なメトリクスで、Performance Insights は 1 秒ごとに DB 負荷を収集します。

トピック

- アクティブなセッション
- 平均アクティブセッション
- 平均アクティブ実行
- ディメンション

アクティブなセッション

データベースセッションは、リレーショナルデータベースとのアプリケーションのダイアログを表します。アクティブなセッションとは、DB エンジンに作業を送信し、レスポンスを待っている接続です。

セッションは、CPU での動作中、またはリソースが使用可能になるのを待っているときにアクティブになります。例えば、アクティブなセッションでは、ページ (またはブロック) がメモリに読み込まれるのを待機し、ページからデータを読み取る間に CPU を消費することがあります。

平均アクティブセッション

平均アクティブセッション (AAS)はDBLoadPerformance Insights のメトリクスの単位です。データベース上で同時にアクティブなセッション数を測定します。

毎秒、Performance Insights は、クエリを同時に実行するセッションの数をサンプリングします。Performance Insights は、アクティブなセッションごとに以下のデータを収集します。

- SQL ステートメント
- セッション状態 (CPU で実行中または待機中)
- ホスト
- SQL を実行しているユーザー

Performance Insights は、特定期間の総セッション数を総サンプル数で割って AAS を計算します。たとえば、次の表は、1 秒間隔で実行中のクエリの連続する 5 つのサンプルを示しています。

例	クエリを実行しているセッション数	AAS	計算
1	2	2	合計 2 セッション/1 サンプル
2	0	1	合計 2 セッション/2 サンプル
3	4	2	合計 6 セッション/3 サンプル
4	0	1.5	合計 6 セッション/4 サンプル
5	4	2	合計 10 セッション/5 サンプル

前述の例では、時間間隔の DB ロードは 2 AAS でした。この測定は、5 つのサンプルを採取した期間に、平均して 2 つのセッションがある時点でアクティブであったことを意味します。

平均アクティブ実行

1 秒あたりの平均アクティブ実行 (AAE) は AAS に関連しています。AAE を計算するために、Performance Insightsでは、クエリの合計実行時間を時間間隔で割ります。次の表に、前述の表の同じクエリに対する AAE 計算を示します。

経過時間 (秒)	合計実行時間 (秒)	AAE	計算
60	120	2	120 実行秒 / 60 経過秒
120	120	1	120 実行秒 / 120 経過秒
180	380	2.11	380 実行秒 /180秒経過
240	380	(1.58)	380 実行秒/240秒経過
300	600	2	600 実行秒/300 経過秒

ほとんどの場合、クエリの AAS と AAE はほぼ同じです。ただし、計算への入力は異なるデータソースであるため、計算はわずかに異なります。

ディメンション

この db.load メトリクスは、ディメンションと呼ばれるサブコンポーネントに分割できるため、他の時系列メトリクスとは異なります。ディメンションは、DBLoad メトリクスのさまざまな特性のカテゴリにより「スライス化されている」と考えることができます。

パフォーマンスの問題を診断する場合、多くの場合、以下のディメンションが最も役立ちます。

トピック
- 待機イベント
- 上位の SQL
- プラン

のディメンションの詳細なリストについては、Amazon RDSエンジン、「ディメンションでスライスされた DB のロード」を参照してください。

待機イベント

待機イベントを指定すると、SQL ステートメントは、特定のイベントが発生するまで待機してから、実行を継続できます。待機イベントは、作業が妨げられる場所を示すため、DB ロードの重要なディメンションまたはカテゴリになります。

すべてのアクティブなセッションはCPU 上で実行されているか、待っています。例えば、セッションがメモリでバッファを検索したり、計算を実行したり、プロシージャコードを実行したりするときに CPU を消費します。セッションが CPU を消費していないときは、メモリバッファが空くのを待っているか、データファイルの読み取りや口グの書き込みを待っている可能性があります。セッションのリソース待機時間が長くなると、CPU 上で動作する時間は短くなります。

データベースのチューニングのとき、セッションが待っているリソースを見つけようとすることがよくあります。例えば、2 つまたは 3 つの待機イベントが DBロードの 90% を占めることがあります。これは、平均して、アクティブなセッションが少数のリソースを待機するためにほとんどの時間を費やしていることを意味します。これらの待機の原因がわかれば、解決策を試すことができます。

待機イベントは、DB エンジンごとに異なります。

- MariaDB および MySQL のすべての待機イベントの詳細については、MySQL ドキュメントの「イベント待機サマリーテーブル」を参照してください。

- すべての PostgreSQL 待機イベントの詳細については、PostgreSQL ドキュメントの「統計コレクター > 待機イベントテーブル」を参照してください。
- すべての Oracle 待機イベントについては、Oracle ドキュメントの「待機リストの説明」を参照してください。
- SQL Server のすべての待機イベントについては、SQL Server ドキュメントの「待機の種類」を参照してください。

> **Note**
>
> Oracle では、関連付けられた SQL ステートメントがなくてもバックグラウンドプロセスが実行されることがあります。このような場合、Performance Insights はバックグラウンドプロセスのタイプとそのバックグラウンドプロセスに関連付けられた待機クラスをコロンで連結してレポートします。バックグラウンドプロセスのタイプには、`LGWR`、`ARC0`、`PMON`などが含まれます。
> 例えば、アーカイブ処理で I/O を実行しているとき、Performance Insights によるレポートは `ARC1:System I/O` のようになります。バックグラウンドプロセスタイプが欠落していて、Performance Insights が `:System I/O` のように待機クラスだけをレポートする場合もあります。

上位の SQL

待機イベントはボトルネックを示しますが、上位の SQL は、どのクエリが DB ロードの最も大きな原因になっているかを示します。例えば、多くのクエリが現在データベースで実行されている可能性がありますが、1 つのクエリが DB ロードの 99% を占めている可能性もあります。この場合、負荷が高いと、クエリに問題がある可能性があります。

デフォルトでは、Performance Insights コンソールには、データベース負荷の原因となっている上位の SQL クエリが表示されます。コンソールには、各ステートメントに関連する統計情報も表示されます。特定のステートメントのパフォーマンスの問題を診断するには、その実行プランを調べます。

プラン

実行プラン (あるいはシンプルにプランとも呼ばれるもの) は、データにアクセスする際の一連のステップのことです。例えば、テーブル `t1` と `t2` を統合するプランでは、`t1` 内のすべての行をループしながら、その各行を `t2` 内の行と比較します。リレーショナルデータベースでのオプティマイザとは組み込みのコードのことで、これは SQL クエリのための最も効率的なプランを決定します。

DB インスタンスの場合、Performance Insights が実行プランを自動的に収集します。SQL のパフォーマンスに関する問題を診断するには、リソースの使用量が高い SQL クエリに関して取得したプランを検証します。プランから、データベースがどのようにクエリを解析して実行したかがわかります。

プランを使用して DB 負荷を分析する方法については、以下を参照してください。

- Oracle: Amazon RDS の Performance Insights ダッシュボードを使用した Oracle 実行プランの分析
- SQL Server: Amazon RDS の Performance Insights ダッシュボードを使用した SQL Server 実行プランの分析

プランキャプチャ

Performance Insights は、5 分ごとに、リソースを最も多く消費しているクエリを特定し、対応するプランをキャプチャします。したがって、膨大な数のプランを手動で収集して管理する必要はありません。ユーザーは、[Top SQL] (上位の SQL) タブを通じて、最も問題のあるクエリのプランに対処することができます。

> **Note**
>
> Performance Insights では、含まれるテキストが収集可能な最大数を超えるクエリのプランはキャプチャされません。詳しくは、「Performance Insights ダッシュボードでより多くの SQL テキストにアクセスする」を参照してください。

実行プランの保持期間は、Performance Insights のデータの保持期間と同じです。無料利用枠の保持設定は「デフォルト (7 日)」です。パフォーマンスデータをさらに長期間保持するには、1～24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。

ダイジェストクエリ

[Top SQL] (上位の SQL) タブには、デフォルトでダイジェストクエリが表示されます。ダイジェストクエリ自体にはプランはありませんが、リテラル値を使用するすべてのクエリにはプランがあります。例えば、ダイジェストクエリにテキストWHERE `email`=?が含まれる場合があります。このダイジェストの中には、テキスト `WHERE email=user1@example.com` を含むクエリと、`WHERE email=user2@example.com` を含むクエリの 2 つがあります。これらのリテラルクエリには、それぞれ複数のプランが含まれる可能性があります。

ダイジェストクエリを選択すると、選択したダイジェストの子ステートメントのすべてのプランがコンソールに表示されます。したがって、プランを見つけるために、子ステートメント全体を確認する必要はありません。表示されているプランの中に、上位 10 の子ステートメントのリストにないものが含まれる場合があります。コンソールには、クエリが上位 10 であるかどうかにかかわらず、プランが収集されたすべての子クエリのプランが表示されます。

最大 CPU 容量

ダッシュボードの [データベースロード] グラフで、セッション情報が収集、集計、表示されます。アクティブなセッションが最大 CPU 容量を超えているかどうかを確認するには、最大 vCPU ラインとの関係を調べます。Performance Insights は、最大 vCPU 値を DB インスタンスの vCPU (仮想 CPU) のコア数によって決定します。

vCPU では一度に 1 つのプロセスを実行できます。プロセスの数が vCPU の数を超えると、プロセスはキューイングを開始します。キューイングが増加すると、データベースのパフォーマンスが低下します。DB 負荷が [Max vCPU (最大 vCPU)] ラインをしばしば超過し、プライマリ待機状態が CPU である場合、CPU が過負荷になっています。この場合、インスタンスへの接続を抑制したり、CPU ロードの高い SQL クエリを調整したり、より大きなインスタンスクラスを検討する必要があります。待機状態の高い一貫したインスタンスは、解決するボトルネックまたはリソースの競合問題がある可能性があることを示します。これは、DB ロードが最大 vCPU ラインを超えていない場合にも該当します。

Amazon RDS DB エンジンとインスタンスクラスでサポートされている Performance Insights

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までにニーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロッ

ク診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

次の表に、Performance Insights をサポートしている Amazon RDS DB エンジンを示します。

> **Note**
>
> Amazon Aurora については、Amazon Aurora ユーザーガイドの Performance Insights でサポートされている DB エンジンを参照してください。

Amazon RDS DB エンジン	サポート対象のエンジンバージョンとリージョン	インスタンスクラスに関する制限
Amazon RDS for MariaDB	RDS for MariaDB に関する Performance Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	Performance Insights は、 次のインスタンスクラスではサポートされていません。 • db.t2.micro • db.t2.small • db.t3.micro •

Amazon RDS DB エンジン	サポート対象のエンジンバージョンとリージョン	インスタンスクラスに関する制限
		db.t3.small • db.t4g.micro • db.t4g.small
RDS for MySQL	RDS for MySQL に関する Performance Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	Performance Insights は、 次のインスタンスクラスではサポートされていません。 • db.t2.micro • db.t2.small • db.t3.micro • db.t3.small • db.t4g.micro • db.t4g.small
Amazon RDS for Microsoft SQL Server	RDS for SQL Server に関する Performance Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし

Amazon RDS DB エンジン	サポート対象のエンジンバージョンとリージョン	インスタンスクラスに関する制限
Amazon RDS for PostgreSQL	RDS for PostgreSQL に関する Performance Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし
Amazon RDS for Oracle	RDS for Oracle に関する Performance Insights のバージョンとリージョンの可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	該当なし

Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能

次の表に、Performance Insights 機能をサポートしている Amazon RDS DB エンジンを示します。

機能	料金階層	サポートされるリージョン	サポートされる DB エンジン	サポートされるインスタンスクラス
Performance Insights の SQL 統計	すべて	すべて	すべて	すべて
Amazon RDS の Performance Insights ダッシュボードを使	すべて	すべて	RDS for Oracle	すべて

機能	料金階層	サポートされるリージョン	サポートされるDB エンジン	サポートされるインスタンスクラス
用した Oracle 実行プランの分析				
一定期間のデータベースパフォーマンスの分析	有料階層のみ	すべて	RDS for PostgreSQL	すべて

機能	料金階層	サポートされるリージョン	サポートされるDB エンジン	サポートされるインスタンスクラス
Performance Insights のプロアクティブ推奨事項の表示	有料階層のみ	• 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アジアパシフィック (ムンバイ) • アジアパシフィック (ソウル) • アジアパシフィック (シンガポール) • アジアパシフィック (シドニー) • アジアパシフィック (東京) • カナダ (中部) • 欧州 (フランクフルト) • 欧州 (アイルランド) • 欧州 (ロンドン)	すべて	すべて

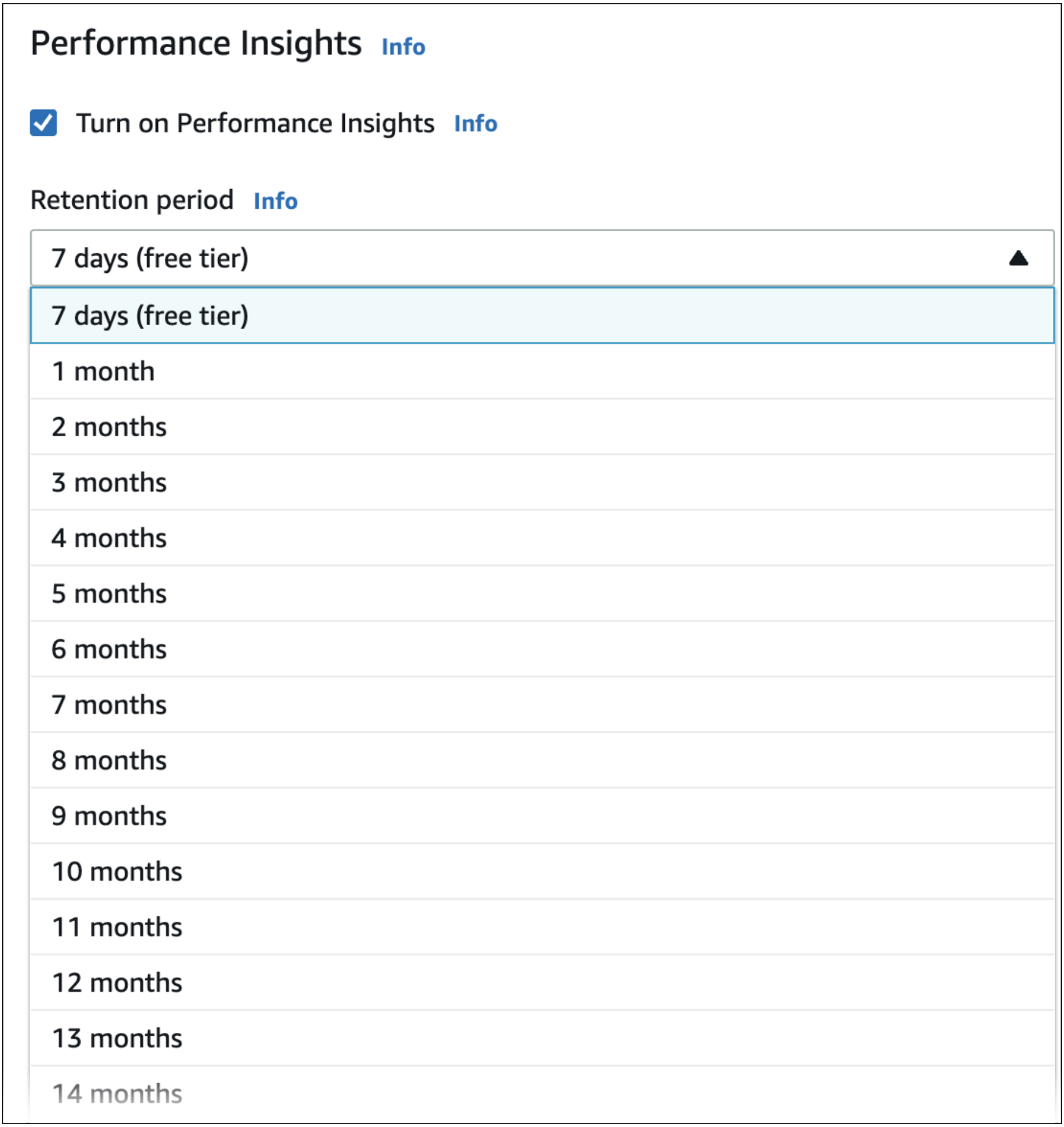
機能	料金階層	サポートされるリージョン	サポートされるDB エンジン	サポートされるインスタンスクラス
		• 欧州 (パリ) • 欧州 (ストックホルム) • 南米 (サンパウロ)		

Performance Insights の料金とデータ保持

Performance Insights には、7 日間のパフォーマンスデータ履歴と毎月 100 万件の API リクエストが、デフォルトで含まれています。また、より長い保持期間を購入することもできます。料金情報の詳細については、「Performance Insights の料金」を参照してください。

RDS コンソールでは、Performance Insights データの保持期間を次の中から選択できます。

- デフォルト (7 日)
- ***n*** か月 (***n*** は 1 ~ 24 の数値)



AWS CLI を使用して保持期間を設定する方法については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。

> **Note**
>
> Performance Insights が有効な DB インスタンスまたはマルチ AZ DB クラスターを停止しても、データ保持には影響しません。DB インスタンスまたはマルチ AZ DB クラスターが停止している間、Performance Insights はデータを収集しません。

Amazon RDS の Performance Insights の有効化と無効化

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
> Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までにニーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
> アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
> CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うことができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

DB インスタンスまたはマルチ AZ DB クラスターの作成時に、Performance Insights をオンにすることができます。必要に応じて、後でコンソールから DB インスタンスを変更することでオフにできます。Performance Insights を有効化または無効化した場合も、ダウンタイム、再起動、フェイルオーバーが発生することはありません。

> **Note**
>
> Performance Schema は、 Amazon RDS for MariaDB または MySQL で使用される、オプションのパフォーマンスツールです。Performance Schema のオンとオフを切り替える場合は、再起動する必要があります。ただし、Performance Insights のオンとオフを切り替えた場合は、再起動する必要はありません。詳細については、「Amazon RDS for MariaDB または MySQL における Performance Insights のPerformance Schema の概要」を参照してください。

Performance Insights エージェントは DB ホストの限られた CPU とメモリを消費します。DB の ロードが高い場合、エージェントはデータ収集の頻度を下げることでパフォーマンスへの影響を抑えます。

Console

コンソールでは、DB インスタンスまたはマルチ AZ DB クラスターの作成時または変更時に、Performance Insights のオンとオフを切り替えることができます。

DB インスタンスまたはマルチ AZ DB クラスター の作成時に Performance Insights のオンとオフを切り替える

新しい DB インスタンスまたはマルチ AZ DB クラスターを作成した後、Amazon RDS では Performance Insights がデフォルトで有効になります。Performance Insights をオフにするには、[データベースインサイト - スタンダード] オプションを選択し、[Performance Insights を有効化] オプションの選択を解除します。

詳細については、以下のトピックを参照してください。

- DB インスタンスを作成するには、「Amazon RDS DB インスタンスの作成」の DB エンジンの手順に従ってください。
- マルチ AZ DB クラスターを作成するには、「Amazon RDS 用のマルチ AZ DB クラスターの作成」の DB エンジンの手順に従ってください。

次のスクリーンショットは [Performance Insights] セクションを示しています。

[Performance Insights の有効化] を選択すると、次のオプションがあります。

- 保有 (Database Insights のスタンダードモードのみ) – Performance Insights データを保持する時間。保持期間の設定は、デフォルト (7 日間) です。パフォーマンスデータをさらに長期間保持するには、1 ~ 24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。
- AWS KMS key –を指定します。。AWS KMS keyPerformance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Performance Insights 用の AWS KMS ポリシーの変更」を参照してください。

DB インスタンスまたはマルチ AZ DB クラスターの変更時に Performance Insights のオン/オフを切り替える

コンソールでは、DB インスタンスまたはマルチ AZ DB クラスターを変更して Performance Insights を管理できます。

コンソールを使用して DB インスタンスまたはマルチ AZ DB クラスターの Performance Insights を管理するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベース] をクリックします。

3. DB インスタンスまたはマルチ AZ DB クラスターを選択し、[変更] を選択します。
4. Performance Insights を有効にするには、[Performance Insights を有効にする] を選択します。Performance Insights をオフにするには、[データベースインサイト - スタンダード] オプションを選択し、[Performance Insights を有効化] オプションの選択を解除します。

   [Performance Insights の有効化] を選択すると、次のオプションがあります。

   - 保有 (Database Insights のスタンダードモードのみ) – Performance Insights データを保持する時間。保持期間の設定は、デフォルト (7 日間) です。パフォーマンスデータをさらに長期間保持するには、1 ~ 24 か月を指定します。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。
   - AWS KMS key - KMS キーを指定します。Performance Insights は、潜在的に機密性の高いすべてのデータを KMS キーを使用して暗号化します。データは、転送中と不使用時のいずれも暗号化されます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
5. [Continue] を選択します。
6. [変更のスケジュール] で、[今すぐ適用] を選択します。次にスケジュールされたメンテナンスウィンドウで [Apply] (適用) を選択すると、インスタンスではこの設定が無視され、Performance Insights が直ちにオンになります。
7. [インスタンスの変更] を選択します。

AWS CLI

create-db-instance AWS CLI コマンドを使用する場合は、`--enable-performance-insights` を指定して Performance Insights をオンにし、`--database-insights-mode` を `advanced` または `standard` に設定します。Performance Insights をオフにするには、`--no-enable-performance-insights` を指定し、`database-insights-mode` を `standard` に設定します。

以下の AWS CLI コマンドを使用してこれらの値を指定することもできます。

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

CLI で Performance Insights をオンにする際に、`--performance-insights-retention-period` オプションを使用して Performance Insights のデータを保持する日数を指定できます (オプション)。7、*`month`* * 31 (*`month`* は 1 ~ 23 の範囲の数値)、または 731 を指定できます。例えば、パフォーマンスデータを 3 か月間保持する場合は、93 (3 * 31) を指定します。デフォルトは 7 日間です。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。

次の例では、`sample-db-cluster` で Performance Insights をオンにして、Performance Insights のデータの保持期間を 93 日間 (3 か月) に指定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Windows の場合:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

94 日などの有効な値ではない保持期間を指定すると、RDS はエラーを発行します。

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> Performance Insights がクラスターレベルで管理されていない DB クラスター内のインスタンスに対してのみ、Performance Insights を切り替えることができます。

RDS API

Amazon RDS API 操作の CreateDBInstance オペレーションを使用して 新しい DB インスタンスを作成する場合、`EnablePerformanceInsights` を `True` に設定して、Performance Insights をオンにします。Performance Insights をオフにするには、`EnablePerformanceInsights` を `False` に設定し、`DatabaseInsightsMode` を `standard` に設定します。

以下の API オペレーションを使用して `EnablePerformanceInsights` 値を指定することもできます。

- CreateDBCluster (マルチ AZ DB クラスター)
- ModifyDBCluster (マルチ AZ DB クラスター)
- ModifyDBInstance
- CreateDBInstanceReadReplica
- RestoreDBInstanceFromS3

Performance Insights をオンにする際に、`PerformanceInsightsRetentionPeriod` パラメータを使用して Performance Insights のデータを保持する期間を日数で指定できます (オプション)。7、*`month`* * 31 (*`month`* は 1 ~ 23 の範囲の数値)、または 731 を指定できます。例えば、パフォーマンスデータを 3 か月間保持する場合は、93 (3 * 31) を指定します。デフォルトは 7 日間です。保持期間の詳細については、「Performance Insights の料金とデータ保持」を参照してください。

Amazon RDS for MariaDB または MySQL における Performance Insights のPerformance Schema の概要

Performance Schema は、Amazon RDS for MariaDB または MySQL ランタイムのパフォーマンスを低い詳細レベルでモニタリングするオプション機能です。Performance Schema は、データベースのパフォーマンスへの影響を最小限に抑えるように設計されています。Performance Insights は、Performance Schema の有無に関係なく使用できる独立した機能です。

トピック

- Performance Schema の概要
- Performance Insights と Performance Schema
- Performance Insights による Performance Schema の自動管理
- Performance Schema の再起動による影響

- Performance Insights が Performance Schema を管理しているかどうかの確認
- Amazon RDS for MariaDB または MySQL における Performance Schema の有効化

Performance Schema の概要

Performance Schema は、MariaDB および MySQL データベースのイベントをモニタリングします。イベントとは、時間を消費し、タイミング情報を収集できるように実装されたデータベースサーバーアクションです。イベントの例には、以下のようなものがあります。

- 関数呼び出し
- オペレーティングシステムの待機
- SQL 実行のステージ
- SQL ステートメントのグループ

`PERFORMANCE_SCHEMA` ストレージエンジンは、Performance Schema 機能を実装するためのメカニズムです。このエンジンは、データベースのソースコード内の計測を使用してイベントデータを収集します。エンジンは、イベントを `performance_schema` データベースのメモリ専用テーブルに保存します。他のテーブルにクエリを実行するのと同様に、`performance_schema` をクエリできます。詳細については、MySQL リファレンスマニュアルの「MySQL Performance Schema」を参照してください。

Performance Insights と Performance Schema

Performance Insights と Performance Schema は別々の機能ですが、両者は関連しています。Amazon RDS for MariaDB または MySQL の Performance Insights の動作は、Performance Schema がオンになっているかどうか、およびオンになっている場合は、Performance Insights が Performance Schema を自動的に管理するかどうかによって異なります。次の表は、動作の説明です。

Performance Schema がオンになっている	Performance Insights 管理モード	Performance Insights の動作
はい	自動	• 詳細な低レベルのモニタリング情報を収集します。 • アクティブなセッションメトリクスを毎秒収集します。

Performance Schema がオンになっている	Performance Insights 管理モード	Performance Insights の動作
		• DB ロードを詳細な待機イベントごとに分類して表示し、ボトルネックの特定に使用できます。
はい	手動	• 待機イベントと SQL ごとのメトリクスを収集します • アクティブなセッションメトリクスを毎秒収集します。 • 挿入や送信などのユーザー状態をレポートしますが、ボトルネックの特定には役立ちません。
いいえ	該当なし	• 待機イベント、SQL ごとのメトリクス、またはその他の詳細な低レベルのモニタリング情報を収集しません。 • アクティブなセッションメトリックスを毎秒ではなく 5 秒ごとに収集します。 • 挿入や送信などのユーザー状態をレポートしますが、ボトルネックの特定には役立ちません。

Performance Insights による Performance Schema の自動管理

Performance Insights を有効にした状態で Amazon RDS for MariaDB または MySQL DB インスタンスを作成すると、Performance Schema も有効になります。この場合、Performance Insights は Performance Schema パラメータを自動的に管理します。この設定を推奨します。

Performance Insights が Performance Schema を自動的に管理するとき、`performance_schema` の Source は `System default` です。

> **Note**
>
> t4g.medium インスタンスクラスでは、パフォーマンススキーマの自動管理はサポートされていません。

performance_schema パラメータの値を手動で変更し、後で自動管理に戻す方法については、「Amazon RDS for MariaDB または MySQL における Performance Schema の有効化」を参照してください。

> **Important**
>
> Performance Insights で Performance Schema を有効にしても、パラメータグループ値は変更されません。ただし、値は実行中の DB インスタンスで変更されます。変更された値を表示する唯一の方法は、SHOW GLOBAL VARIABLES コマンドを実行することです。

Performance Schema の再起動による影響

Performance Insights と Performance Schema は、DB インスタンスの再起動の要件が異なります。

Performance Schema

この機能をオンまたはオフにするには、DB インスタンスを再起動する必要があります。

Performance Insights

この機能をオンまたはオフにするために、DB インスタンスを再起動する必要はありません。

Performance Schema が現在有効になっていない場合、DB インスタンスを再起動せずに Performance Insights を有効にすると、Performance Schema は有効になりません。

Performance Insights が Performance Schema を管理しているかどうかの確認

Performance Insights が、現在、サポート対象のメジャーエンジンバージョンの Performance Schema を管理しているかどうかを確認するには、次の表を参照してください。

performance_schema パラメータの設定	[Source] (ソース) 列の設定	Performance Insights が Performance Schema を管理しているかどうか
0	System default	はい
0 または 1	Modified	いいえ

次の手順では、Performance Insights が Performance Schema を自動的に管理しているかどうかを確認します。

Performance Insights が Performance Schema を自動管理しているかどうかを確認するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [パラメータグループ] を選択します。
3. DB インスタンスのパラメータグループを選択します。
4. 検索バーに **performance_schema** と入力します。
5. [ソース] がシステムデフォルト、[値] が [0] であることをチェックします。上記の設定の場合、Performance Insights は Performance Schema を自動管理します。

   上記の例では、Performance Insights は Performance Schema を自動管理していません。

Amazon RDS for MariaDB または MySQL における Performance Schema の有効化

DB インスタンスまたはマルチ AZ DB クラスターで Performance Insights がオンになっているが、現在 Performance Schema は管理していないと仮定します。Performance Insights が Performance Schema を自動管理できるようにするには、次のステップを実行します。

自動管理用 Performance Schema を設定するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [パラメータグループ] を選択します。
3. DB インスタンスまたはマルチ AZ DB クラスターのパラメータグループの名前を選択します。
4. [編集] を選択します。
5. 検索バーに **performance_schema** と入力します。
6. performance_schema パラメータを選択します。
7. [デフォルト値に設定] を選択します。
8. [値をデフォルトに設定] を選択して確定します。

9. [Save changes] (変更の保存) をクリックします。
10. DB インスタンスまたはマルチ AZ DB クラスターを再起動します。

> **Important**
>
> Performance Schema のオンとオフを切り替えるたびに、DB インスタンスまたはマルチ AZ DB クラスターを必ず再起動します。

インスタンスのパラメータの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。ダッシュボードのページの詳細については、「Performance Insights ダッシュボードを使用してメトリクスを分析する」を参照してください。MySQL パフォーマンススキーマの詳細については、MySQL ドキュメントの「MySQL Performance Schema」(8.0 の場合) および「MySQL Performance Schema」(8.4 の場合) を参照してください。

Performance Insights 用のアクセスポリシーの設定

Performance Insights にアクセスするには、プリンシパルが AWS Identity and Access Management (IAM) から適切な許可を得る必要があります。

> **Note**
>
> カスタマーマネージドキーで Performance Insights を使用するには、ユーザーに AWS AWS KMS キーの `kms:Decrypt` および `kms:GenerateDataKey` アクセス許可を付与します。

以下の方法を使用して Performance Insights にアクセスします。

- 読み取り専用アクセスのための `AmazonRDSPerformanceInsightsReadOnly` 管理ポリシーをアタッチする
- Performance Insights API のすべてのオペレーションにアクセスするための `AmazonRDSPerformanceInsightsFullAccess` 管理ポリシーをアタッチする
- 特定のアクセス許可を持つカスタム IAM ポリシーを作成する
- 暗号化された Performance Insights データの AWS KMS アクセス許可を設定する
- リソースレベルのアクセス許可を使用してきめ細かなアクセスを設定する
- タグベースのアクセスコントロールを使用して、リソースタグを介してアクセス許可を管理する

AmazonRDSPerformanceInsightsReadOnly ポリシーの IAM プリンシパルへのアタッチ

AmazonRDSPerformanceInsightsReadOnly は Amazon RDS Performance Insights API のすべての読み取り専用オペレーションへのアクセス許可を付与する AWS マネージドポリシーです。

アクセス許可セットまたはロールに AmazonRDSPerformanceInsightsReadOnly をアタッチする場合は、次の CloudWatch アクセス許可もアタッチする必要があります。

- GetMetricStatistics
- ListMetrics
- GetMetricData

これらのアクセス許可により、受信者は他のコンソール機能で Performance Insights を使用できます。

Amazon CloudWatch のアクセス許可の詳細については、「Amazon CloudWatch の許可リファレンス」を参照してください。

AmazonRDSPerformanceInsightsReadOnly の詳細については、「AWS マネージドポリシー: AmazonRDSPerformanceInsightsReadOnly」を参照してください。

AmazonRDSPerformanceInsightsFullAccess ポリシーの IAM プリンシパルへのアタッチ

AmazonRDSPerformanceInsightsFullAccess は Amazon RDS Performance Insights API のすべての操作へのアクセス許可を付与する AWS マネージドポリシーです。

アクセス許可セットまたはロールに AmazonRDSPerformanceInsightsFullAccess をアタッチする場合は、次の CloudWatch アクセス許可もアタッチする必要があります。

- GetMetricStatistics
- ListMetrics
- GetMetricData

これらのアクセス許可により、受信者は他のコンソール機能で Performance Insights を使用できます。

Amazon CloudWatch のアクセス許可の詳細については、「Amazon CloudWatch の許可リファレンス」を参照してください。

AmazonRDSPerformanceInsightsFullAccess の詳細については、「AWS 管理ポリシー: AmazonRDSPerformanceInsightsFullAccess」を参照してください。

Performance Insights 用のカスタム IAM ポリシーの作成

AmazonRDSPerformanceInsightsReadOnly または AmazonRDSPerformanceInsightsFullAccess ポリシーを持たないユーザーの場合、ユーザーマネージド IAM ポリシーを作成または変更することによって、Performance Insights へのアクセス許可を付与できます。ポリシーを IAM アクセス許可セットまたはロールにアタッチすると、受取人は Performance Insights を使用できます。

カスタムポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、ポリシー を選択してください。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーの作成] ページで、[JSON] オプションを選択します。
5. 「AWS マネージドポリシーリファレンスガイド」の「JSON ポリシードキュメント」セクションに記載されている文字列をコピーして、AmazonRDSPerformanceInsightsReadOnly または AmazonRDSPerformanceInsightsFullAccess ポリシーに貼り付けます。
6. [ポリシーの確認] を選択します。
7. ポリシーの名前と (必要に応じて) 説明を入力し、[ポリシーの作成] を選択します。

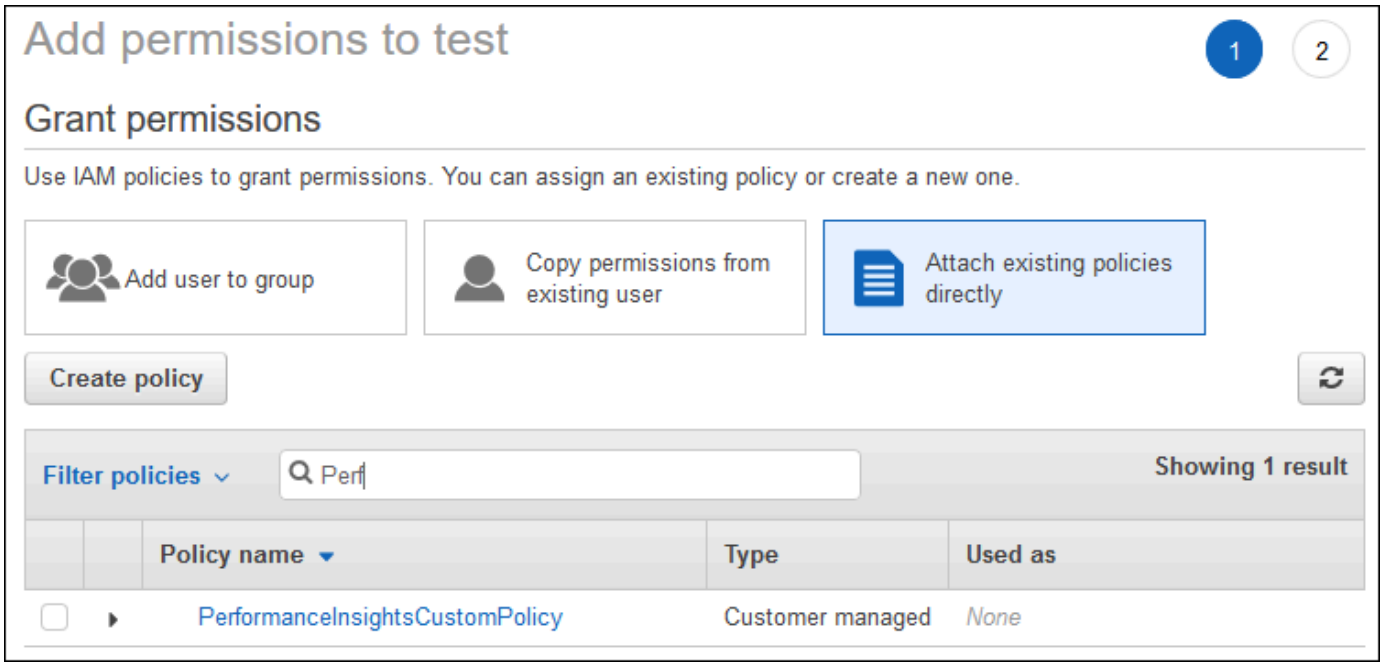
これで、そのポリシーをアクセス許可セットまたはロールにアタッチできます。次の手順では、この目的で使用できるユーザーが既に存在することを前提としています。

ポリシーをユーザーにアタッチするには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [ユーザー] を選択します。
3. リストから存在するユーザーを 1 人選択します。

> **Important**
>
> Performance Insights を使用するには、カスタムポリシーのほかに別のポリシーで、Amazon RDS へのアクセスを許可されている必要があります。例えば、`AmazonRDSPerformanceInsightsReadOnly` 事前定義ポリシーで、ユーザーに Amazon RDS への読み取り専用アクセスを許可します。詳細については、「ポリシーを使用したアクセスの管理」を参照してください。

4. [Summary] ページで、[Add permissions] を選択します。
5. [Attach existing policies directly (既存のポリシーを直接アタッチする)] を選択します。検索を行う場合は、次の画像に示すようにポリシー名の初期の数文字を入力します。



6. ポリシーを選択し、[次へ: レビュー] を選択します。
7. [アクセス権限の追加] を選択します。

Performance Insights 用の AWS KMS ポリシーの変更

Performance Insights は、AWS KMS key を使用して機密データを暗号化します。API またはコンソールを通じて Performance Insights を有効にする場合は、次のいずれかを実行します。

- デフォルト AWS マネージドキー を選択します。

Amazon RDS は、新しい DB インスタンスに AWS マネージドキー を使用します。Amazon RDS は、AWS アカウント に AWS マネージドキー を作成します。AWS アカウント には、AWS リージョン ごとに Amazon RDS の AWS マネージドキー が別々にあります。

- カスタマーマネージドキーを選択します。

  カスタマーマネージドキーを指定する場合、Performance Insights API を呼び出すアカウント内のユーザーは、KMS キーに対する `kms:Decrypt` および `kms:GenerateDataKey` アクセス許可が必要です。IAM ポリシーを使用して、これらのアクセス許可を設定できます。ただし、KMS キーポリシーを使用してこれらのアクセス許可を管理することをお勧めします。詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS でのキーポリシー」を参照してください。

Example

次の例では、KMS キーポリシーにステートメントを追加する方法を示します。これらのステートメントは、Performance Insights へのアクセスを許可します。KMS キーの使用方法によっては、いくつかの制限を変更することもできます。ポリシーにステートメントを追加する前に、すべてのコメントを削除してください。

JSON

```
{
    "Version":"2012-10-17",
    "Id" : "your-policy",
    "Statement" : [
        {
            "Sid" : "AllowViewingRDSPerformanceInsights",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/Role1"
                ]
                },
             "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
                ],
            "Resource": "*",
            "Condition" : {
```

```
            "StringEquals" : {
                "kms:ViaService" : "rds.us-east-1.amazonaws.com"
                },
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContext:aws:pi:service": "rds",
                "kms:EncryptionContext:service": "pi",
                "kms:EncryptionContext:aws:rds:db-id": "db-
AAAAABBBBBCCCCDDDDDEEEEE"
                }
            }
        }
    ]
}
```

Performance Insights が AWS KMS カスタマー管理キーを使用する方法

Performance Insights は、カスタマー管理型のキー (CMK) を使用して機密データを暗号化します。Performance Insights を有効にすると、API を介して AWS KMS キーを提供できます。Performance Insights は、このキーの AWS KMS アクセス許可を作成します。キーを使用して、機密データを処理するために必要な操作を実行します。機密データには、ユーザー、データベース、アプリケーション、SQL クエリテキストなどのフィールドが含まれます。Performance Insights により、データは保存中も転送中も暗号化されたままになります。

Performance Insights IAM と AWS KMS の連携

IAM は特定の API にアクセス権限を付与します。Performance Insights には以下のパブリック API があり、IAM ポリシーを使用して制限できます。

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

機密データを取得するには、以下の API リクエストを使用できます。

- `DescribeDimensionKeys`

- GetDimensionKeyDetails
- GetResourceMetrics

API を使用して機密データを取得する場合、Performance Insights は呼び出し元の認証情報を利用します。このチェックにより、機密データへのアクセスが KMS キーにアクセスできるユーザーに限定されます。

これらの API を呼び出すときは、IAM ポリシーを通じて API を呼び出す権限と、AWS KMS キーポリシーを通じて kms:decrypt アクションを呼び出す権限が必要となります。

GetResourceMetrics API は、機密データと非機密データの両方を返すことができます。リクエストパラメータにより、レスポンスに機密データを含めるかどうかが決定されます。リクエストのフィルターパラメーターまたは group-by パラメーターのいずれかに機密ディメンションが含まれている場合、API は機密データを返します。

GetResourceMetrics API で使用できるディメンションの詳細については、DimensionGroup を参照してください。

Example例

次の例では、db.user グループの機密データをリクエストしています:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
```

```
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

次の例では、db.load.avg メトリクスの非機密データをリクエストしています:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Performance Insights への詳細なアクセス許可の付与

詳細なアクセスコントロールを使用して、Performance Insights データへの追加のアクセスコントロールを適用することができます。このアクセスコントロールは、Performance Insights の

GetResourceMetrics、DescribeDimensionKeys、GetDimensionKeyDetails アクションの個々のディメンションへのアクセスを許可または拒否できます。詳細なアクセスコントロールを使用するには、条件キーを使用して IAM ポリシーでディメンションを指定します。アクセスの評価は、IAM ポリシーの評価ロジックに基づきます。詳細については、「IAM ユーザーガイド」の「ポリシーの評価論理」を参照してください。IAM ポリシーステートメントでディメンションが指定されていない場合、ステートメントは指定されたアクションのすべてのディメンションへのアクセスをコントロールします。使用可能なディメンションのリストについては、「DimensionGroup」を参照してください。

認証情報でアクセスが許可されているディメンションを確認するには、ListAvailableResourceDimensions の AuthorizedActions パラメータを使用してアクションを指定します。AuthorizedActions の許容値は、次のとおりです。

- GetResourceMetrics
- DescribeDimensionKeys
- GetDimensionKeyDetails

例えば、AuthorizedActions パラメータに GetResourceMetrics を指定すると、ListAvailableResourceDimensions は GetResourceMetrics アクションがアクセスを許可されているディメンションのリストを返します。AuthorizedActions パラメータで複数のアクションを指定すると、ListAvailableResourceDimensions はそれらのアクションがアクセスを許可されているディメンションの交差を返します。

Example

次の例では、GetResourceMetrics および DescribeDimensionKeys アクションに指定されたディメンションへのアクセスを付与します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
```

```
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

リクエストされたディメンションのレスポンスを次に示します。

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}
```

```
// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

次の例は、ディメンションに対する 1 つの許可アクセスと 2 つの拒否アクセスを指定します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
```

```
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
```

```
                }
            }
        }
    ]
}
```

リクエストされたディメンションのレスポンスを次に示します。

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
```

```
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

Performance Insights でのタグベースのアクセスコントロールの使用

親 DB インスタンスから継承されたタグを使用して、Performance Insights メトリクスへのアクセスを制御できます。Performance Insights オペレーションへのアクセスを制御するには、IAM ポリシーを使用します。これらのポリシーは、DB インスタンスのタグをチェックしてアクセス許可を決定できます。

Performance Insights でのタグの仕組み

Performance Insights は、DB インスタンスタグを自動的に適用して Performance Insights メトリクスを承認します。DB インスタンスにタグを追加すると、すぐにそれらのタグを使用して Performance Insights データへのアクセスを制御できます。

- Performance Insights メトリクスのタグを追加または更新するには、DB インスタンスのタグを変更します。
- Performance Insights メトリクスのタグを表示するには、Performance Insights メトリクスリソースで `ListTagsForResource` を呼び出します。メトリクスに関連付けられた DB インスタンスからタグを返します。

> **Note**
>
> Performance Insights メトリクスで直接使用しようとすると、`TagResource` および `UntagResource` オペレーションはエラーを返します。

タグベースの IAM ポリシーの作成

Performance Insights オペレーションへのアクセスを制御するには、IAM ポリシーで `aws:ResourceTag` 条件キーを使用します。これらのポリシーは、yourDB インスタンスのタグをチェックします。

Example

このポリシーは、本番データベースの Performance Insights メトリクスへのアクセスを防止します。このポリシーは、`env:prod` でタグ付けされたデータベースリソースの Performance Insights での `pi:GetResourceMetrics` オペレーションを拒否します。

```
{
   "Version": "2012-10-17",
   "Statement": [
```

```
        {
            "Effect": "Deny",
            "Action": "pi:GetResourceMetrics",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/env": "prod"
                }
            }
        }
    ]
}
```

Performance Insights ダッシュボードを使用してメトリクスを分析する

> **Important**
>
> AWS は、Performance Insights のサポート終了日を 2026 年 7 月 31 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールでのエクスペリエンスのサポートを終了します。Performance Insights コンソールは CloudWatch Database Insights にリダイレクトされます。Database Insights のスタンダードモードでは、柔軟な保持期間 (1 ~ 24 か月) および関連する料金が、現在の Performance Insights と同等のコストで保持されます。Performance Insights API は存続し、変更はありません。Performance Insights API の料金は、CloudWatch Database Insights の料金とともに AWS 請求書に表示されます。
> Performance Insights を使用して DB インスタンスを確認し、2026 年 7 月 31 日までに二ーズに最適な Database Insights モードを選択することをお勧めします。コアモニタリングと柔軟な保持期間が必要な場合は、Database Insights のスタンダードモードにより、これまでのエクスペリエンスと料金体系が維持されます。フリートレベルのモニタリング、ロック診断、実行計画キャプチャなどの高度な機能については、「Amazon RDS で Database Insights のアドバンストモードを有効にする」を参照してください。
> アクションを実行しない場合、Performance Insights を使用する DB インスタンスは、既存の保持期間設定のまま、デフォルトで Database Insights のスタンダードモードを使用します。CloudFormation テンプレート、Terraform 設定、デプロイスクリプトは、現在と同様に動作し続けます。保持期間設定を含むすべての Performance Insights API パラメータは完全に保持されます。2026 年 7 月 31 日以降、Database Insights のアドバンストモードのみが実行プランとオンデマンド分析をサポートします。
> CloudWatch Database Insights を使用すると、データベースのフリートのデータベース負荷をモニタリングし、パフォーマンスの大規模な分析とトラブルシューティングを行うこ

とができます。Database Insights の詳細については、「CloudWatch Database Insights による Amazon RDS データベースのモニタリング」を参照してください。また、詳細を学ぶために今後開催予定のワークショップに登録してください。最新の料金情報については、「Amazon CloudWatch 料金表」を参照してください。

Performance Insights ダッシュボードには、パフォーマンスの問題を分析し、解決するのに役立つ、データベースのパフォーマンス情報が含まれます。ダッシュボードのメインページで、データベース負荷に関する情報を確認できます。待機イベントや SQL などのディメンションによって、DB のロードを「スライス」することが可能です。

Performance Insights ダッシュボード

- Performance Insights ダッシュボードの概要
- Performance Insights ダッシュボードにアクセスする
- 待機イベントによる DB 負荷の分析
- 一定期間のデータベースパフォーマンスの分析
- Performance Insights のトップ SQL タブを使用したクエリの分析
- Oracle PDB の上位負荷の分析
- Amazon RDS の Performance Insights ダッシュボードを使用した実行プランの分析

Performance Insights ダッシュボードの概要

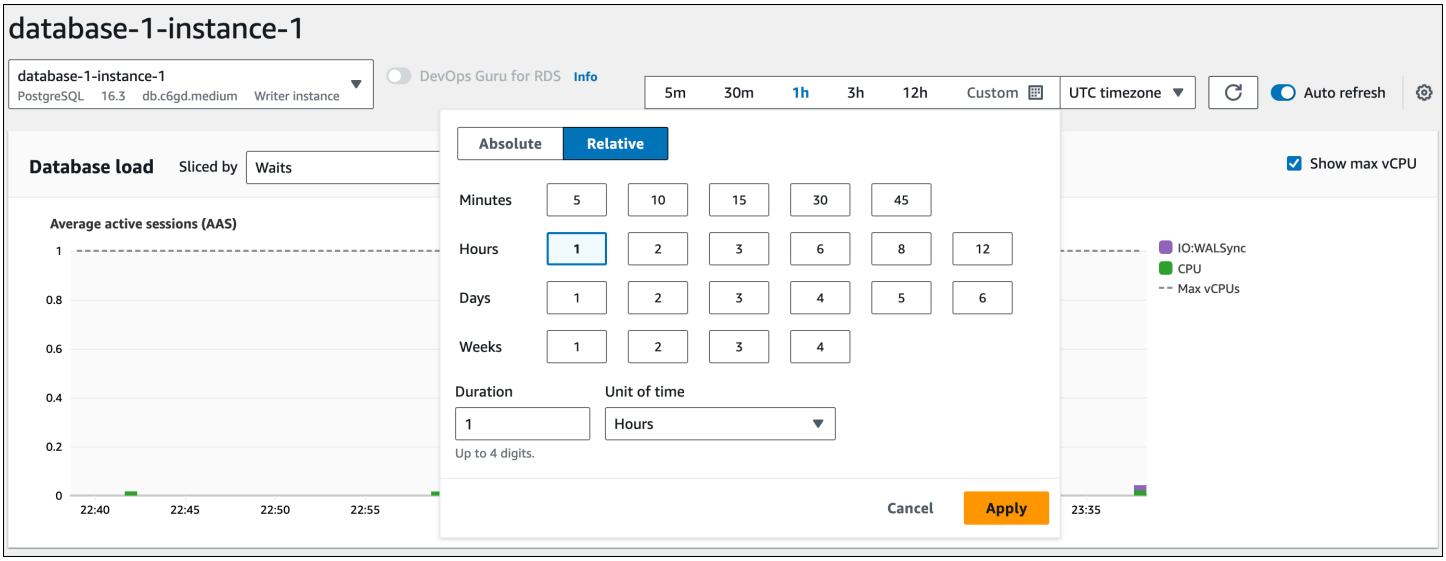
ダッシュボードは、Performance Insights を操作する最も簡単な方法です。次の例では、PostgreSQL DB インスタンスのダッシュボードを示しています。

トピック

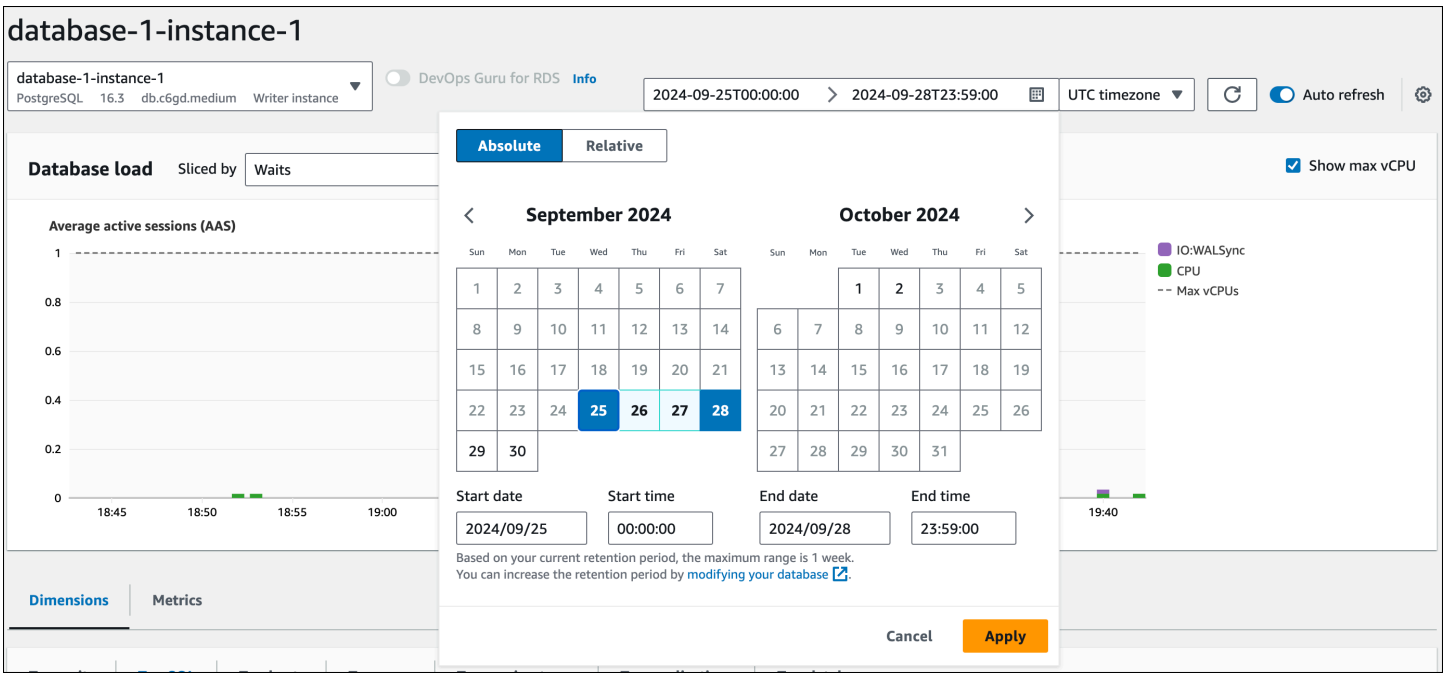
- 時間範囲フィルター
- カウンターメトリクスグラフ
- データベースロードのグラフ
- 上位のディメンションテーブル

時間範囲フィルター

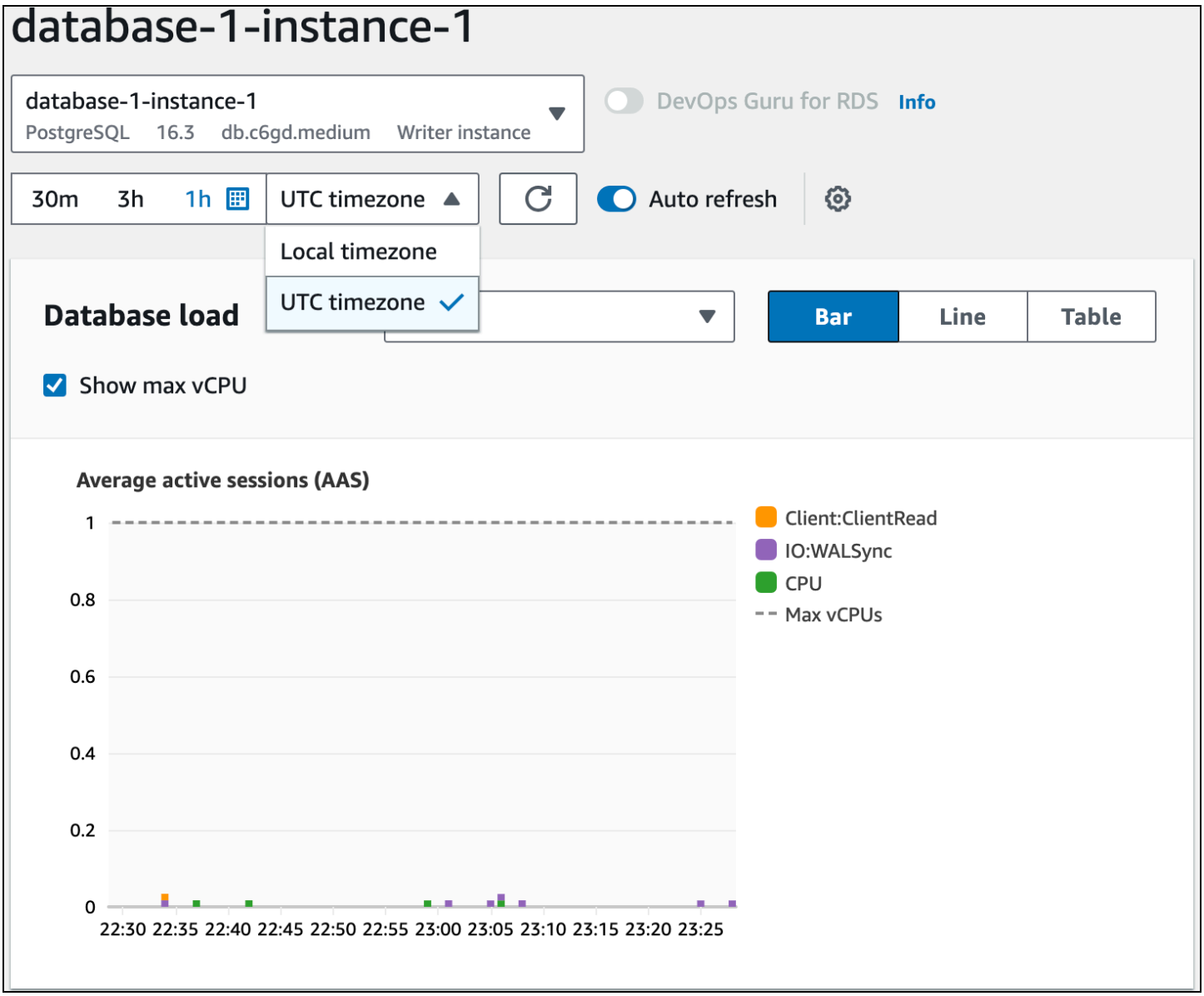
デフォルトでは、Performance Insights ダッシュボードには過去 60 分間の DB ロードが表示されます。この範囲は、最短で 5 分、最長で 2 年まで調整することができます。カスタム相対範囲を選択することもできます。



開始日時と終了日時の絶対範囲を選択できます。次の例は、24/9/25 の午前 0 時から 24/9/28 の午後 11 時 59 分までの時間範囲を示しています。



デフォルトでは、Performance Insights ダッシュボードのタイムゾーンは協定世界時 (UTC) です。ローカルタイムゾーンを選択することもできます。



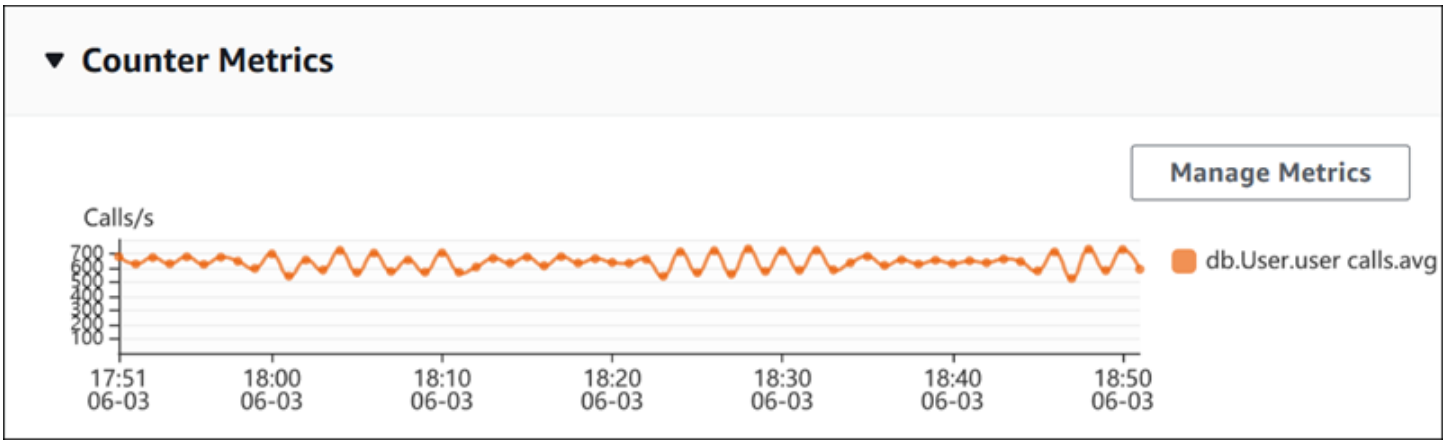
カウンターメトリクスグラフ

カウンターメトリクスを使用すると、Performance Insights ダッシュボードをカスタマイズして最大 10 個の追加グラフを含めることができます。これらのグラフは、数十種類のオペレーティングシステムとデータベースのパフォーマンスメトリクスの一部を示しています。この情報をデータベース負荷と関連付けることで、パフォーマンスの問題を特定して分析できます。

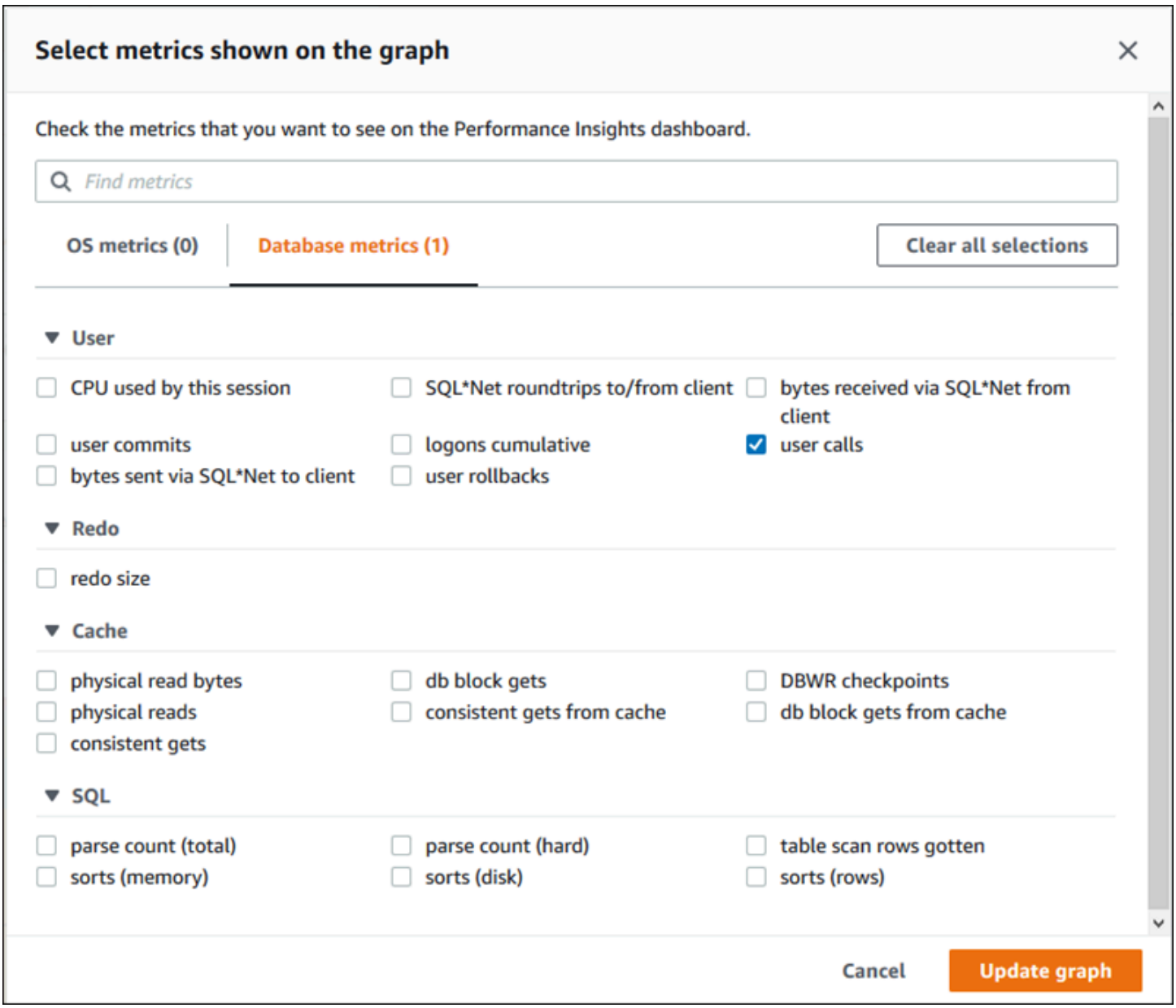
カウンターメトリクスグラフはパフォーマンスカウンターのデータを表示します。デフォルトのメトリクスは DB エンジンによって異なります。

- MySQL と MariaDB - `db.SQL.Innodb_rows_read.avg`
- Oracle - `db.User.user calls.avg`

- Microsoft SQL Server - db.Databases.Active Transactions(_Total).avg
- PostgreSQL - db.Transactions.xact_commit.avg



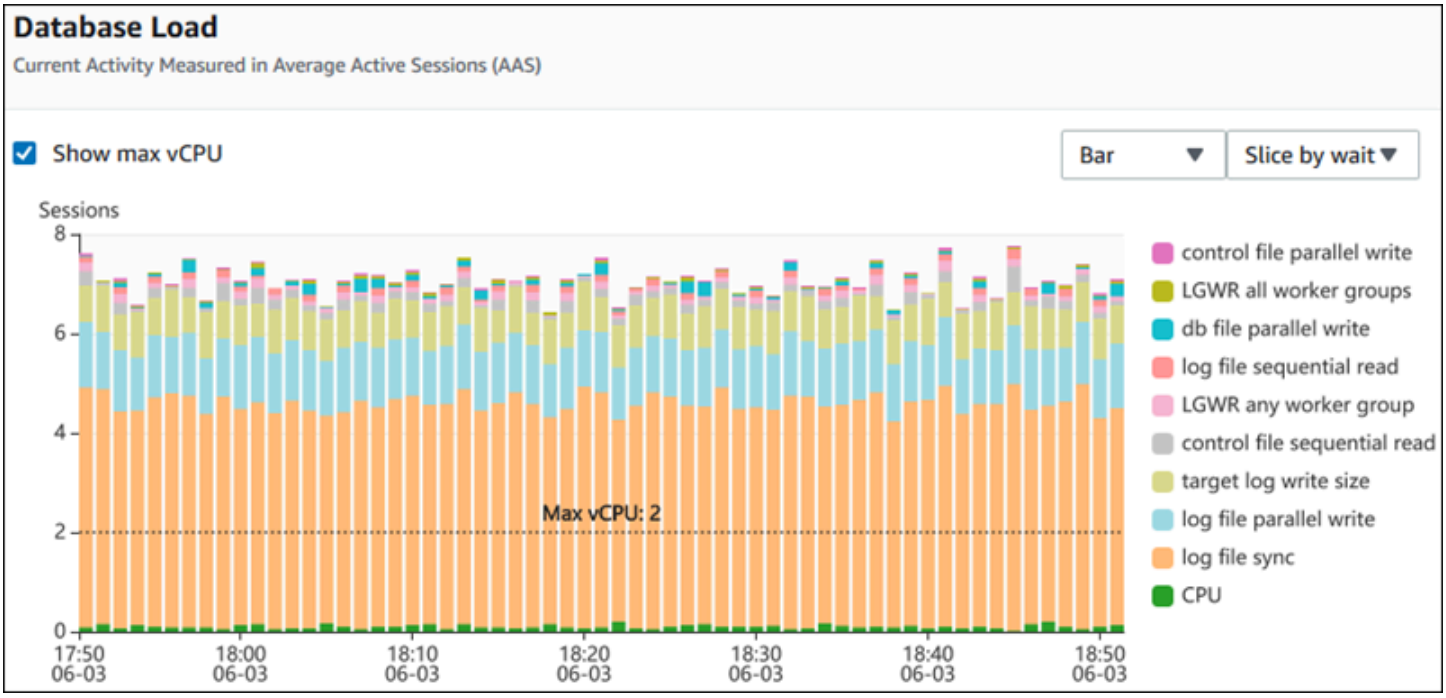
「メトリクスの管理」 を選択して、パフォーマンスカウンターを変更します。以下のスクリーンショットに示すように、複数の OS メトリクスまたはデータベースメトリクスを選択できます。メトリクスの詳細を表示するには、メトリクス名にカーソルを合わせます。



各 DB エンジンで追加できるカウンターメトリクスの詳細については、「Performance Insights カウンターメトリクス」を参照してください。

データベースロードのグラフ

データベースロードは、データベースアクティビティと DB インスタンス容量の比較結果が 最大 vCPU の折れ線グラフとして表示されます。デフォルトでは、折れ線グラフは DB ロードを単位時間あたりの平均アクティブセッションで表します。DB ロードは、待機状態でスライス (グループ化) されます。



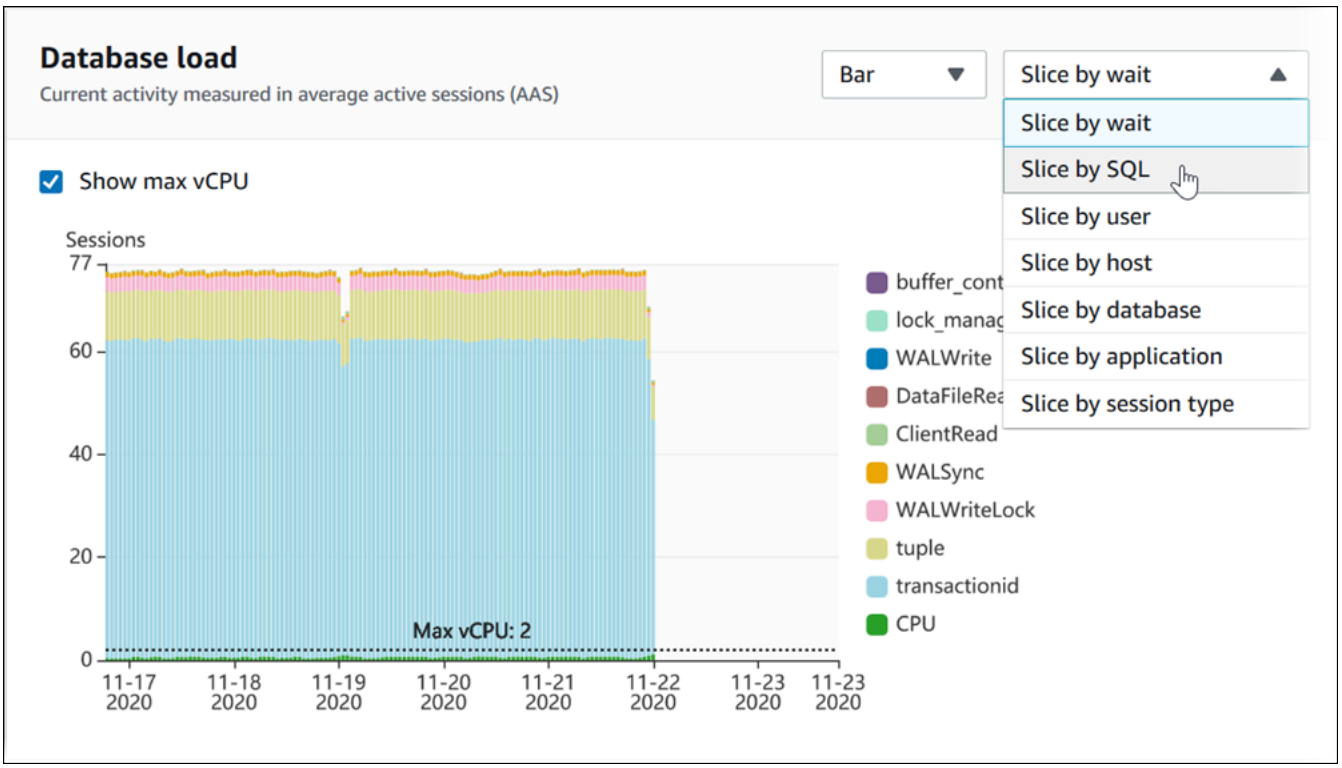
ディメンションでスライスされた DB のロード

サポートされているディメンション別にグループ化された、アクティブなセッションとして負荷を表示するように選択できます。次の表に、各エンジンでサポートされているディメンションを示します。

ディメンション	Oracle	SQL Server	PostgreSQL	MySQL
ホスト	可能	はい	はい	はい
SQL	可能	はい	はい	はい
ユーザー	可能	はい	はい	はい
待機	可能	はい	はい	はい
プラン	はい	いいえ	いいえ	いいえ
アプリケーション	いいえ	なし	はい	いいえ
データベース	いいえ	なし	はい	はい

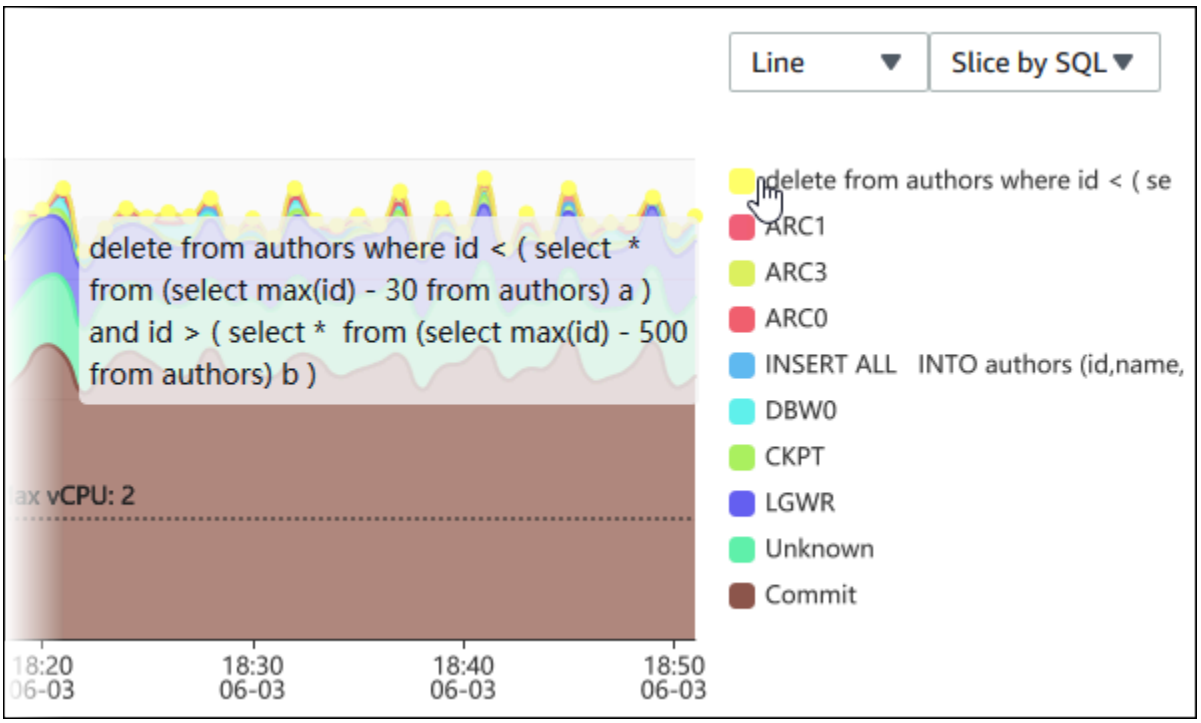
ディメンション	Oracle	SQL Server	PostgreSQL	MySQL
セッションタイプ	いいえ	なし	はい	いいえ

次の図に、PostgreSQL DB インスタンスのディメンションを示します。

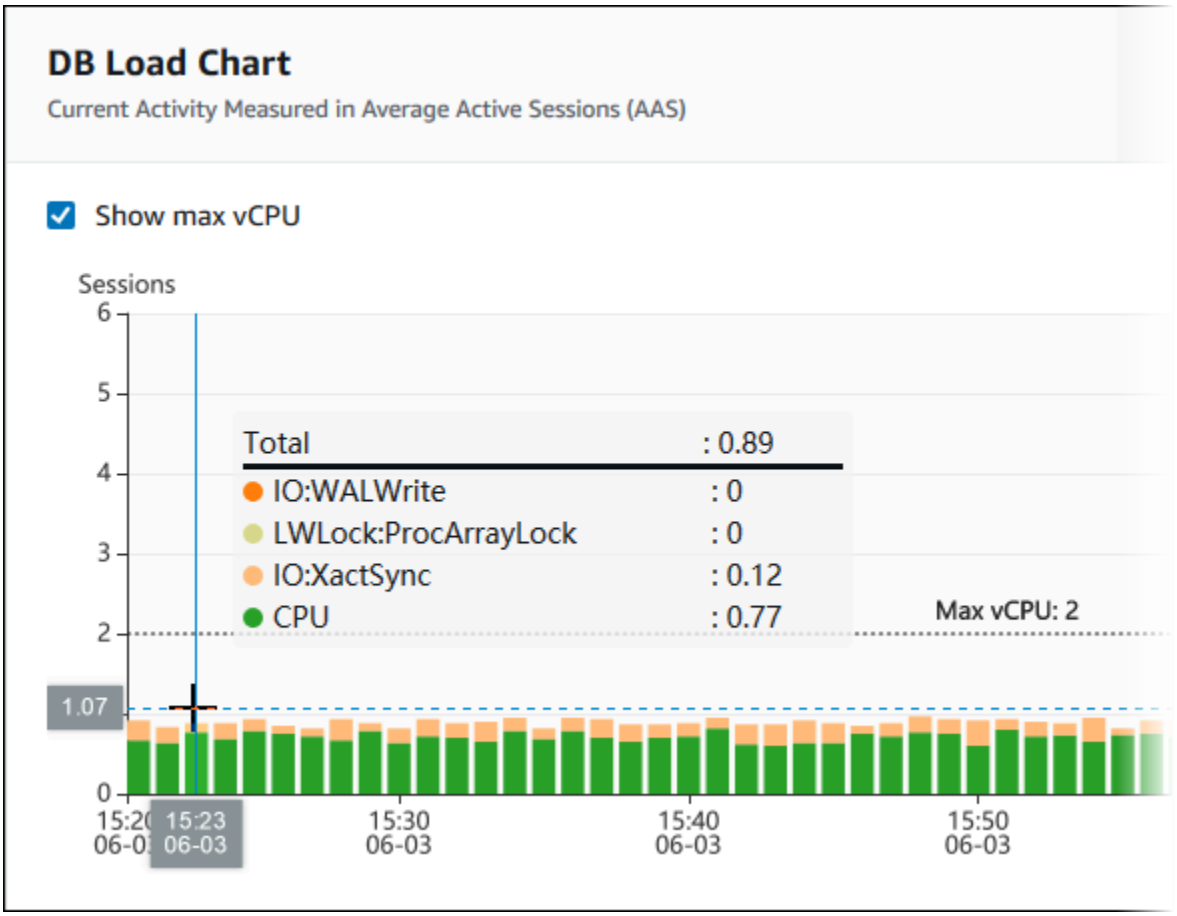


ディメンション項目に関する DB ロードの詳細

ディメンション内の DB 負荷項目の詳細を表示するには、項目名にカーソルを合わせます。次の図は、SQL ステートメントの詳細を示しています。

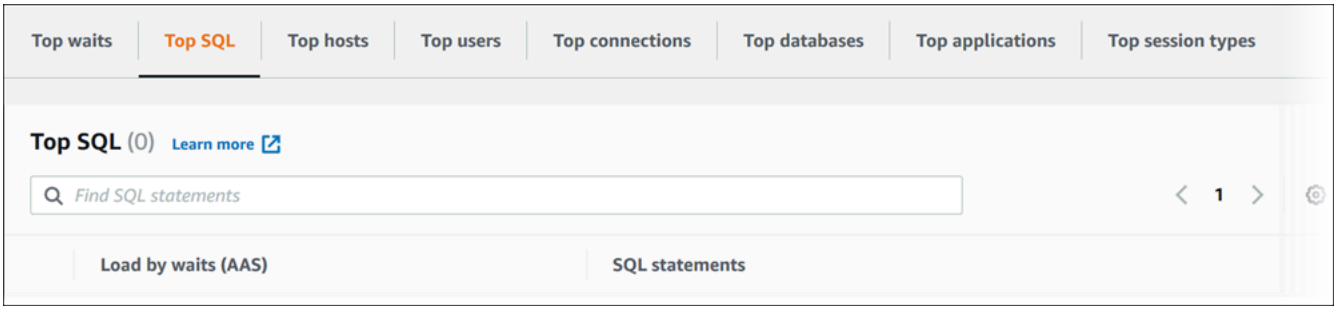


凡例で選択した期間に対する項目の詳細を表示するには、その項目にカーソルを合わせます。



上位のディメンションテーブル

上位ディメンションテーブルは、DB 負荷を異なる次元でスライスします。ディメンションとは、DB ロードの異なる特性についてのカテゴリまたは「スライス化」のことです。ディメンションが SQL の場合、上位の SQLは、DB ロードに最も貢献している SQL ステートメントを表示します。



以下のディメンションタブのいずれかを選択します。

タブ	説明	サポートされているエンジン
上位の SQL	現在実行中の SQL ステートメント	すべて
上位待機	データベースバックエンドが待っているイベント	すべて
上位ホスト	接続されているクライアントのホスト名	すべて
上位ユーザー	データベースにログインしているユーザー	すべて
上位データベース	プロキシが接続しているデータベースユーザーの名前	PostgreSQL、MySQL、MariaDB、SQL Server のみ
上位アプリケーション	データベースに接続されたアプリケーションの名前。	PostgreSQL と SQL Server のみ
上位セッションタイプ	現在のセッションのタイプ	PostgreSQL のみ

[上位の SQL] タブを使用してクエリを分析する方法を学習するには、「[トップ SQL] タブの概要」を参照してください。

Performance Insights ダッシュボードにアクセスする

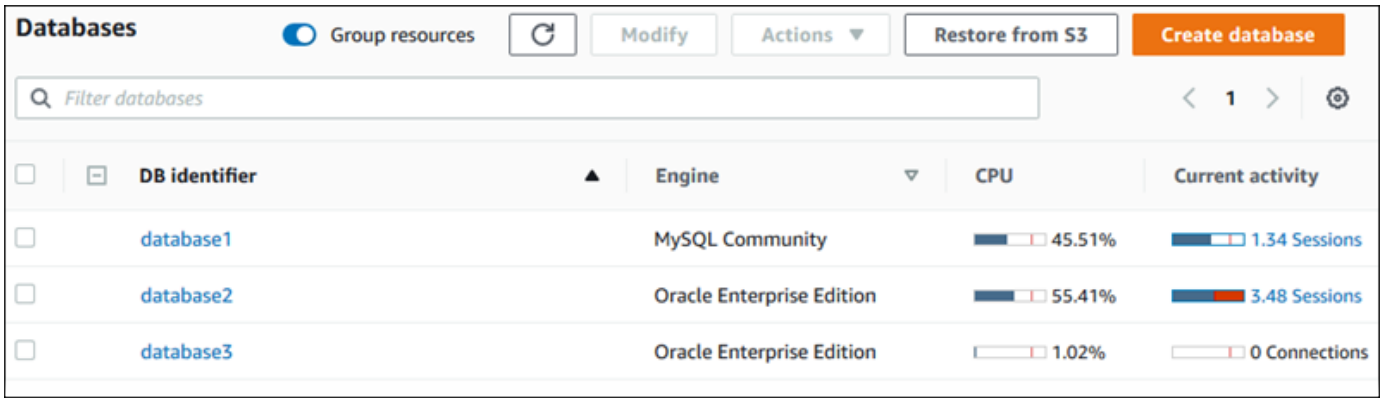
Amazon RDS のPerformance Insights ダッシュボードでは、Performance Insights と CloudWatch メトリクスの統合ビューが提供されます。

Performance Insights ダッシュボードにアクセスするには、以下の手順を使用します。

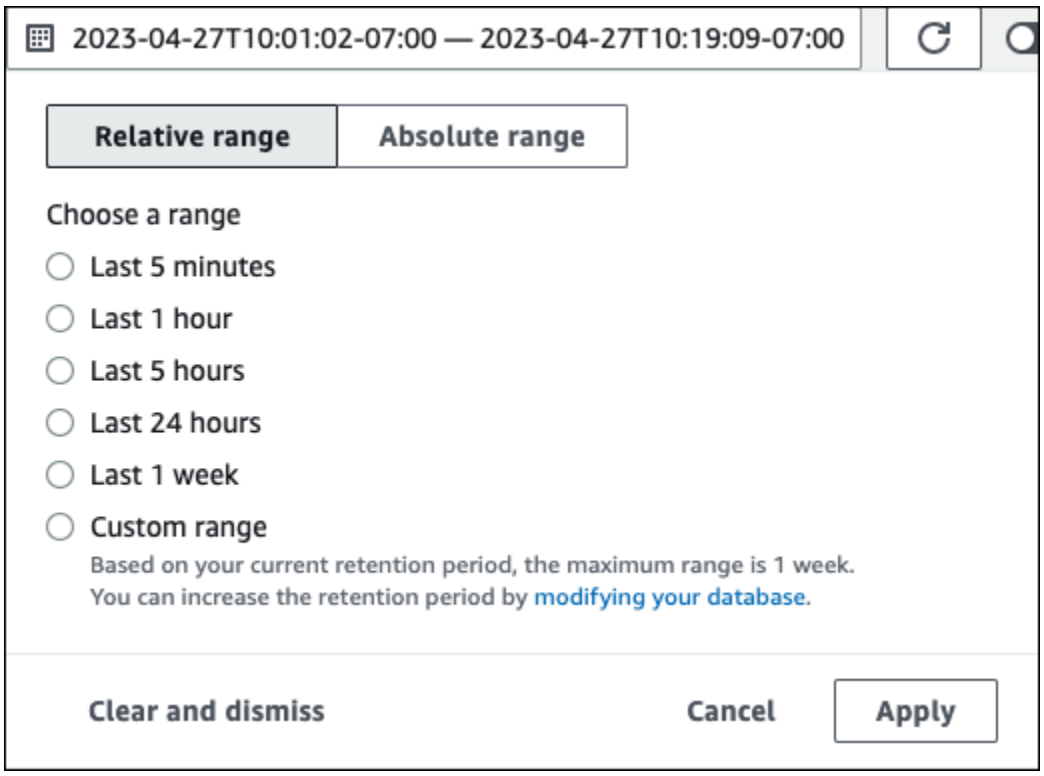
AWS マネジメントコンソールで Performance Insights ダッシュボードを表示するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

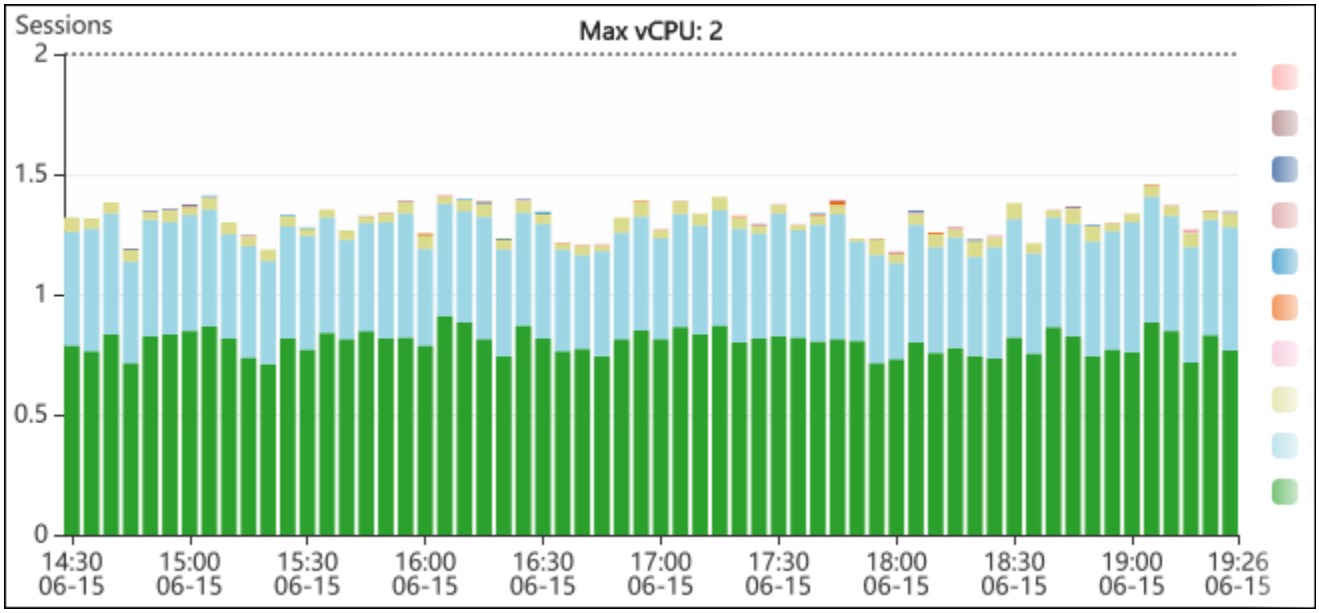
   Performance Insights を有効にした DB インスタンスでは、DB インスタンスのリストで [セッション] 項目を選択して Performance Insights ダッシュボードにアクセスすることもできます。[現在のアクティビティ] の [セッション] 項目には、直近 5 分間におけるアクティブなセッションの平均データベースロードが表示されます。負荷はバーでグラフィカルに示されます。バーが空の場合、DB インスタンスはアイドル状態です。負荷が増加すると、バーが青色で塗りつぶされます。負荷が DB インスタンスクラスのおける仮想 CPU (vCPU) の数を超えると、バーが赤色になり、ボトルネックとなる可能性があることが示されます。



4. (オプション) 日付または時間範囲を選択し、別の相対時間間隔または絶対時間間隔を指定します。これで、期間を指定して、データベースパフォーマンス分析レポートを生成できます。レポートには、特定されたインサイトと推奨事項が記載されています。詳細については、「Performance Insights でのパフォーマンス分析レポートの作成」を参照してください。

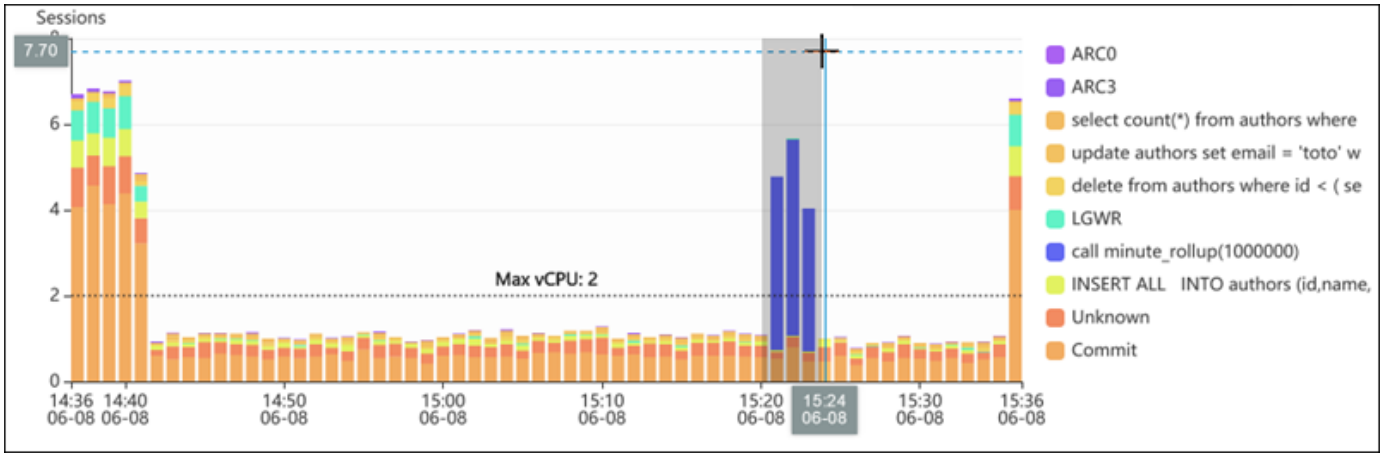


以下のスクリーンショットでは、DB 負荷の間隔は 5 時間です。

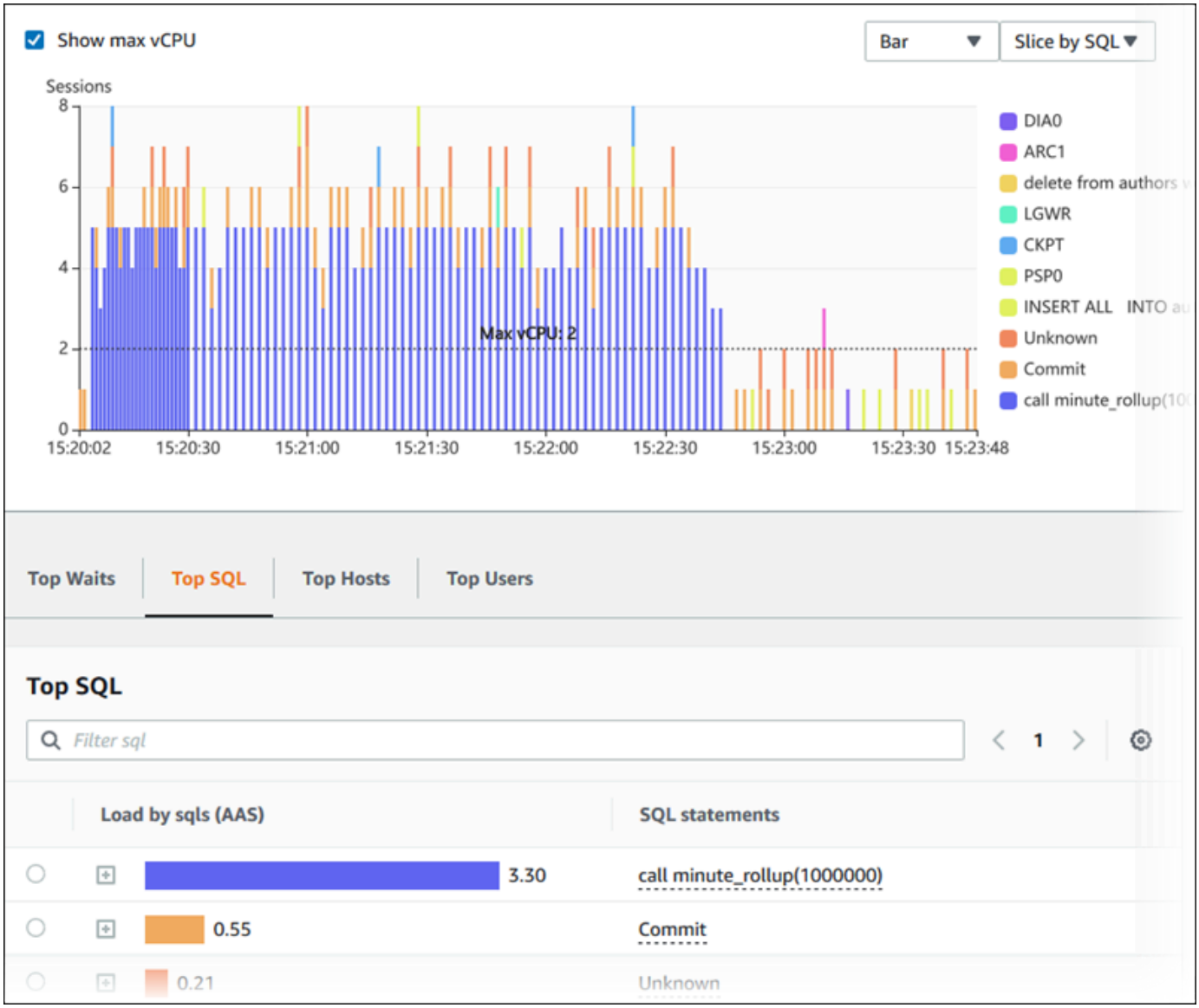


5. (オプション) DB ロードグラフの一部を拡大表示するには、スタート時間を選択し、目的の期間の最後までドラッグします。

   選択した領域が DB ロードチャートで強調表示されます。



マウスを離すと、選択した AWS リージョンの DB ロードグラフが拡大表示され、上位ディメンションのテーブルが再計算されます。



6. (オプション) データを自動的に更新するには、[自動更新] を選択します。

Performance Insights ダッシュボードが自動的に新しいデータで更新されます。更新の頻度は、表示されるデータの量によって異なります。

- 「5 分」は 10 秒ごとに更新されます。
- 「1 時間」は 5 分ごとに更新されます。
- 「5 時間」は 5 分ごとに更新されます。
- 「24 時間」は 30 分ごとに更新されます。
- 「1 週間」は 1 日ごとに更新されます。

- 「1 か月」は 1 日ごとに更新されます。

待機イベントによる DB 負荷の分析

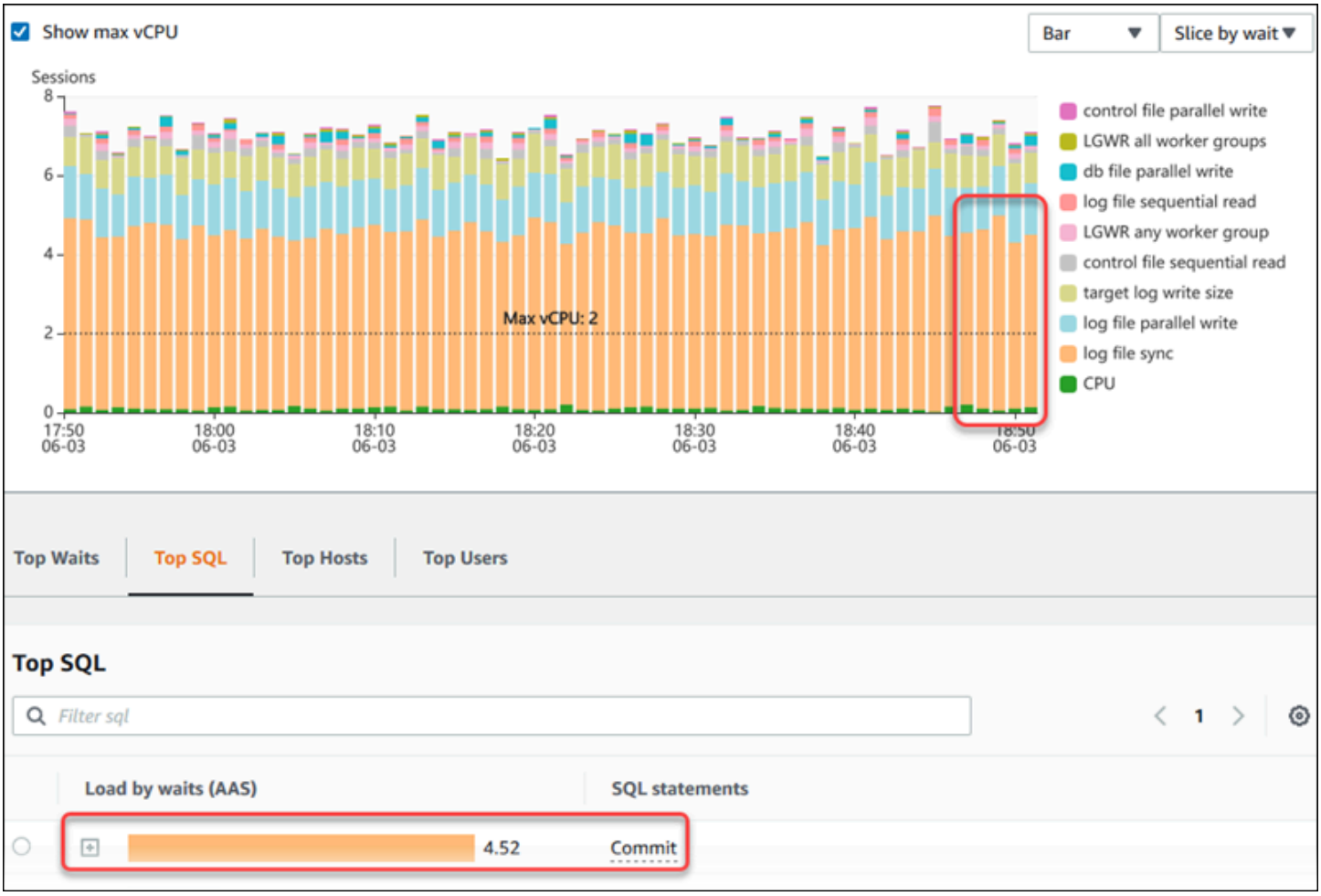
データベースロード のグラフにボトルネックが表示される場合、ロードの発生源を確認できます。これを実行するには、データベースロードグラフ下にある[上位ロード項目] テーブルを参照してください。SQL クエリやユーザーのような特定の項目を選択すると、その項目をドリルダウンして詳細を表示できます。

待機および上位 SQL クエリによってグループ分けされた DB 負荷は、Performance Insights ダッシュボードのデフォルトビューです。通常、この組み合わせは、パフォーマンス問題に関する最も正しい情報を提供します。待機でグループ化された DB 負荷は、データベースにリソースまたは同時のボトルネックがあるかどうかを示します。この場合、上位負荷項目のテーブルの SQL タブには、どのクエリがその負荷をかけているかが表示されます。

パフォーマンスの問題を診断するための一般的なワークフローは次のとおりです。

1. 「データベースロード」 グラフを確認し、最大 CPU ラインを超えているデータベースロードのインシデントがあるかどうかを確認します。
2. ある場合は、「データベースロード」 グラフを確認して、どの待機状態 (複数) が主に原因であるかを特定します。
3. 上位の負荷項目テーブルの SQL タブが待機状態に最も影響しているクエリを確認することによって、ロードを引き起こすダイジェストクエリを特定します。これらは [DB Load by Wait] 列で識別できます。
4. [SQL] タブでこれらのダイジェストクエリの 1 つを選択して展開し、構成されている子クエリを確認します。

例えば、以下のダッシュボードで、[ログファイルの同期] の待機はほとんどの DB 負荷の主な原因となっています。[LGWR すべてのワーカーグループ] の待機も高くなっています。「上位の SQL」 グラフでは、「ログファイルの同期」 の待機の発生元として、頻繁な`COMMIT` ステートメントが示されています。この場合、コミット頻度を下げると、DB 負荷が軽減されます。



一定期間のデータベースパフォーマンスの分析

一定期間のパフォーマンス分析レポートを作成して、オンデマンド分析でデータベースのパフォーマンスを分析します。パフォーマンス分析レポートで、リソースのボトルネックや DB インスタンスでのクエリの変更などのパフォーマンスの問題を確認します。Performance Insights ダッシュボードでは、期間を選択してパフォーマンス分析レポートを作成できます。レポートに 1 つ以上のタグを追加することもできます。

この機能を使用するには、有料利用枠の保持期間を使用している必要があります。詳細については、「Performance Insights の料金とデータ保持」を参照してください。

レポートは [パフォーマンス分析レポート - 新規] タブで選択して表示できます。レポートには、インサイト、関連するメトリクス、およびパフォーマンス問題を解決するための推奨事項が含まれています。レポートは、Performance Insights 保存期間中は表示できます。

レポート分析期間の開始時刻が保存期間外の場合、レポートは削除されます。保存期間が終了する前にレポートを削除することもできます。

パフォーマンス問題を検出し、DB インスタンスの分析レポートを生成するには、Performance Insights を有効にする必要があります。Performance Insights をオンにする方法の詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。

この機能のリージョン、DB エンジン、およびインスタンスクラスのサポート情報については、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

以下のセクションでは、パフォーマンス分析レポートの作成、表示、タグの追加、削除を行うことができます。

トピック

- Performance Insights でのパフォーマンス分析レポートの作成
- Performance Insights でのパフォーマンス分析レポートの表示
- Performance Insights のパフォーマンス分析レポートにタグを追加する
- Performance Insights のパフォーマンス分析レポートの削除

Performance Insights でのパフォーマンス分析レポートの作成

Performance Insights ダッシュボードで特定期間のパフォーマンス分析レポートを作成できます。期間を選択し、分析レポートに 1 つ以上のタグを追加できます。

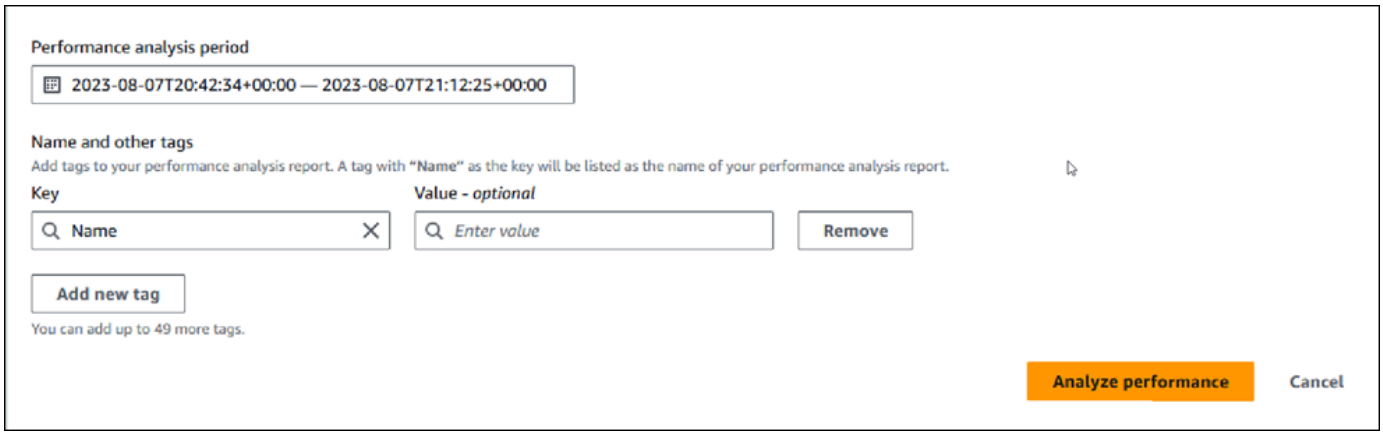
分析期間は 5 分から 6 日間までに設定できます。分析開始時刻の前に、少なくとも 24 時間のパフォーマンスデータが必要です。

この機能のリージョン、DB エンジン、およびインスタンスクラスのサポート情報については、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

一定期間のパフォーマンス分析レポートを作成するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。
4. Performance Insights ダッシュボードの [データベース負荷] セクションで [パフォーマンスを分析] を選択します。

期間を設定するフィールドと、パフォーマンス分析レポートに 1 つ以上のタグを追加するフィールドが表示されます。

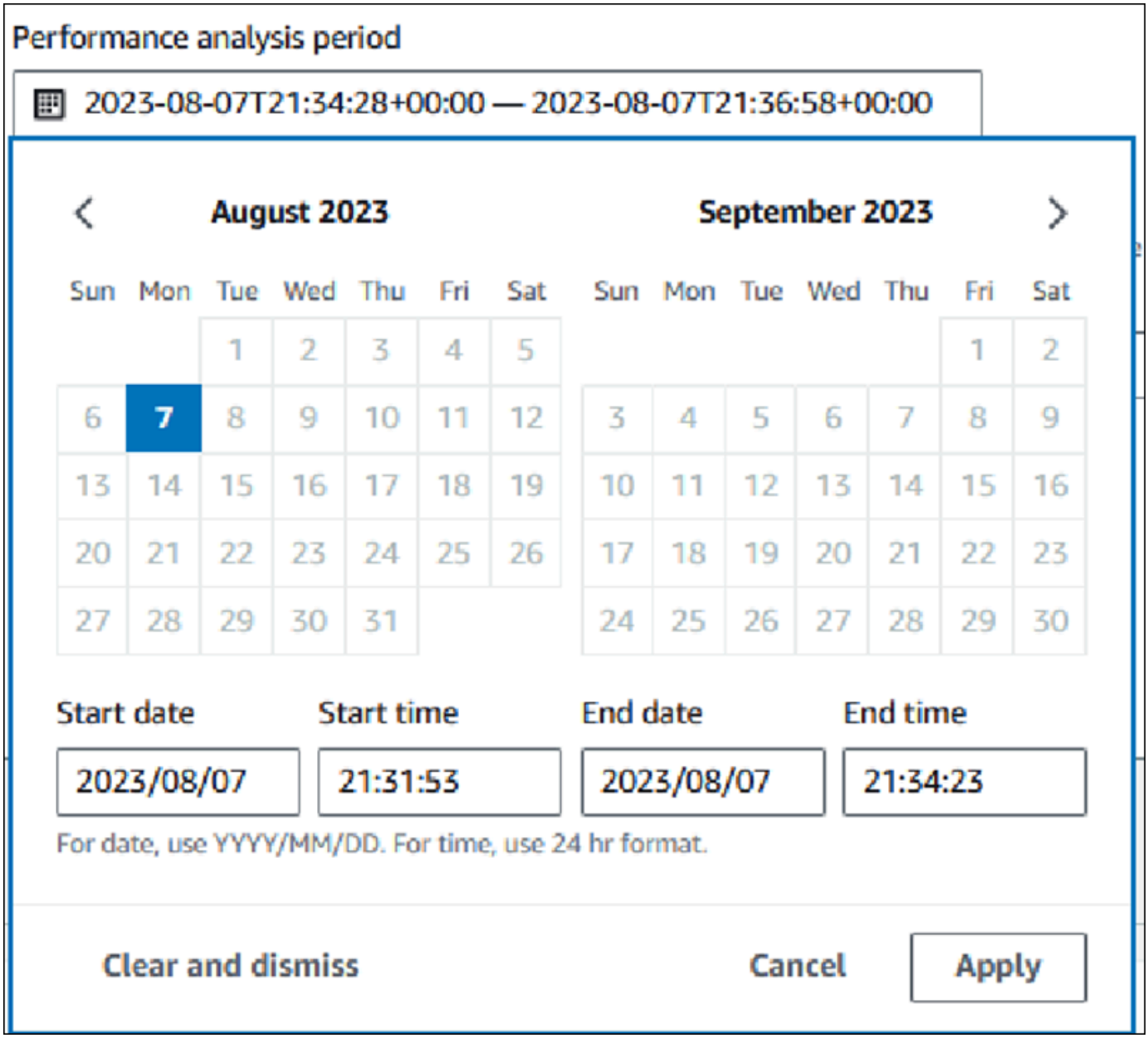


5. 期間を選択します。[相対的な範囲] または [絶対的な範囲] で期間を設定した場合は、この期間内の分析レポートの日付と時刻のみを入力または選択できます。この期間以外の分析期間を選択すると、エラーメッセージが表示されます。

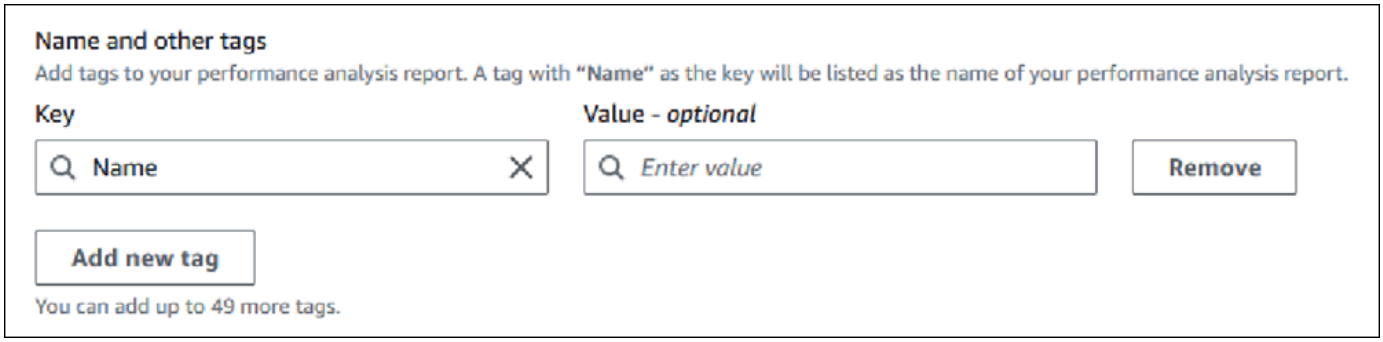
   期間を設定するには、次のいずれかを行います。

   - DB 負荷チャートのいずれかのスライダーを押してドラッグします。

     [パフォーマンス分析期間] ボックスに選択した期間が表示され、DB 負荷チャートに選択した期間が強調表示されます。
   - [パフォーマンス分析期間] ボックスで、[開始日]、[開始時間]、[終了日]、および [終了時間] を選択します。



6. (オプション) [キー] と [値 - オプション] を入力して、レポートにタグを追加します。



7. [パフォーマンスを分析] を選択します。

   バナーに、レポート生成が成功したか失敗したかを示すメッセージが表示されます。メッセージには、レポートを表示するためのリンクも記載されています。

   次の例は、レポート作成成功メッセージを含むバナーを示しています。

   レポートは [パフォーマンス分析レポート - 新規] タブで表示できます。

パフォーマンス分析レポートは、AWS CLI を使用して作成できます。AWS CLI を使用してレポートを作成する方法の例については、「一定期間のパフォーマンス分析レポートの作成」を参照してください。

Performance Insights でのパフォーマンス分析レポートの表示

[パフォーマンス分析レポート - 新規] タブには、DB インスタンスについて作成されたすべてのレポートが表示されます。各レポートには、以下が表示されます。

- [ID]: レポートの一意識別子。
- [名前]: レポートに追加されたタグキー。
- [レポート作成時間]: レポートを作成した時刻。
- [分析開始時間]: レポート内の分析の開始時刻。
- [分析終了時間]: レポート内の分析の終了時刻。

パフォーマンス分析レポートを表示するには

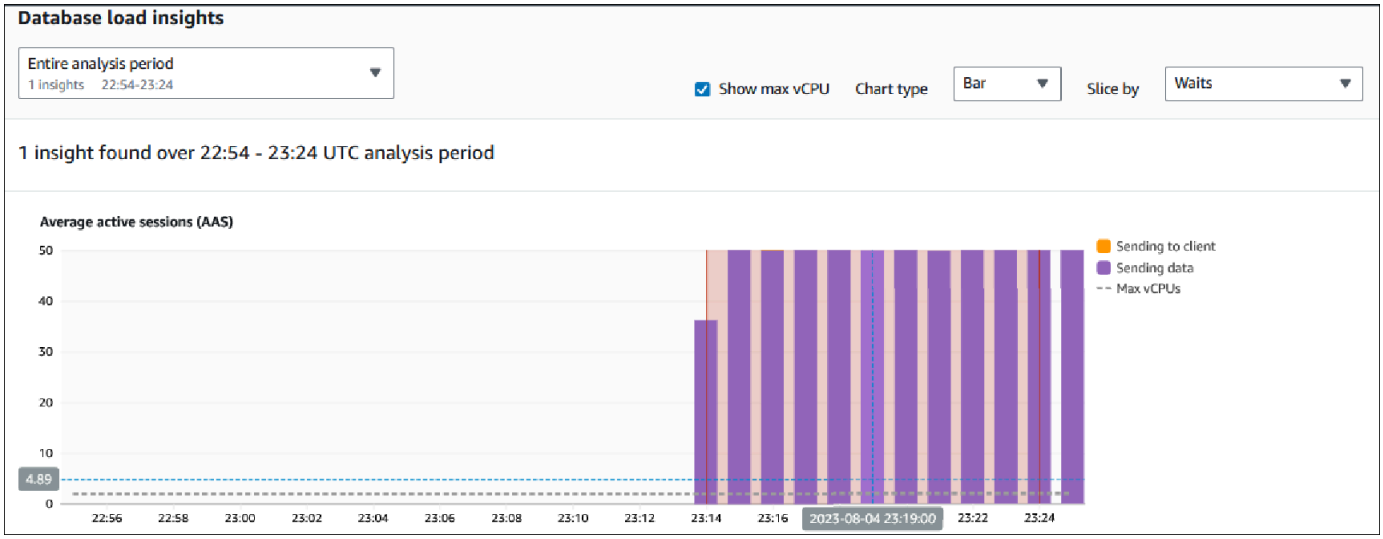
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. 分析レポートを表示する DB インスタンスを選択します。
4. 下にスクロールして、Performance Insights ダッシュボードの [パフォーマンス分析レポート - 新規] タブを選択します。

   異なる期間のすべての分析レポートが表示されます。
5. 表示するレポートの [ID] を選択します。

複数のインサイトが特定された場合、DB 負荷チャートにはデフォルトで分析期間全体が表示されます。レポートで 1 つのインサイトが特定された場合、DB 負荷チャートにはデフォルトでそのインサイトが表示されます。

ダッシュボードには、[タグ] セクションにレポートのタグも一覧表示されます。

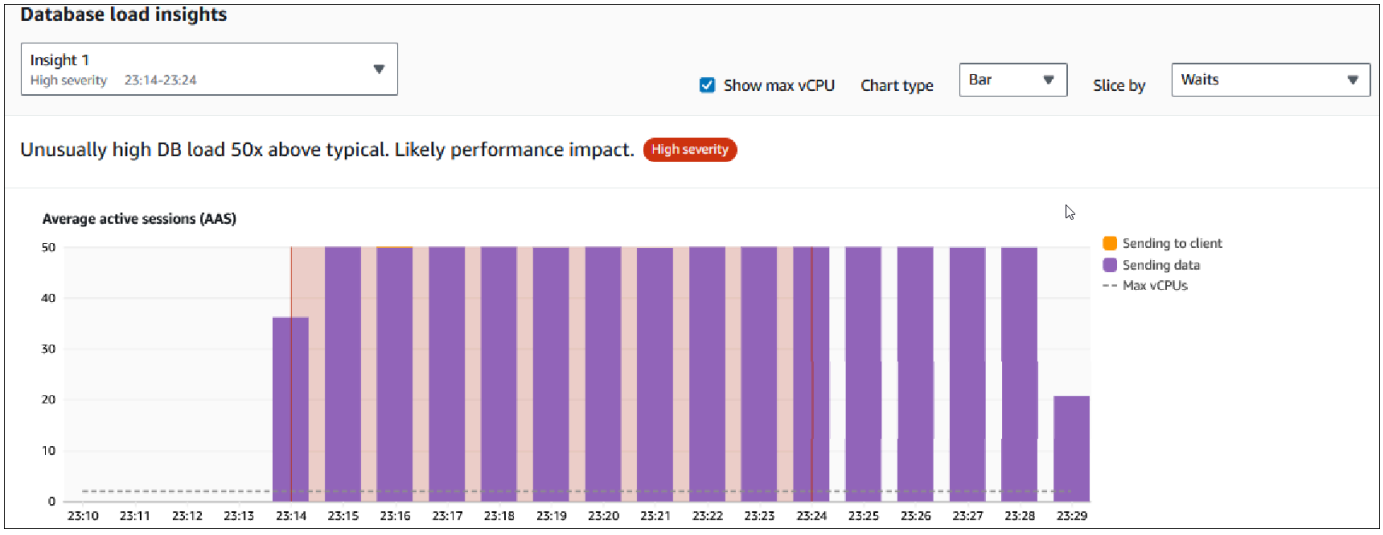
次の例は、レポートの分析期間全体を示しています。

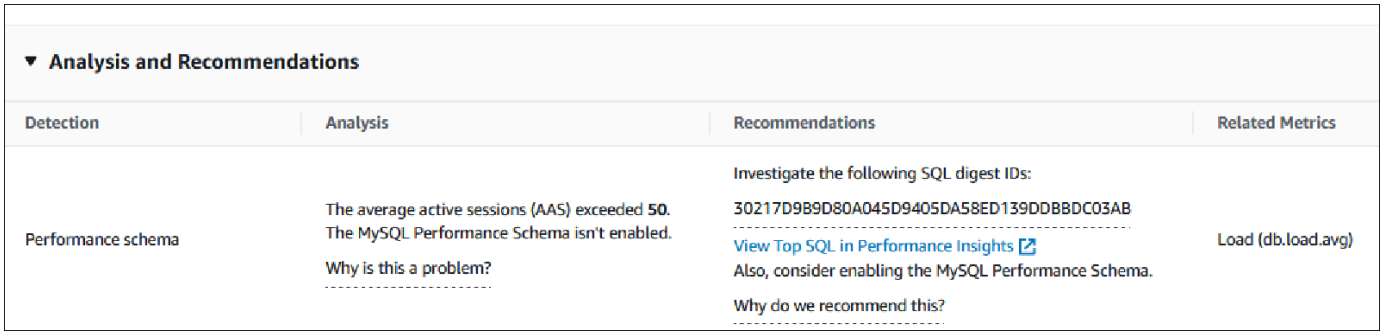


6. レポートで複数のインサイトが特定された場合、[データベース負荷に関するインサイト] リストで、表示するインサイトを選択します。

ダッシュボードには、インサイトメッセージ、インサイトの期間が強調表示された DB 負荷チャート、分析と推奨事項、およびレポートタグのリストが表示されます。

次の例は、レポートの DB 負荷インサイトを示しています。





Performance Insights のパフォーマンス分析レポートにタグを追加する

レポートを作成または表示するときに、タグを追加できます。レポートには最大 50 個のタグを追加できます。

タグを追加するアクセス許可が必要です。Performance Insights のアクセスポリシーの詳細については、「Performance Insights 用のアクセスポリシーの設定」を参照してください。

レポートの作成時に 1 つ以上のタグを追加するには、手順 Performance Insights でのパフォーマンス分析レポートの作成 のステップ 6 を参照してください。

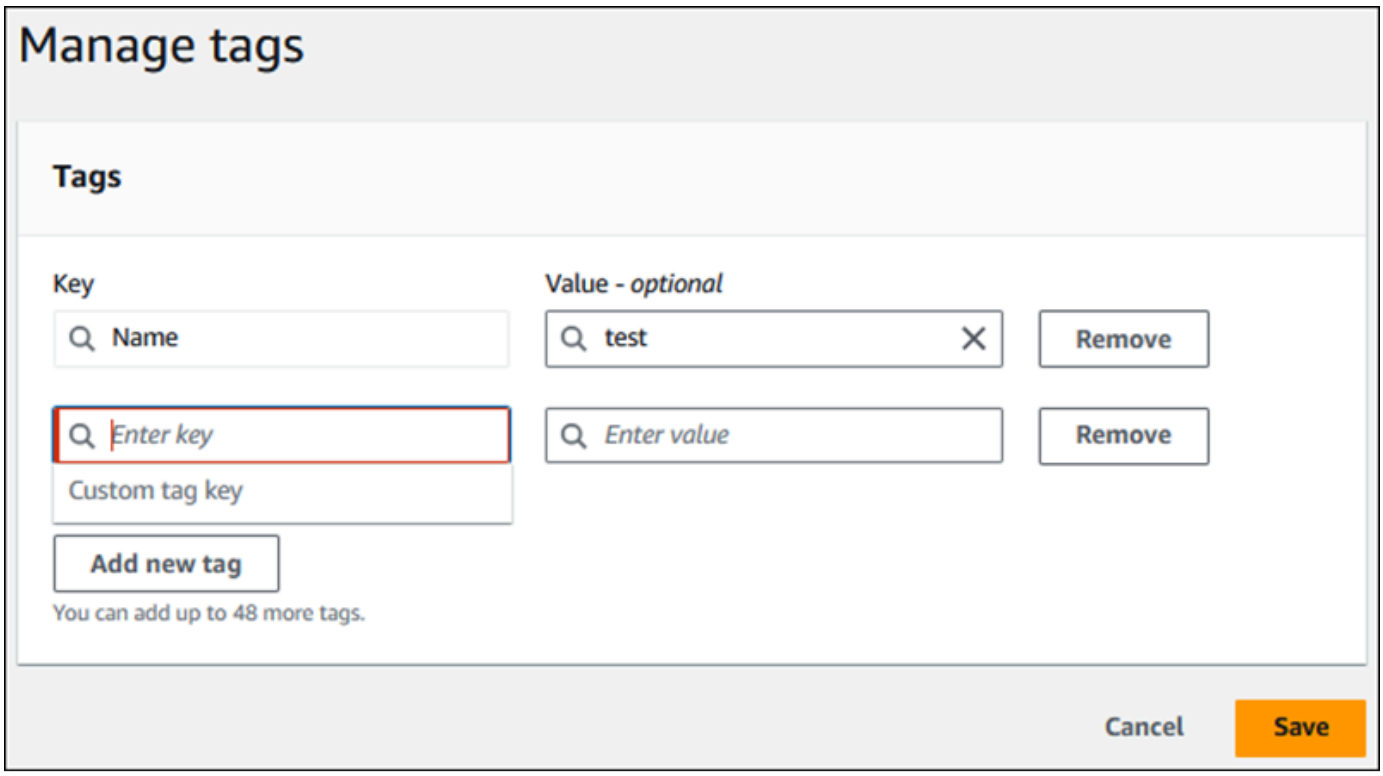
レポートを表示するときに 1 つ以上のタグを追加するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. 下にスクロールして、パフォーマンス分析レポート - 新規] タブを選択します。
5. タグを追加するレポートを選択します。

   ダッシュボードにレポートが表示されます。
6. [タグ] までスクロールして、[タグを管理] を選択します。
7. [新しいタグを追加] をクリックします。
8. [キー] と [値 - オプション] を入力し、[新しいタグを追加] を選択します。

   次の例では、選択したレポートに新しいタグを追加するオプションを提供しています。



レポート用に新しいタグが作成されます。

ダッシュボードの [タグ] セクションにレポートのタグのリストが表示されます。レポートからタグを削除する場合は、タグの横の [削除] を選択します。

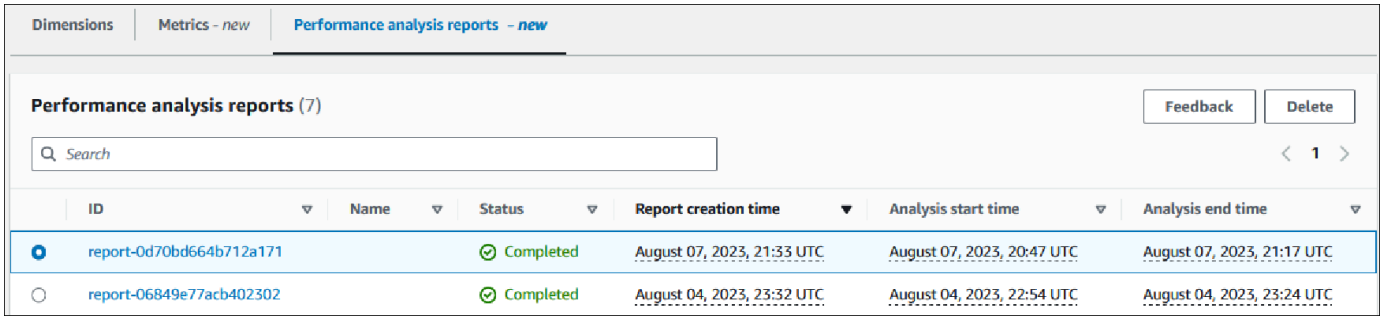
Performance Insights のパフォーマンス分析レポートの削除

[パフォーマンス分析レポート] タブに表示されているレポートのリストから、またはレポートの表示中にレポートを削除できます。

レポートを削除するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。

   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. 下にスクロールして、パフォーマンス分析レポート - 新規] タブを選択します。
5. 削除するレポートを選択して、[削除] を選択します。



確認ウィンドウが表示されます。確認を選択すると、レポートは削除されます。

6. (オプション) 削除するレポートの ID を選択します。

   レポートページで、[削除] を選択します。

   確認ウィンドウが表示されます。確認を選択すると、レポートは削除されます。

Performance Insights のトップ SQL タブを使用したクエリの分析

Amazon RDS Performance Insights ダッシュボードでは、実行中のクエリや最近のクエリに関する情報を [Top dimensions] (上位ディメンション) テーブルの[Top SQL] (上位の SQL) タブで見ることができます。この情報を使用して、クエリをチューニングできます。

トピック

- [トップ SQL] タブの概要
- Performance Insights ダッシュボードでより多くの SQL テキストにアクセスする
- Performance Insights ダッシュボードでの SQL 統計の表示

[トップ SQL] タブの概要

デフォルトでは、[Top SQL] (上位の SQL) タブはデータベースロードに最も貢献している 25 クエリを表示します。クエリをチューニングするために、クエリテキスト、SQL 統計などの情報を分析できます。また、[トップ SQL] タブに表示する統計を選択することもできます。

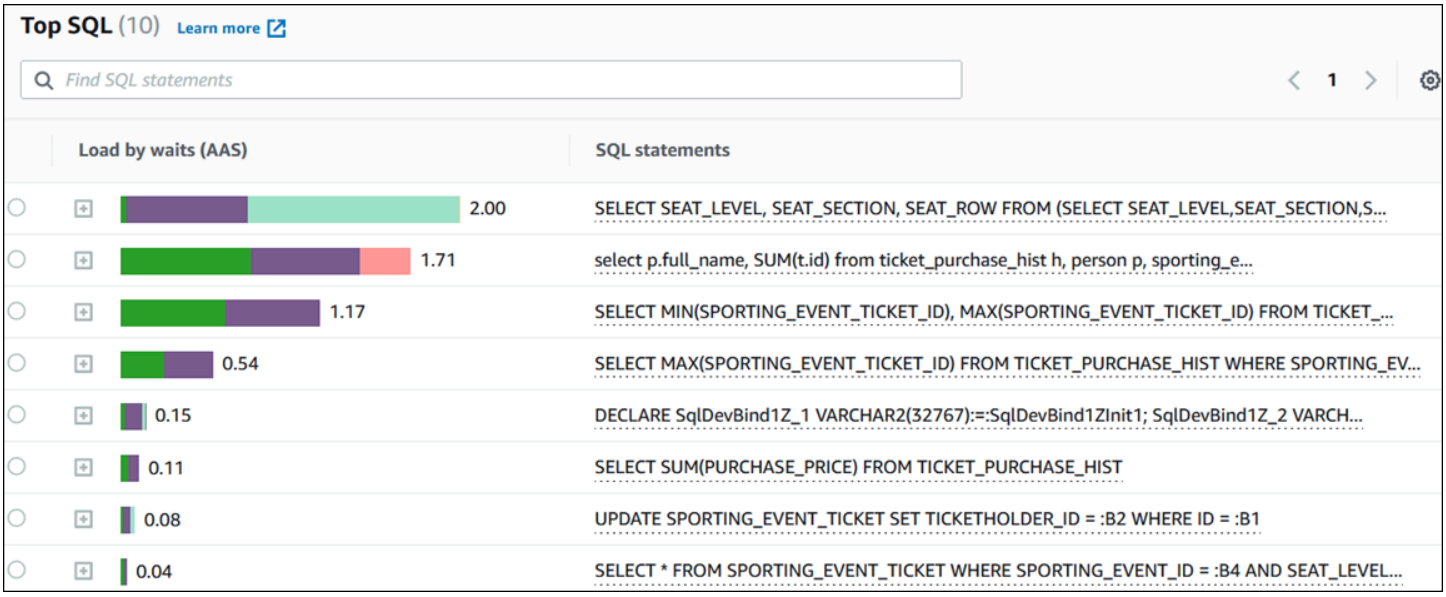
トピック

- SQL テキスト
- SQL 統計
- 待機によるロード (AAS)

- SQL 情報の表示
- 統計設定を選択する

SQL テキスト

デフォルトでは、[Top SQL] (上位の SQL) テーブルの各行にはステートメントごとに 500 バイトのテキストが表示されます。

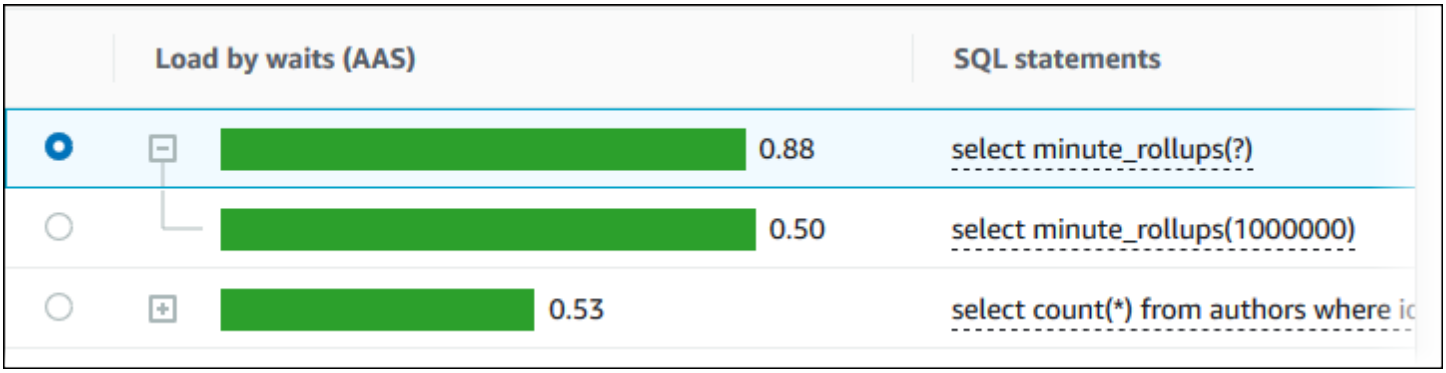


デフォルトの 500 バイト以上の SQL テキストを表示する方法については、「Performance Insights ダッシュボードでより多くの SQL テキストにアクセスする」を参照してください。

SQL ダイジェストは、構造的には類似しているが、異なるリテラル値を含む可能性の高い、複数の実際のクエリの複合体です。ダイジェストは、ハードコードされた値を疑問符に置き換えます。例えば、ダイジェストはSELECT * FROM emp WHERE lname= ?のことがあります。このダイジェストには、次の子クエリが含まれます。

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

ダイジェスト内でリテラル SQL ステートメントを表示するには、クエリを選択してからプラス記号 (+) を選択します。以下の例では、選択されたクエリはダイジェストです。



> **Note**
>
> SQL ダイジェストでは、類似した SQL ステートメントがグループ化されますが、機密情報は編集されません。

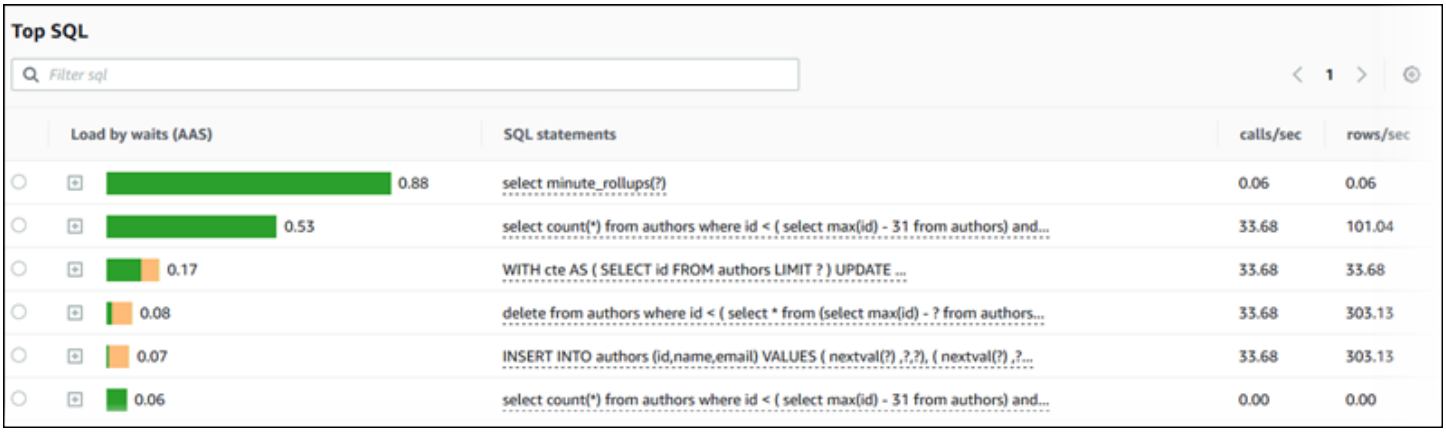
Performance Insights では、Oracle SQL テキストが [Unknown] (不明) と表示される可能性があります。次のような状況では、テキストはこの状態になります。

- SYS 以外の Oracle データベースユーザーはアクティブですが、現在 SQL を実行していません。例えば、並列クエリが完了すると、クエリコーディネータはヘルパープロセスがセッション統計を送信するのを待ちます。待機中は、クエリテキストに [Unknown] (不明) と表示されます。
- Standard Edition 2 の RDS for Oracle インスタンスの場合、リソースマネージャは並列スレッドの数を制限します。この作業を行うバックグラウンドプロセスにより、クエリテキストに [不明] と表示されます。

SQL 統計

SQL 統計は、SQL クエリに関するパフォーマンス関連のメトリックです。例えば、Performance Insights には 1 秒あたりの実行数や 1 秒あたりの処理行数が表示されることがあります。Performance Insights は、最も一般的なクエリのみの統計を収集します。通常、これらは Performance Insights ダッシュボードに負荷別に表示される上位のクエリと一致します。

[トップ SQL] テーブル内の各行は、次の例のように、SQL ステートメントまたはダイジェストに関連する統計を示します。



Performance Insights は、SQL 統計で `0.00` および - (不明) をレポートする可能性があります。この状況は、以下の条件で発生します。

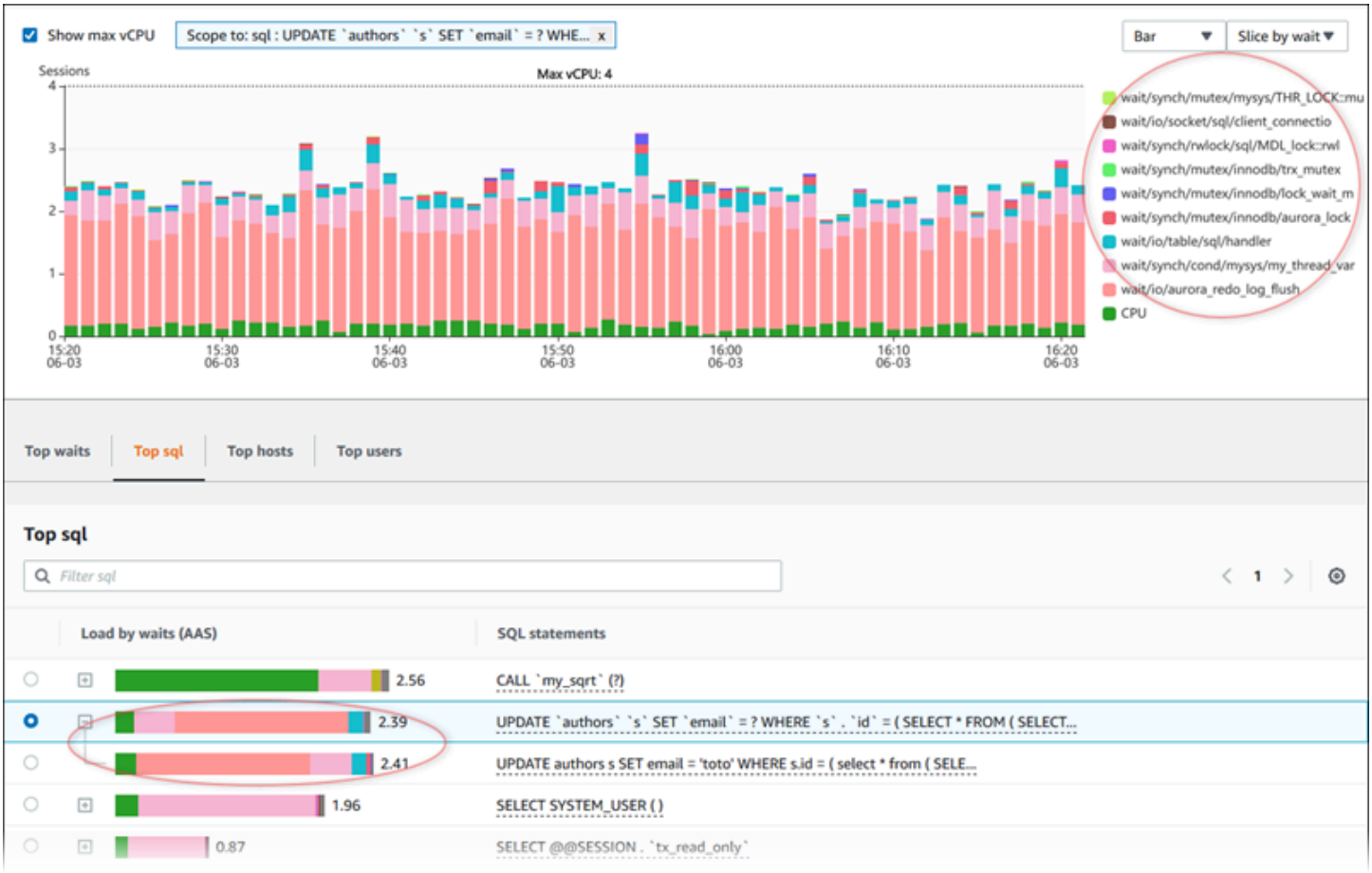
- サンプルが 1 つだけ存在する。例えば、Performance Insights は、`pg_stat_statements` ビューからの複数のサンプルに基づいて、RDS PostgreSQL クエリの変更率を計算します。ワークロードが短時間実行されると、Performance Insights ではサンプルが 1 つしか収集されない場合があります。つまり、変更率を計算できません。不明な値はダッシュ (-) で表されます。
- 2 つのサンプルが同じ値を持っている。Performance Insights は、変更が発生していないため、変更率を計算することができず、変化率を `0.00` と報告します。
- RDS PostgreSQL ステートメントに有効な識別子がない。PostgreSQL は、構文解析と分析の後にのみ、ステートメントの識別子を作成します。したがって、PostgreSQL の内部インメモリ構造に、識別子なしでステートメントが存在する可能性があります。Performance Insights は内部メモリ内構造を 1 秒に 1 回サンプリングするため、低レイテンシーのクエリは 1 つのサンプルに対してのみ表示されることがあります。このサンプルでクエリ識別子を使用できない場合、Performance Insights はこのステートメントを統計に関連付けることはできません。不明な値はダッシュ (-) で表されます。

Amazon RDS エンジンの SQL 統計の説明については、「Performance Insights の SQL 統計」を参照してください。

待機によるロード (AAS)

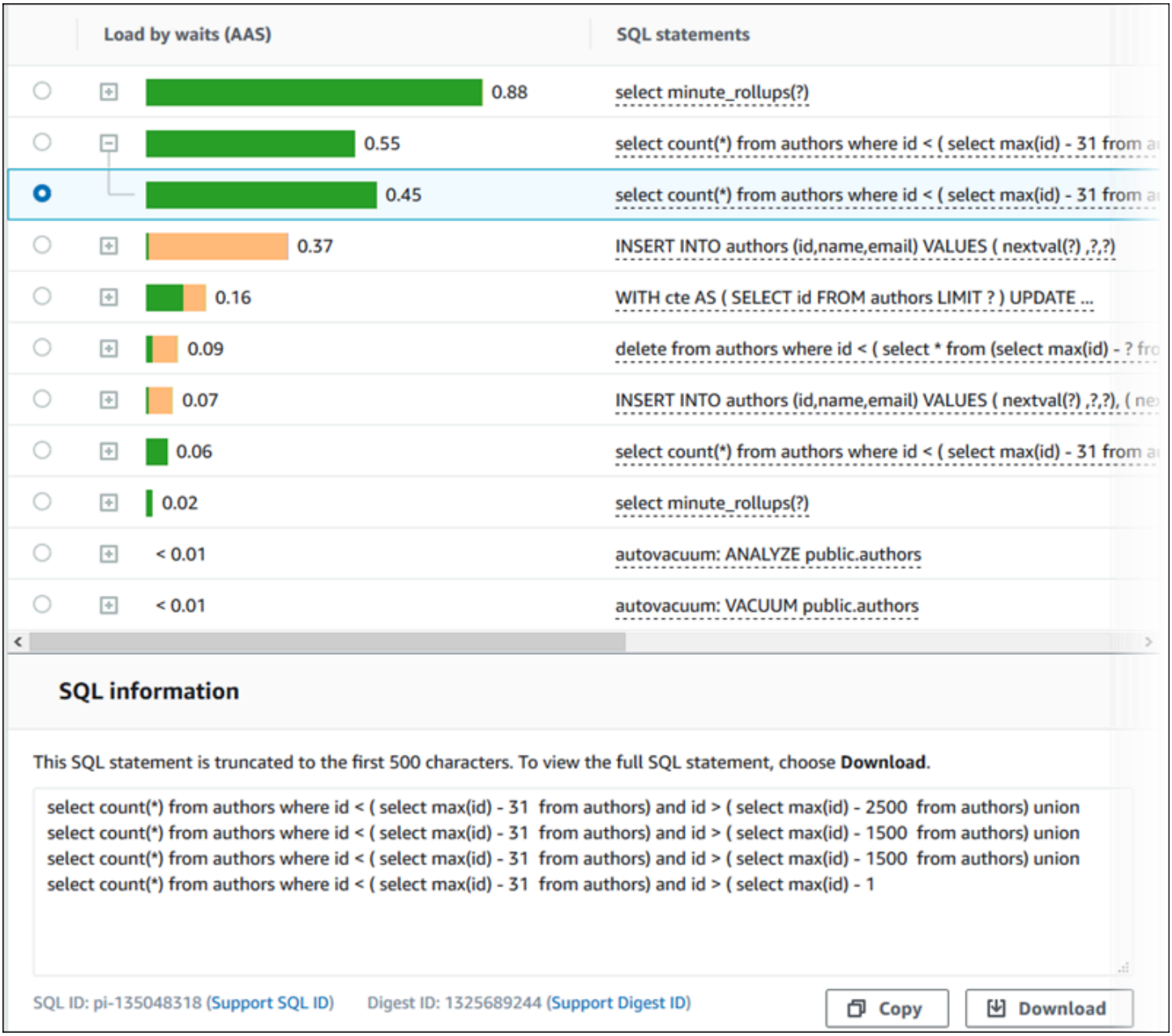
[トップ SQL] の [待機別の負荷 (AAS)] 列は、上位の各ロード項目に関連付けられているデータベースロードの割合を示しています。この列には、DB 負荷グラフで現在選択されているグループ化に応じて、その項目に対する負荷が反映されます。平均アクティブセッション (AAS) の詳細については、平均アクティブセッション を参照してください。

例えば、DB 負荷グラフを待機状態別にグループ化できます。上位負荷項目のテーブルで SQL クエリを調べます。この場合は、[待機別の DB 負荷] バーは、クエリが貢献している待機状態の量を示すために、サイズ、セグメント、および色で分けられています。また、選択したクエリに影響を与えている待機状態も示されます。



SQL 情報の表示

[トップ SQL] テーブルで、ステートメントを開いてその情報を表示できます。詳細ペインに情報が表示されます。

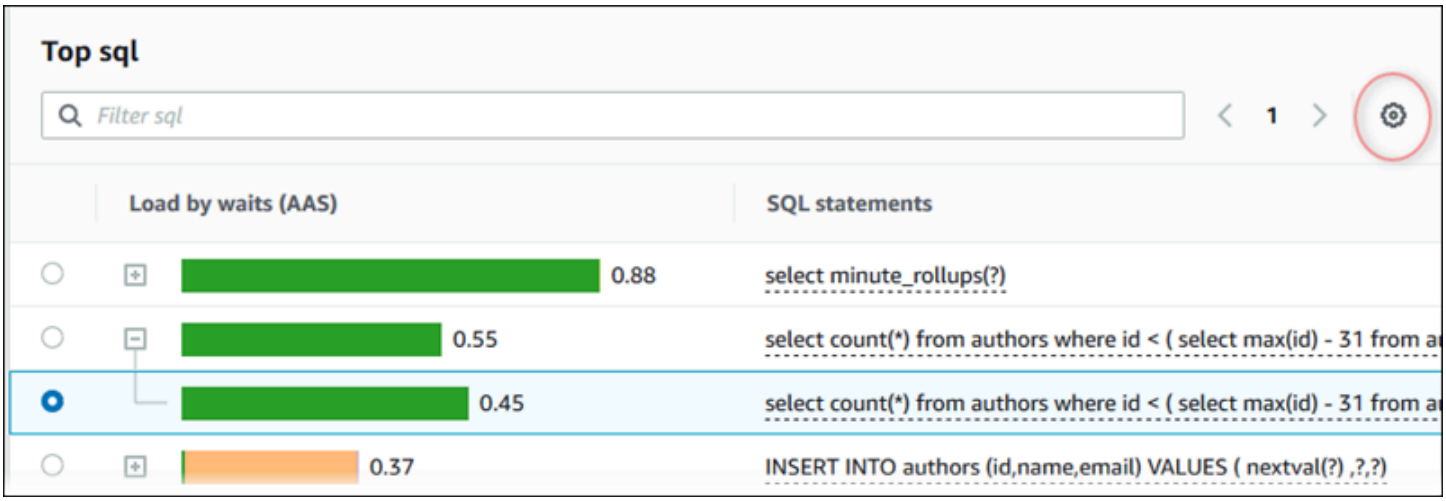


SQL ステートメントに関連付けられているタイプの識別子 (ID) は以下のとおりです。

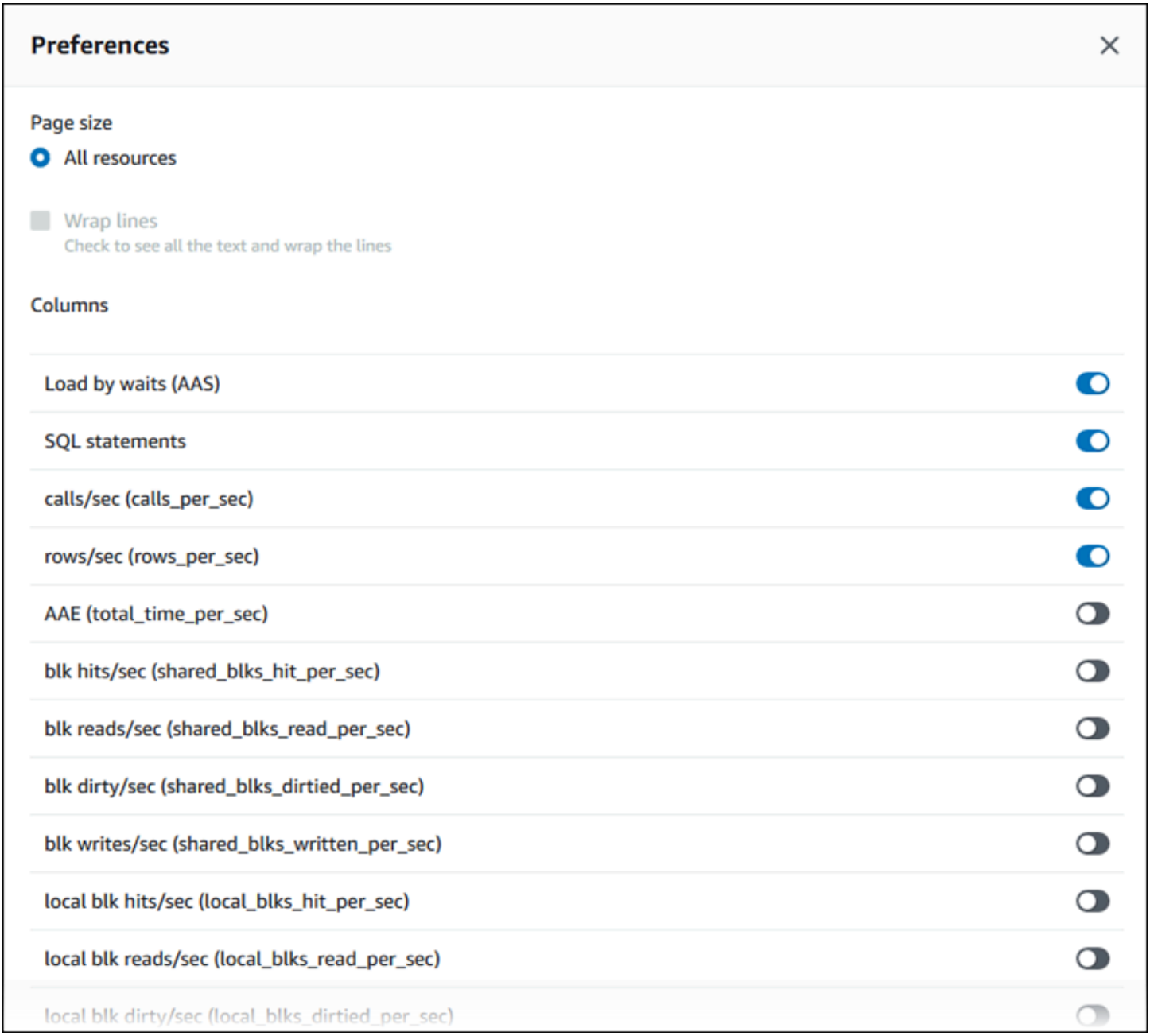
- Support SQL ID - SQL ID のハッシュ値。この値は、AWS サポートを利用しているときに SQL ID を参照するためだけのものです。AWS サポートが実際の SQL ID や SQL テキストにアクセスすることはできません。
- ダイジェスト ID のサポート - Digest ID のハッシュ値。この値は、AWS サポートを利用しているときにダイジェスト ID を参照するためだけのものです。AWSサポートが実際のダイジェスト ID や SQL テキストにアクセスすることはできません。

統計設定を選択する

「設定」 アイコンを選択すると、[トップ SQL] タブに表示される統計を制御できます。



[設定] アイコンを選択すると、[設定] ウィンドウが開きます。次のスクリーンショットは、[Preferences] (環境設定) ウィンドウの例です。



[トップ SQL] タブに表示させたい統計を有効にするには、マウスを使用してウィンドウの下部までスクロールし、[続行] を選択します。

Amazon RDS エンジンの秒単位または呼び出し単位の統計の詳細については、Performance Insights の SQL 統計 の「エンジン固有の SQL 統計」セクションを参照してください。

Performance Insights ダッシュボードでより多くの SQL テキストにアクセスする

デフォルトでは、[トップ SQL] テーブルの各行には SQL ステートメントごとに 500 バイトの SQL テキストが表示されます。

SQL ステートメントのサイズが 500 バイトを超える場合、[Top SQL] (トップ SQL) テーブルの [SQL text] (SQL テキスト) セクションでテキストの表示量を増やすことができます。この場合、[SQL text] (SQL テキスト) に表示されるテキストの最大長は 4 KB です。コンソールで導入されるこの制限には、データベースエンジンによって設定された制限が適用されます。[SQL text] (SQL テキスト) に表示されているテキストを保存するには、[Download] (ダウンロード) を選択します。

トピック

- Amazon RDS エンジン のテキストサイズの制限
- Amazon RDS for PostgreSQL DB インスタンスの SQL テキスト制限の設定
- Performance Insights ダッシュボードでの SQL テキストの表示とダウンロード

Amazon RDS エンジン のテキストサイズの制限

SQL テキストをダウンロードするときに、データベースエンジンがテキストの最大長を決定します。エンジンごとのダウンロードできる SQL テキストの上限は次のとおりです。

DB エンジン	ダウンロードされるテキストの最大長
Amazon RDS for MySQL および MariaDB	Performance Schema が有効になっている場合、長さは 4,096 バイトに固定されます。Performance Schema が有効になっていない場合、長さは 65,535 バイトに固定されます。
Amazon RDS for Microsoft SQL Server	4,096 文字
Amazon RDS for Oracle	1,000 バイト

Performance Insights コンソールの [SQL text] (SQL テキスト) では、エンジンが返すテキストが最大値まで表示できます。例えば、 MySQL は、Performance Insights に対して最大 1 KB を返します。元のクエリが大きい場合でも、収集して表示できるのは 1 KB のみです。したがって、[SQL text] (SQL テキスト) でクエリを表示するか、ダウンロードすると、Performance Insights は同じバイト数を返します。

AWS CLI または API を使用する場合、Performance Insights には、コンソールで適用される 4 KB の制限がありません。`DescribeDimensionKeys` と `GetResourceMetrics` は、最大で 500 バイトを返します。

> **Note**
>
> `GetDimensionKeyDetails` はクエリ全体を返しますが、サイズにはエンジンの制限が適用されます。

Amazon RDS for PostgreSQL DB インスタンスの SQL テキスト制限の設定

Amazon RDS for PostgreSQL は、テキストを異なる方法で処理します。DB インスタンスパラメータ `track_activity_query_size` を使用して、テキストサイズの制限を設定できます。このパラメータには次の特徴があります。

デフォルトのテキストサイズ

Amazon RDS for PostgreSQL バージョン 9.6 では、`track_activity_query_size` パラメータのデフォルト設定は 1,024 バイトです。Amazon RDS for PostgreSQL バージョン 10 以降では、デフォルトは 4,096 バイトです。

最大テキストサイズ

Amazon RDS for PostgreSQL バージョン 12 以前の場合、`track_activity_query_size` の制限は 102,400 バイトです。バージョン 13 以降の場合、最大値は 1 MB です。

エンジンが Performance Insights に対して 1 MB を返す場合、コンソールでは最初の 4 KB のみが表示されます。クエリをダウンロードする場合、1 MB すべてを取得できます。この場合、表示する場合とダウンロードする場合では異なるバイト数が返されます。`track_activity_query_size` DB インスタンスパラメータの詳細については、PostgreSQL ドキュメントで「ランタイム統計」を参照してください。

SQL テキストのサイズを大きくするには、`track_activity_query_size` の制限を引き上げます。パラメータを変更するには、Amazon RDS for PostgreSQL DB インスタンスに関連付けられているパラメータグループのパラメータ設定を変更します。

インスタンスでデフォルトのパラメータグループが使用される際に設定を変更するには

1. 該当する DB エンジンおよび DB エンジンバージョンの新しい DB インスタンスパラメータグループを作成します。
2. 新しいパラメータグループにパラメータを設定します。
3. 新しいパラメータグループを DB インスタンスに関連付けます。

DB インスタンスパラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

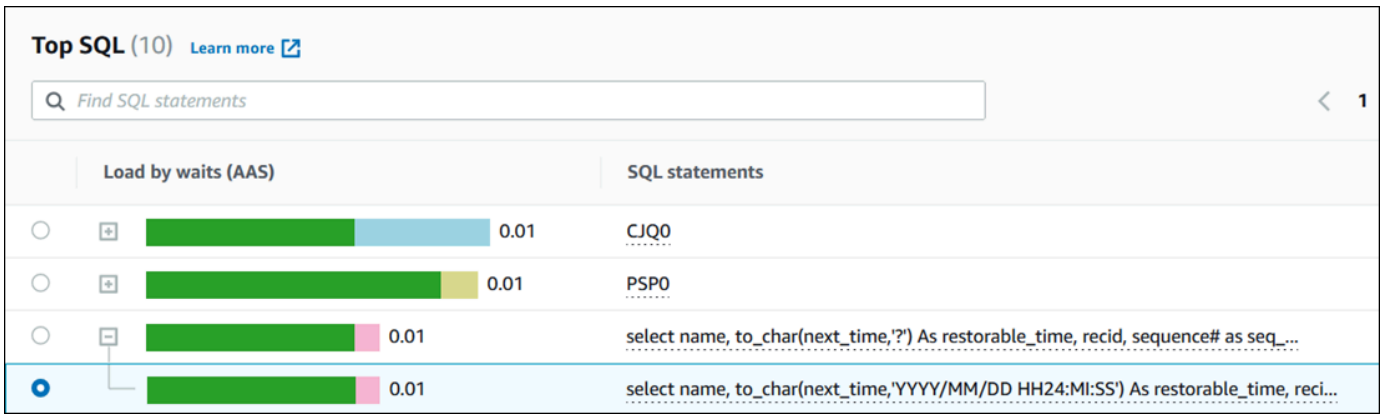
Performance Insights ダッシュボードでの SQL テキストの表示とダウンロード

Performance Insights ダッシュボードで、SQL テキストを表示およびダウンロードできます。

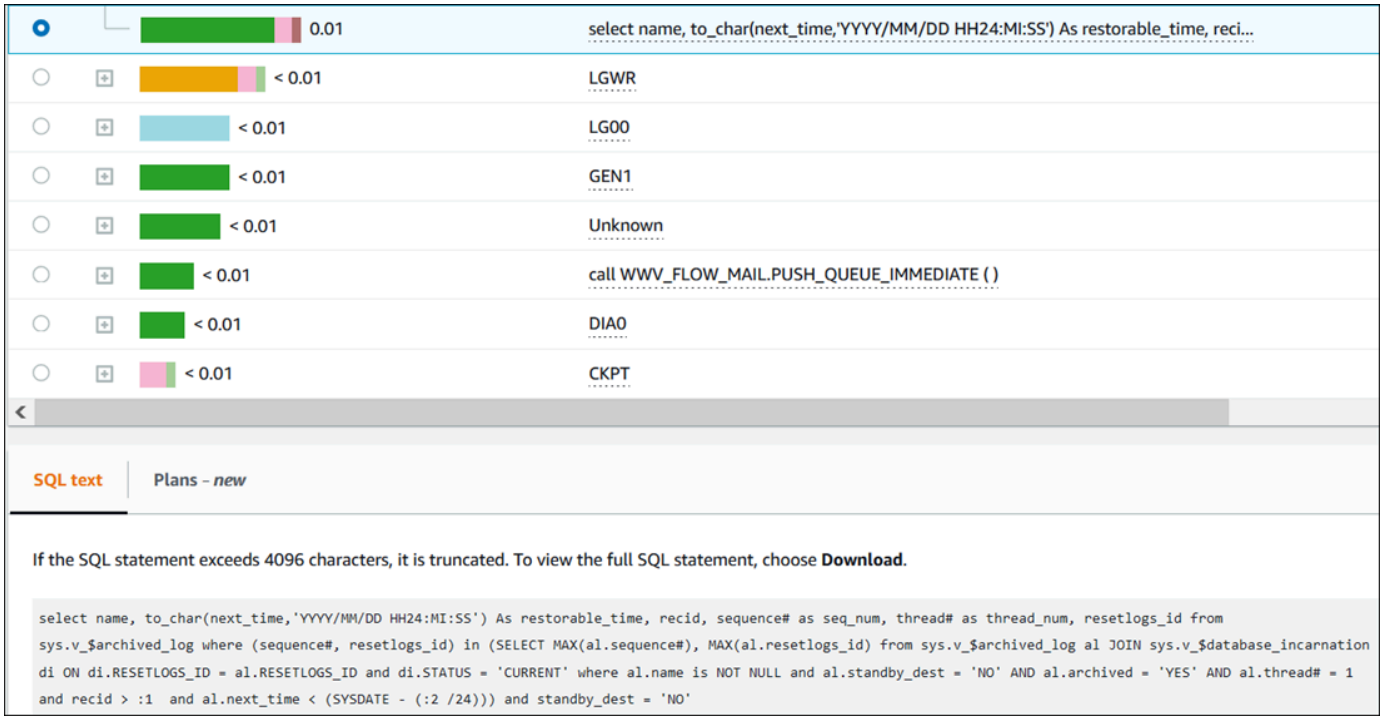
Performance Insights ダッシュボードで SQL テキストの表示量を増やすには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. DB インスタンスを選択します。
4. Performance Insights ダッシュボードの [トップ SQL] タブまでスクロールします。
5. プラス記号を選択して SQL ダイジェストを展開し、ダイジェストの子クエリのいずれかを選択します。

   500 バイトを超える SQL ステートメントは、次のイメージのように表示されます。



6. 下部にある [SQL テキスト] タブまでスクロールします。



Performance Insights ダッシュボードは、各 SQL ステートメントの最大 4,096 バイトまでを表示できます。

7. (オプション) [コピー] を選択して、表示された SQL ステートメントをコピーするか、[ダウンロード] を選択して、DB エンジンに応じた最大サイズの SQL ステートメントをダウンロードします。

> **Note**
>
> SQL ステートメントをコピーまたはダウンロードするには、ポップアップブロッカーを無効にします。

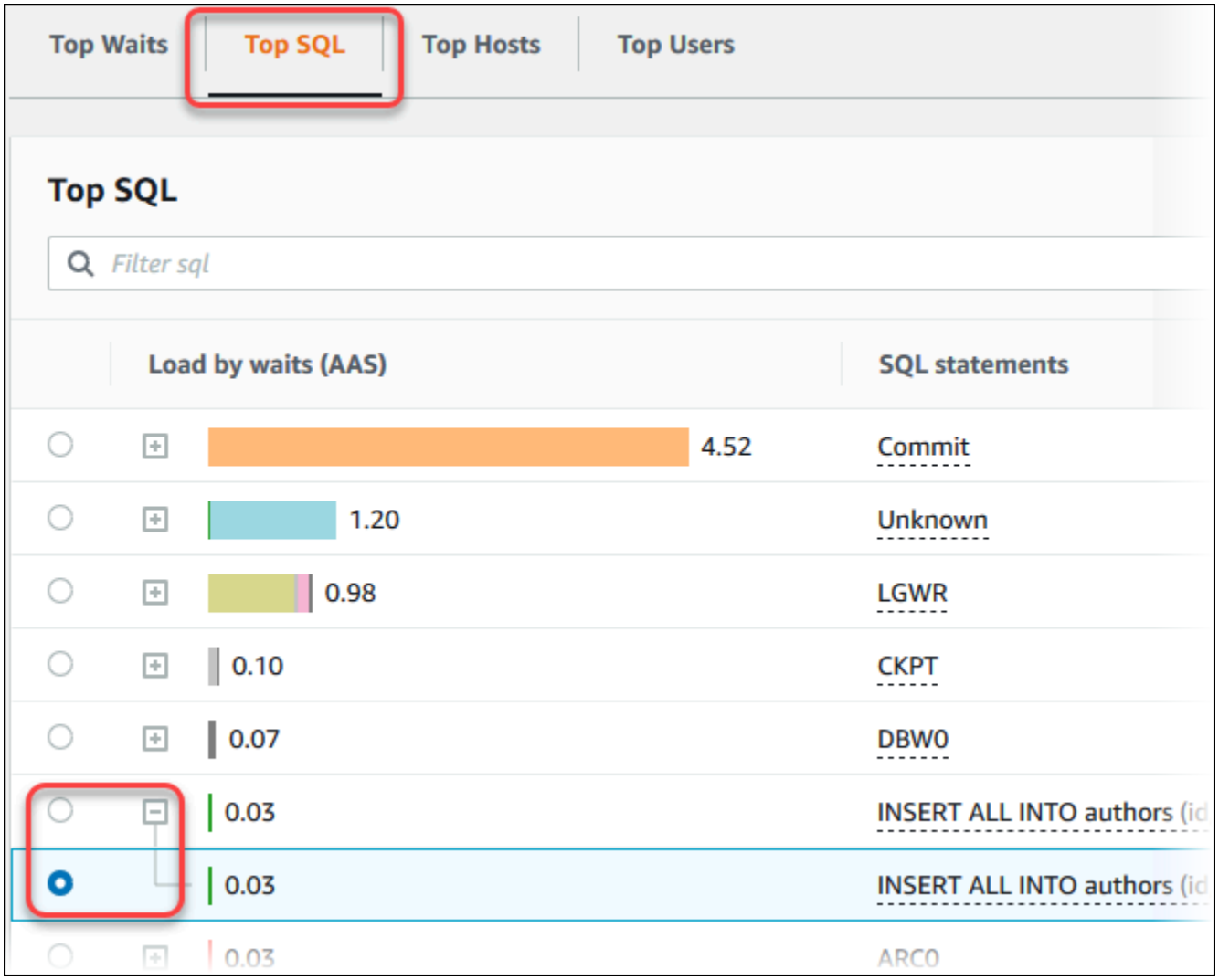
Performance Insights ダッシュボードでの SQL 統計の表示

Performance Insights ダッシュボードでは、SQL 統計を [データベース負荷] グラフの [トップ SQL] タブで見ることができます。

SQL 統計を表示するには

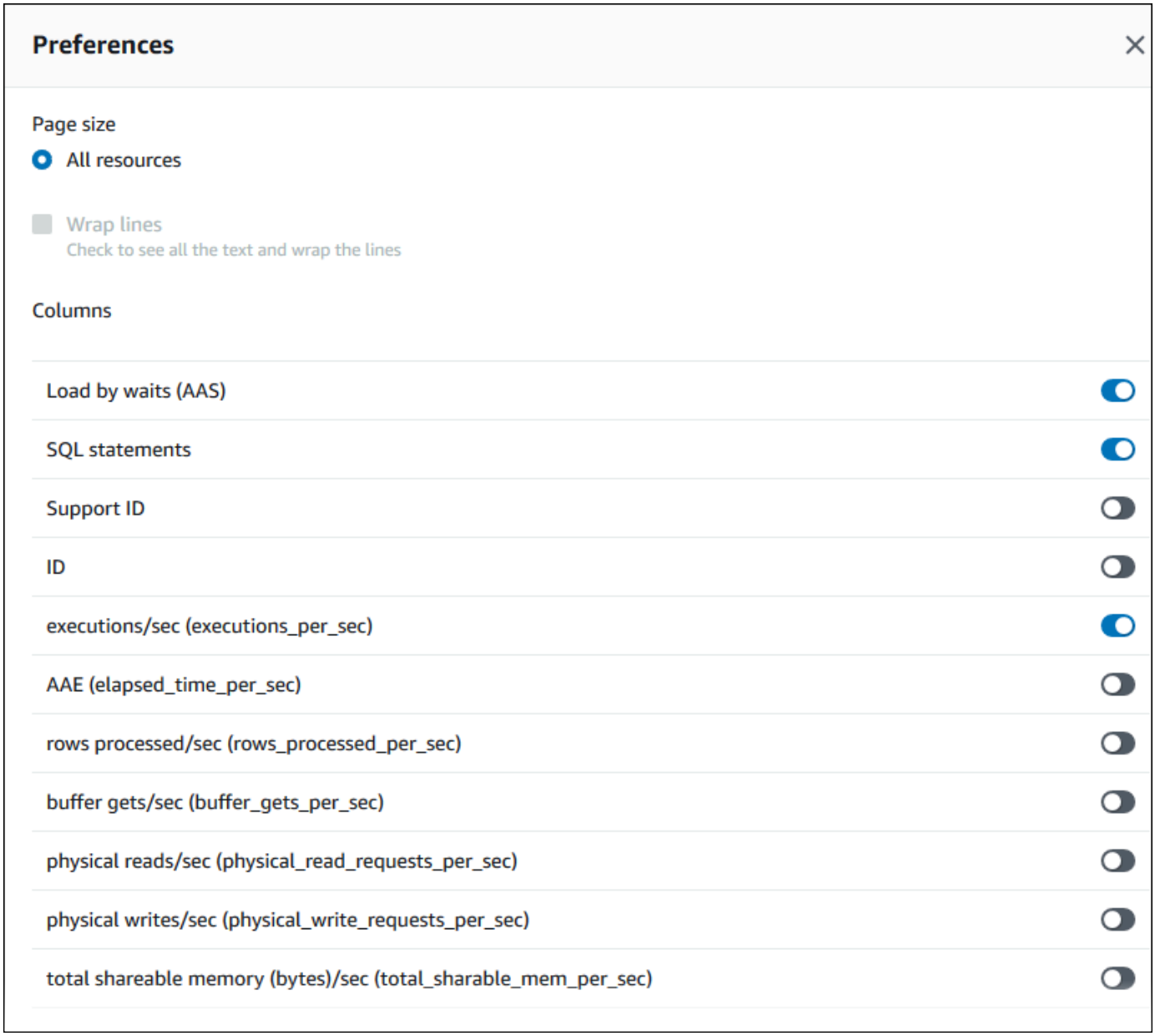
1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。

3. SQL 統計を表示するデータベースを選択します。
4. [上位の SQL] タブを選択します。
5. 個々のステートメント またはダイジェストクエリを選択します。

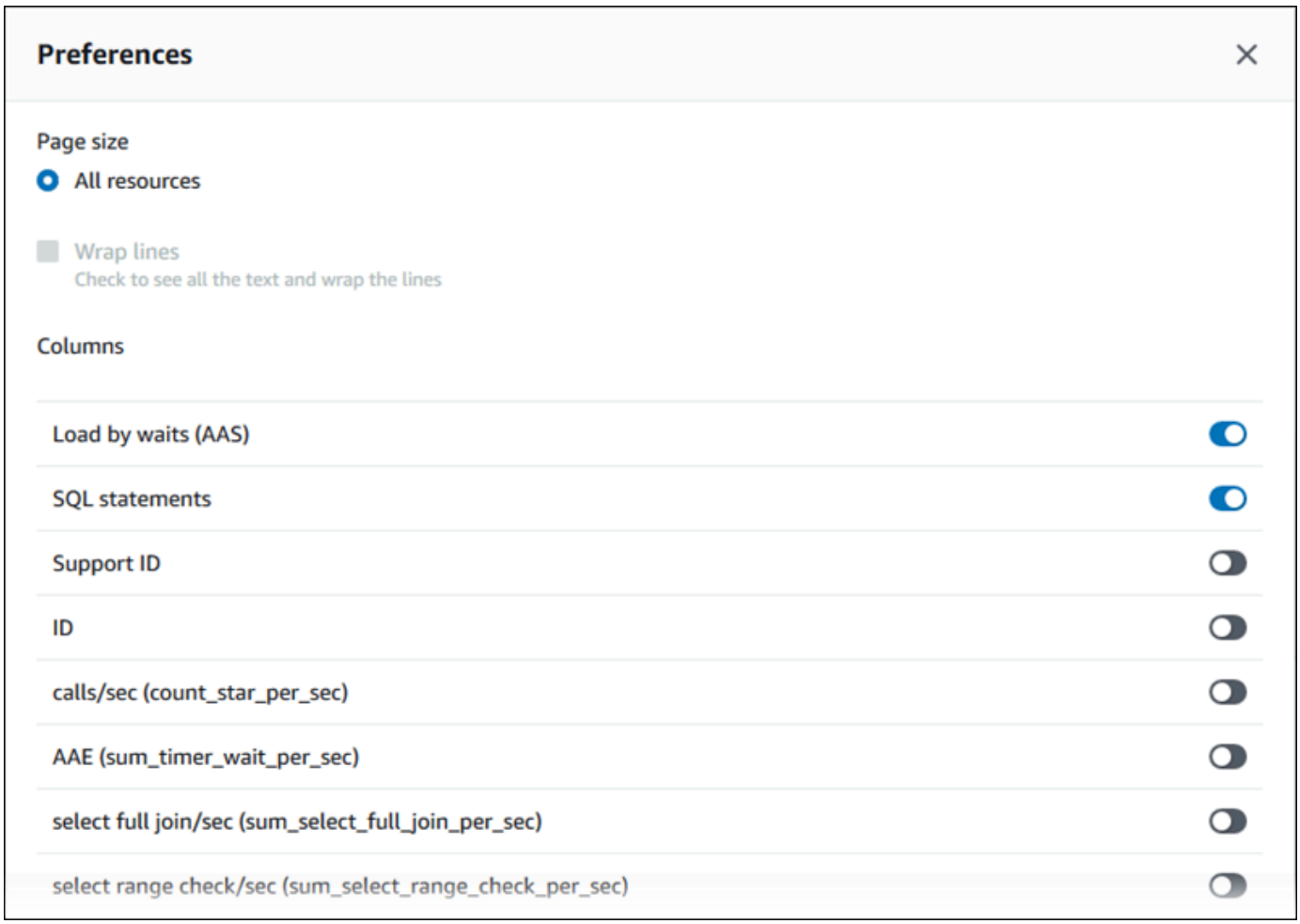


6. 歯車アイコンを選択して、表示する統計を選択します。Amazon RDS エンジンの SQL 統計の説明については、「Performance Insights の SQL 統計」を参照してください。

   次の例は、Oracle DB インスタンスの統計設定を示しています。



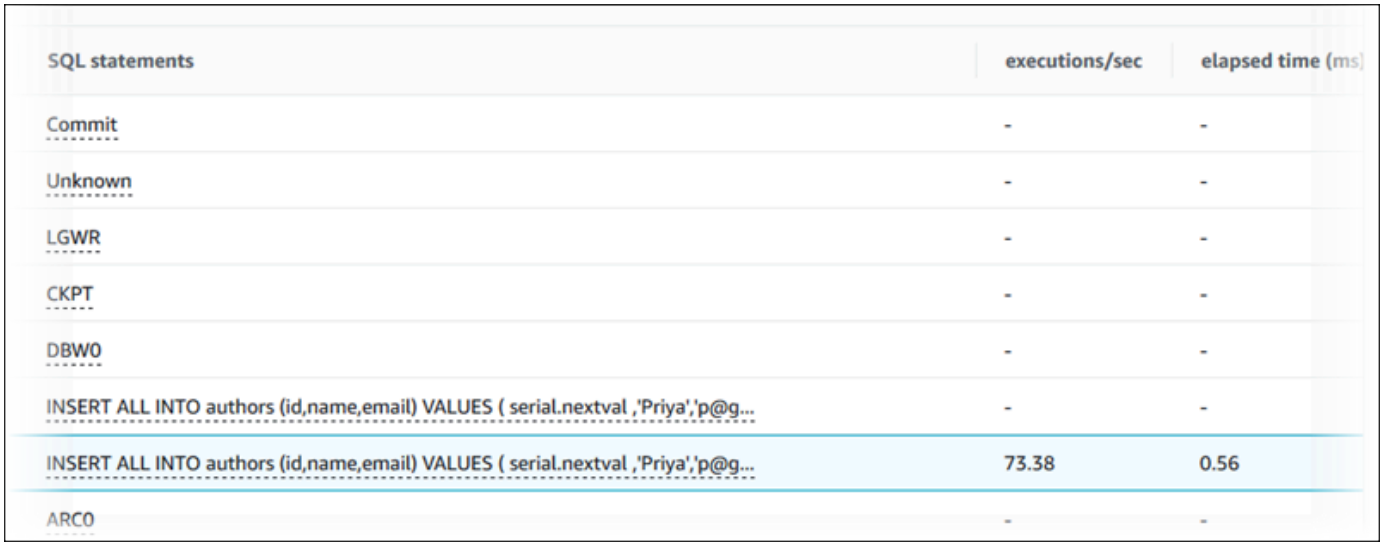
次の例は、MariaDB および MySQL DB インスタンスの設定を示しています。



7. 設定を保存するには [保存] を選択します。

   [トップ SQL] テーブルが更新されます。

   次の例は、Oracle SQL クエリの統計を示しています。



SQL statements	executions/sec	elapsed time (ms)
Commit	-	-
Unknown	-	-
LGWR	-	-
CKPT	-	-
DBW0	-	-
INSERT ALL INTO authors (id,name,email) VALUES (serial.nextval ,'Priya','p@g...	-	-
INSERT ALL INTO authors (id,name,email) VALUES (serial.nextval ,'Priya','p@g...	73.38	0.56
ARC0	-	-

Oracle PDB の上位負荷の分析

Oracle コンテナ DB (CDB) の負荷を分析する際、DB 負荷に最も寄与しているプラグ可能なデータベース (PDB) を特定できます。また、同様のクエリを実行している個々の PDB のパフォーマンスを比較して、パフォーマンスを微調整することもできます。Oracle CDB の詳細については、「Oracle データベースアーキテクチャの RDS」を参照してください。

Amazon RDS Performance Insights ダッシュボードでは、プラグ可能なデータベースに関する情報は、[ディメンション] タブの [上位 PDB] タブにあります。

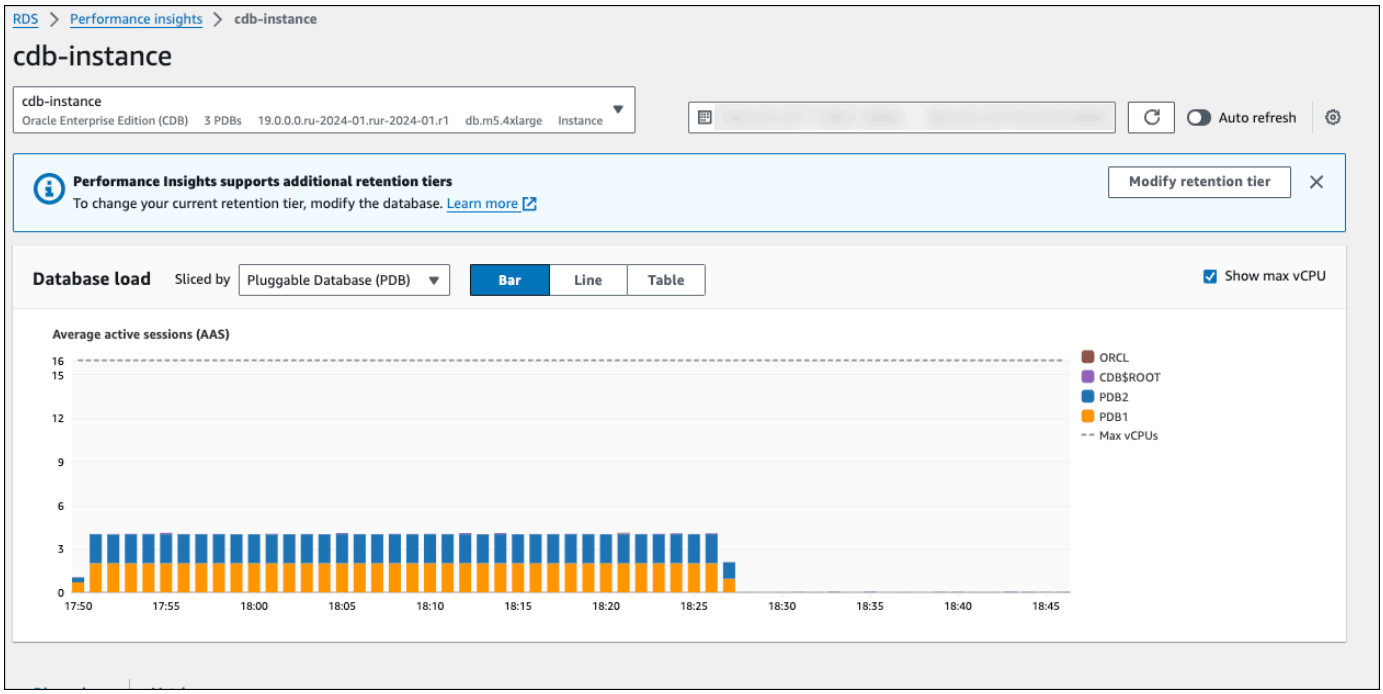
この機能のリージョン、DB エンジン、およびインスタンスクラスのサポート情報については、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

Oracle CDB の上位 PDB 負荷を分析するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. Oracle CDB インスタンスを選択します。

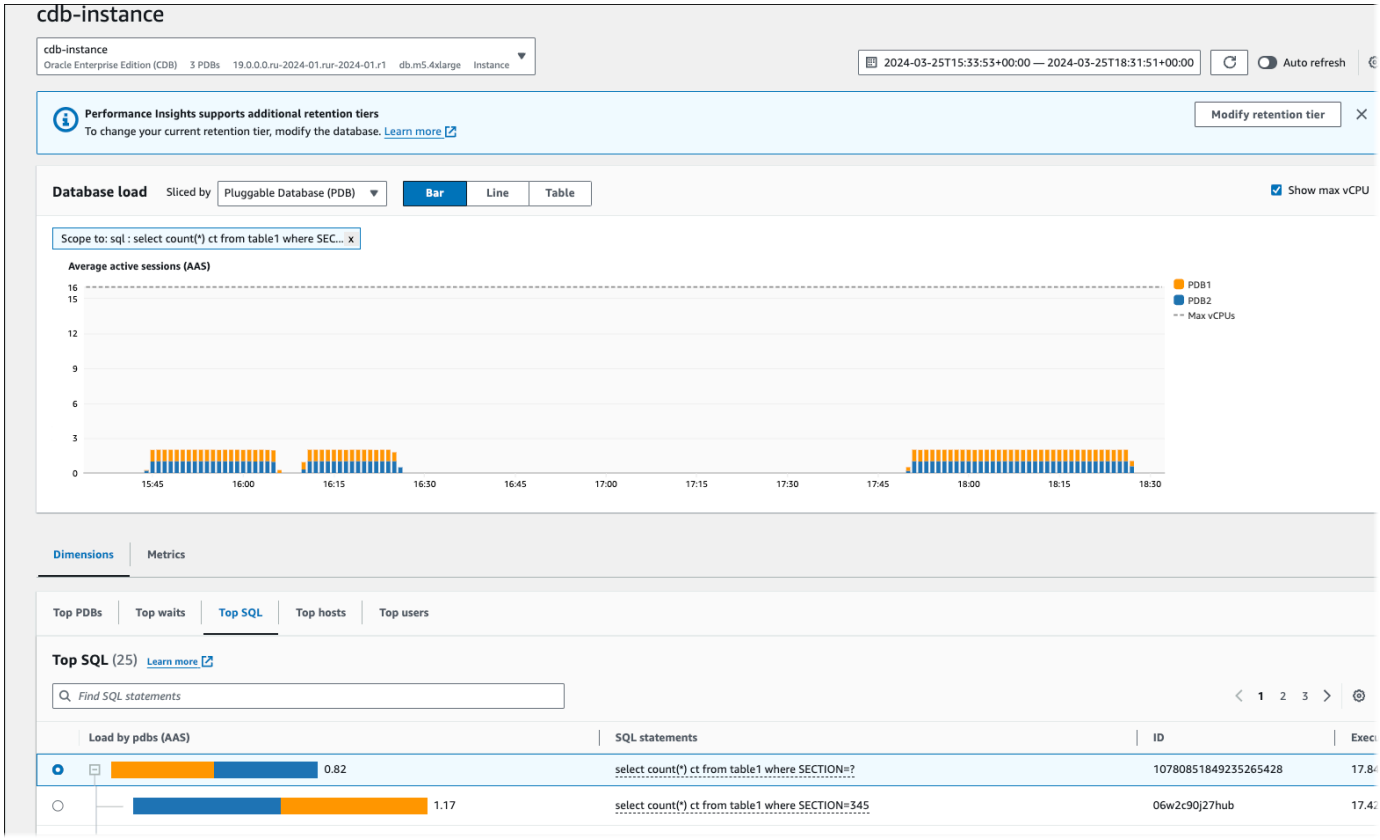
   この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. [データベース負荷 (DB 負荷)] セクションで、[スライス基準] の横にある [プラグ可能なデータベース (PDB)] を選択します。

平均アクティブセッショングラフには、負荷が最も高い PDB が表示されます。PDB 識別子は、色分けされた四角形の右側に表示されます。各識別子は PDB を一意に識別します。

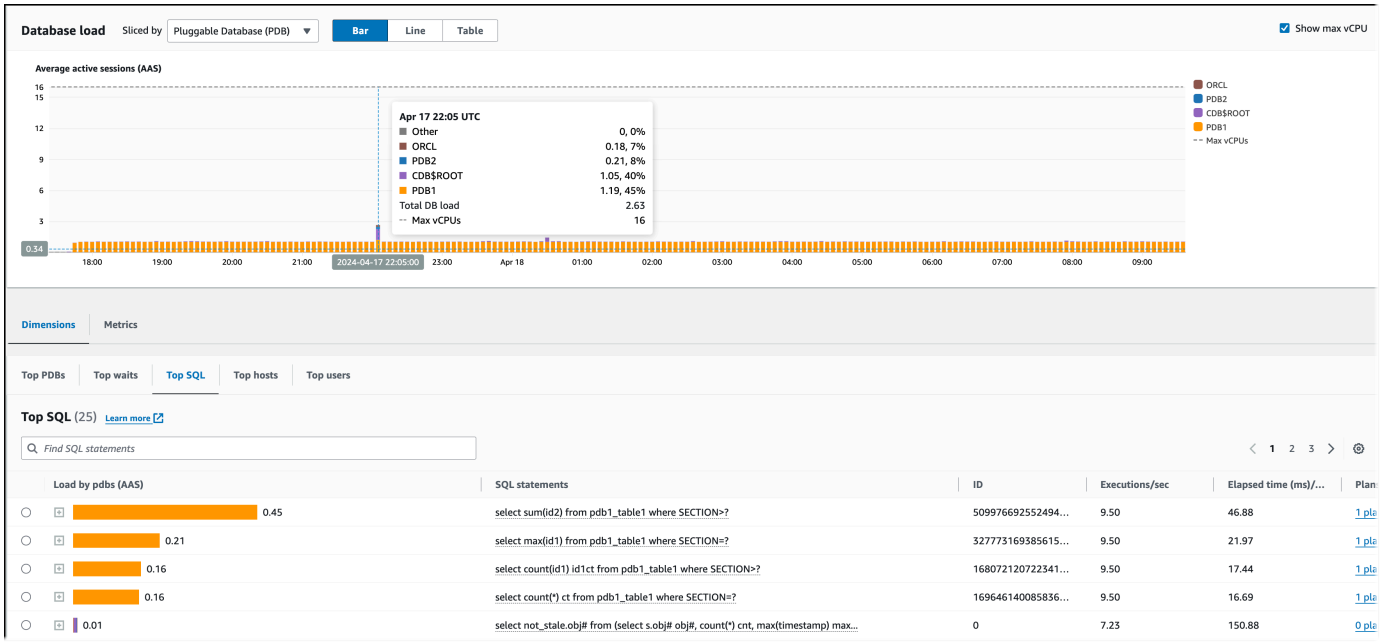


5. 下部にある [トップ SQL] タブまでスクロールします。

次の例では、同じ SQL クエリと、それが複数の PDB に駆動する負荷を確認できます。



次の例では、1 つの PDB が CDB 内の他の PDB よりも高い負荷を処理しています。



Oracle CDB の詳細については、「CDBs and PDBs」を参照してください。

Amazon RDS の Performance Insights ダッシュボードを使用した実行プランの分析

Amazon RDS Performance Insights ダッシュボードでは、Oracle および SQL Server DB インスタンスの実行プランに関する情報を確認できます。この情報を使用して、どのプランが DB 負荷に最も影響しているかを知ることができます。

Oracle または SQL Server の実行プランを分析するには、以下のトピックを参照してください。

実行プランの分析

- Amazon RDS の Performance Insights ダッシュボードを使用した Oracle 実行プランの分析
- Amazon RDS の Performance Insights ダッシュボードを使用した SQL Server 実行プランの分析

Amazon RDS の実行プランの分析の概要

Amazon RDS Performance Insights ダッシュボードを使用して、Oracle および SQL Server DB インスタンスの DB 負荷に最も影響しているプランを確認できます。

例えば、特定の時点の上位の SQL ステートメントが、次の表に示すプランを使用している場合があります。

上位の SQL	計画
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 10	プラン A
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 521	プラン B
SELECT SUM(s_total) FROM sales WHERE region = 10	プラン A
SELECT * FROM emp WHERE emp_id = 1000	プラン C
SELECT SUM(amount_sold) FROM sales WHERE prod_id = 72	プラン A

Performance Insights のプランニング機能を使用すると、以下を実行できます。

- 上位 SQL クエリでどのプランが使用されているかを確認する。

例えば、DB の負荷の大部分は、プラン A とプラン B を使用したクエリによって生成され、プラン C を使用する割合はごく小規模だということがあります。

- 同じクエリについてさまざまなプランを比較する。

  前の例では、製品 ID を除けば、3 つのクエリは同一です。この内 2 つのクエリでプラン A が使用されますが、1 つのクエリではプラン B を使用しています。2 つのプランの違いを確認するには、Performance Insights を使用します。

- クエリが新しいプランに切り替わった時期を確認する。

  プラン A を使用するクエリで、特定のタイミングでプラン B に切り替えられたことを確認できます。この時点で、データベースに変更があったかどうか。例えば、テーブルが空の場合、オプティマイザはテーブル全体に対するスキャンを選択することがあります。仮にテーブルに 100 万行がロードされたとすると、オプティマイザはインデックス範囲のスキャンに切り替えます。

- コストが最も高いプランの特定のステップにドリルダウンします。

  例えば、実行時間の長いクエリで、等価結合に結合条件がないと表示される場合があります。この欠落条件により、2 つのテーブルのすべての行を結合するデカルト結合が強制的に実行されます。

上記のタスクは、Performance Insights のプランキャプチャ機能を使用して実行できます。クエリは、待機イベントと上位 SQL でスライスできるのと同様に、プランのディメンションでスライスできます。

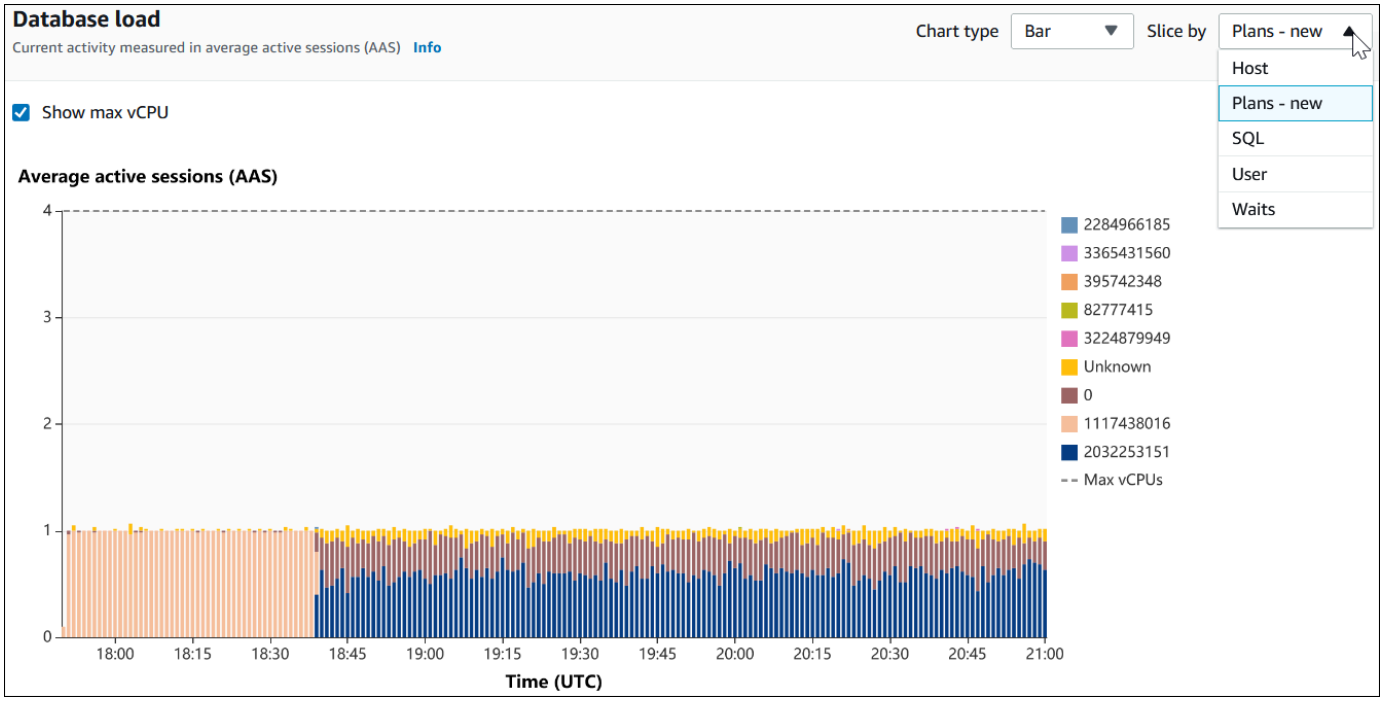
Amazon RDS の Performance Insights ダッシュボードを使用した Oracle 実行プランの分析

Oracle Database の DB 負荷を分析する際に、DB 負荷に最も影響しているプランを知りたい場合があります。DB 負荷に最も影響しているプランを特定するには、Performance Insights のプランキャプチャ機能を使用できます。

コンソールを使用して Oracle 実行プランを分析するには

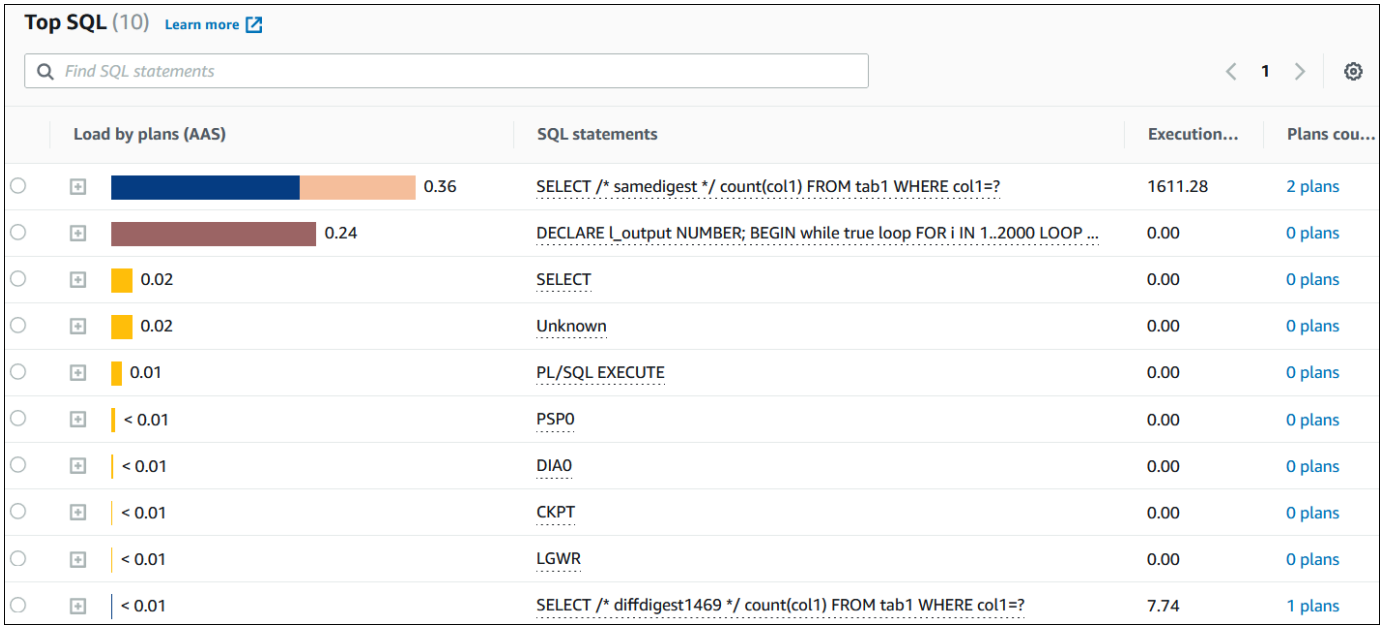
1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. Oracle DB インスタンスを選択します。この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. [データベース負荷 (DB 付加)] セクションで、[スライス基準] の横にある [プラン] をクリックします。

平均アクティブセッションのグラフには、上位 SQL ステートメントで使用されているプランが表示されます。プランのハッシュ値は、色分けされた四角形の右側に表示されます。各ハッシュ値によりプランを一意に識別できます。



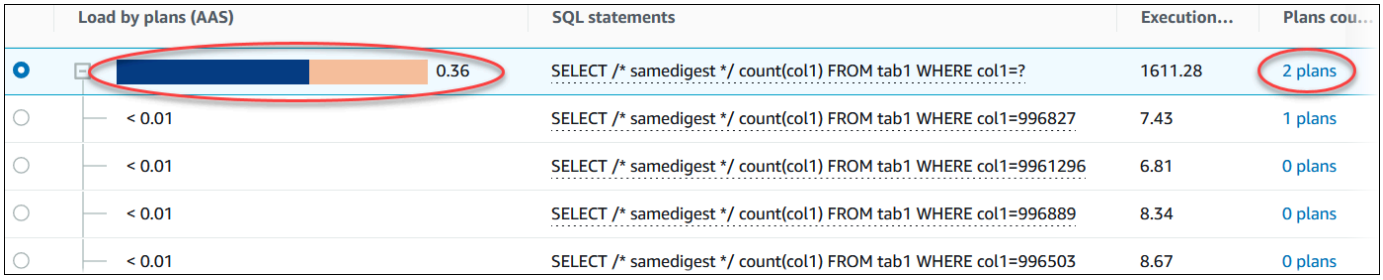
5. 下部にある [トップ SQL] タブまでスクロールします。

次の例では、上位 SQL ダイジェストに 2 つのプランがあります。ステートメント内で疑問符を付けることで、それがダイジェストだと伝えることができます。



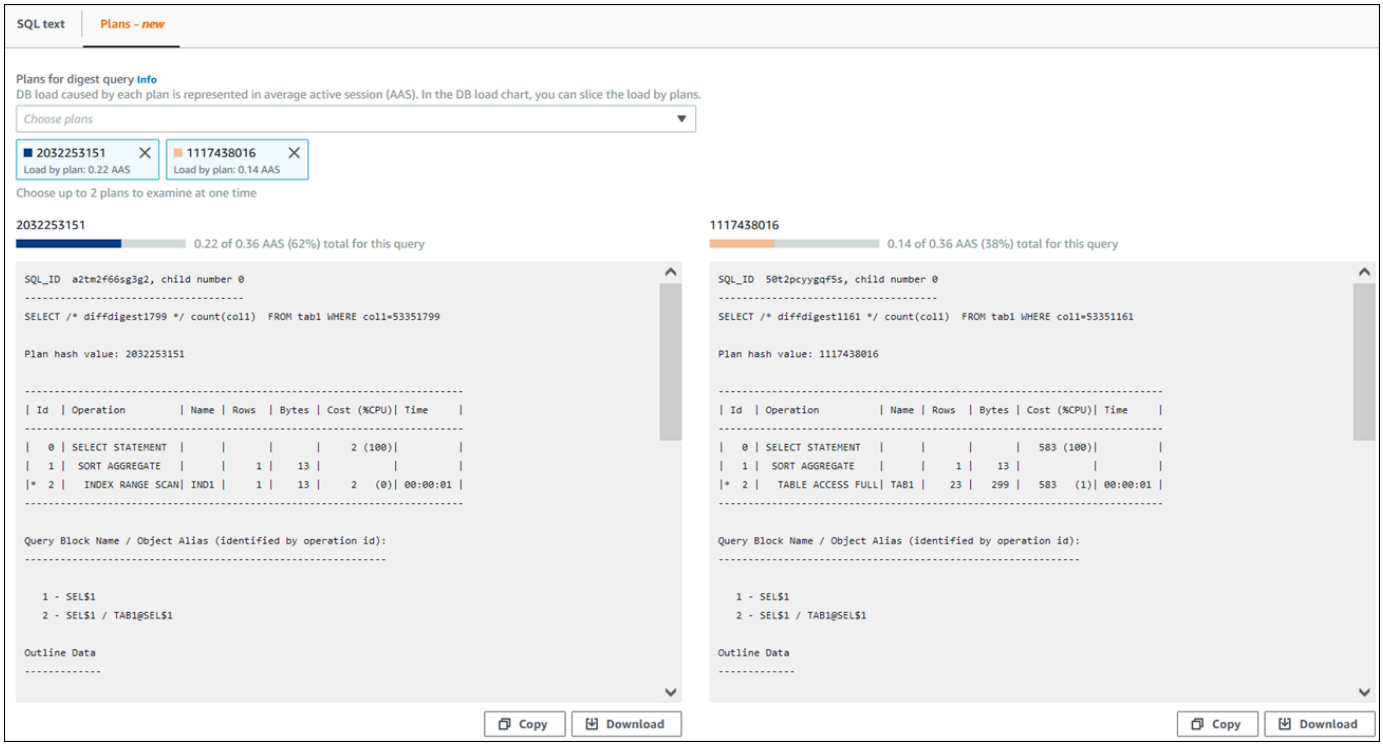
6. ダイジェストを選択して、そのコンポーネントステートメントを展開します。

   以下の例では、SELECT ステートメントがダイジェストクエリです。ダイジェスト内のコンポーネントクエリでは、2 つの異なるプランが使用されます。プランに付けられた色は、データベースの負荷チャートのものと対応しています。ダイジェスト内のプランの総数が、2 列目に表示されます。

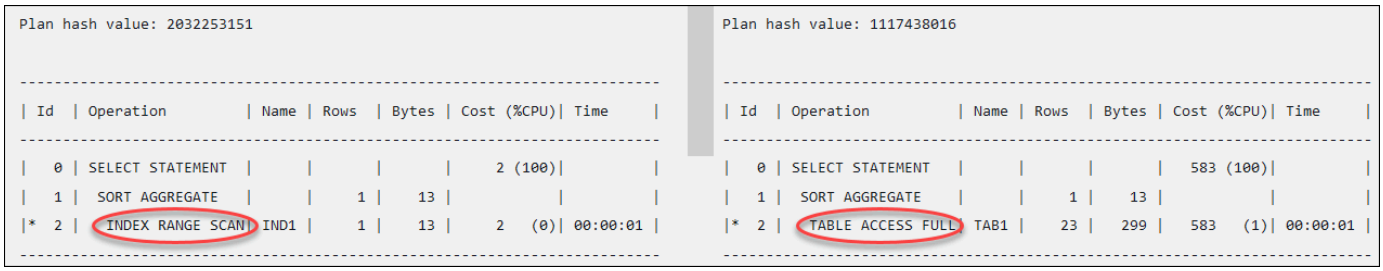


7. 下にスクロールし、[ダイジェストクエリのプラン] リストから、比較する 2 つの [プラン] を選択します。

   1 つのクエリについて、一度に 1 つまたは 2 つのプランを表示できます。次のスクリーンショットは、ダイジェスト内で、ハッシュ 2032253151 とハッシュ 1117438016 を含む 2 つのプランを比較しています。次の例では、このダイジェストクエリを実行する平均アクティブセッションの 62% が左側のプランを使用しているのに対し、38% は右側のプランを使用しています。



この例では、それぞれのプランに重要な相違点があります。プラン 2032253151 のステップ 2 ではインデックススキャンが使用されますが、プラン 1117438016 ではフルテーブルスキャンが使用されます。ほとんどの場合、行数が多いテーブルでは、インデックススキャンを使用すると 1 行のクエリを高速化できます。



```
Plan hash value: 2032253151

-------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |       |       |     2 (100)|          |
|   1 |  SORT AGGREGATE   |      |     1 |    13 |            |          |
|*  2 |   INDEX RANGE SCAN| IND1 |     1 |    13 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

```
Plan hash value: 1117438016

--------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |   583 (100)|          |
|   1 |  SORT AGGREGATE    |      |     1 |    13 |            |          |
|*  2 |   TABLE ACCESS FULL| TAB1 |    23 |   299 |   583   (1)| 00:00:01 |
--------------------------------------------------------------------------------
```

8. (オプション) [コピー] をクリックし、クリップボードにプランをコピーします。あるいは、[ダウンロード] により、ハードドライブにプランを保存します。

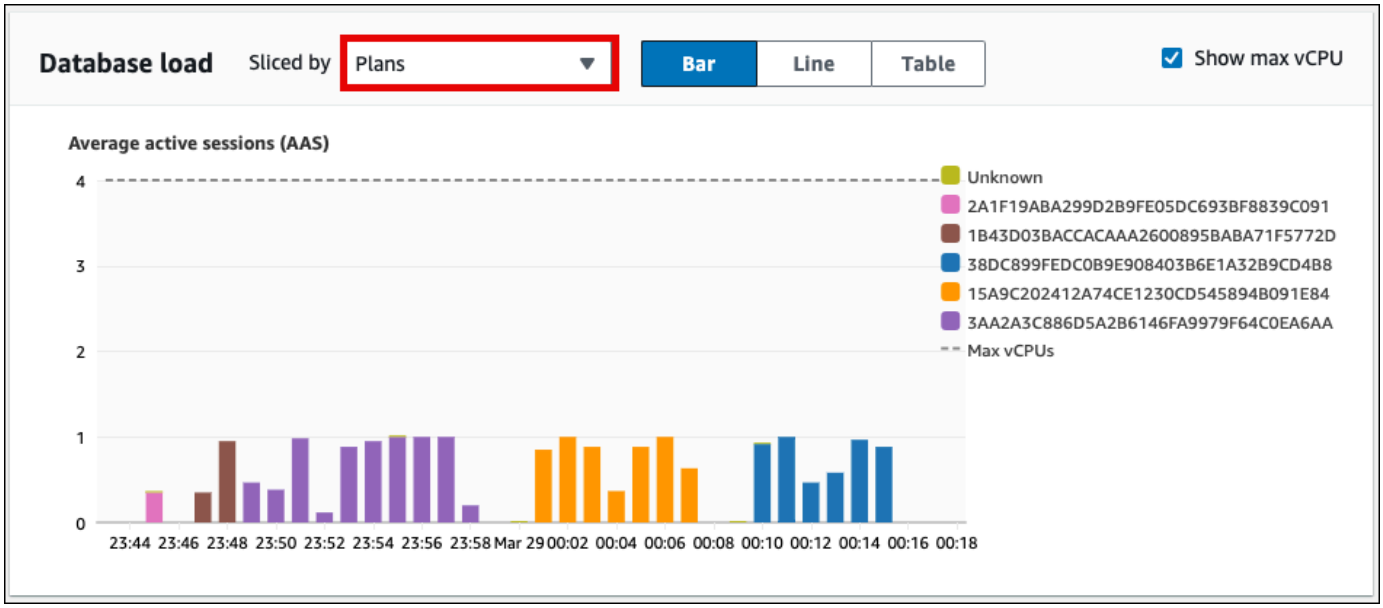
Amazon RDS の Performance Insights ダッシュボードを使用した SQL Server 実行プランの分析

SQL Server データベースの DB 負荷を分析する際に、DB 負荷に最も影響しているプランを確認することもできます。DB 負荷に最も影響しているプランを特定するには、Performance Insights のプランキャプチャ機能を使用できます。

コンソールを使用して SQL Server 実行プランを分析するには

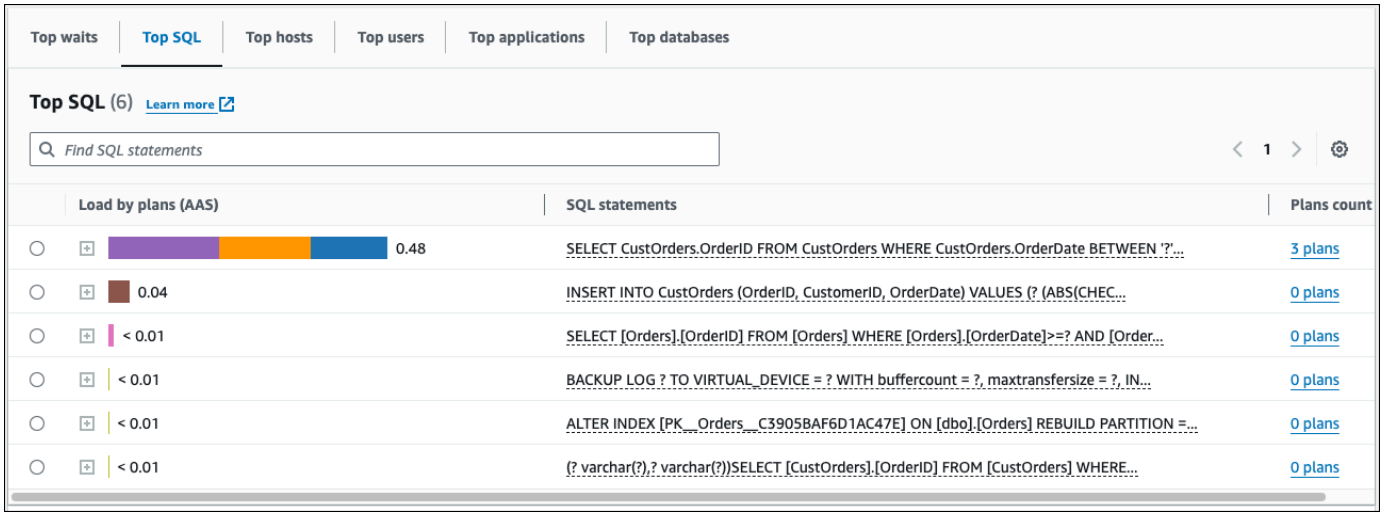
1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Performance Insights] を選択します。
3. SQL Server DB インスタンスを選択します。この DB インスタンスに Performance Insights ダッシュボードが表示されます。
4. [データベース負荷 (DB 付加)] セクションで、[スライス基準] の横にある [プラン] をクリックします。

   平均アクティブセッションのグラフには、上位 SQL ステートメントで使用されているプランが表示されます。プランのハッシュ値は、色分けされた四角形の右側に表示されます。各ハッシュ値によりプランを一意に識別できます。



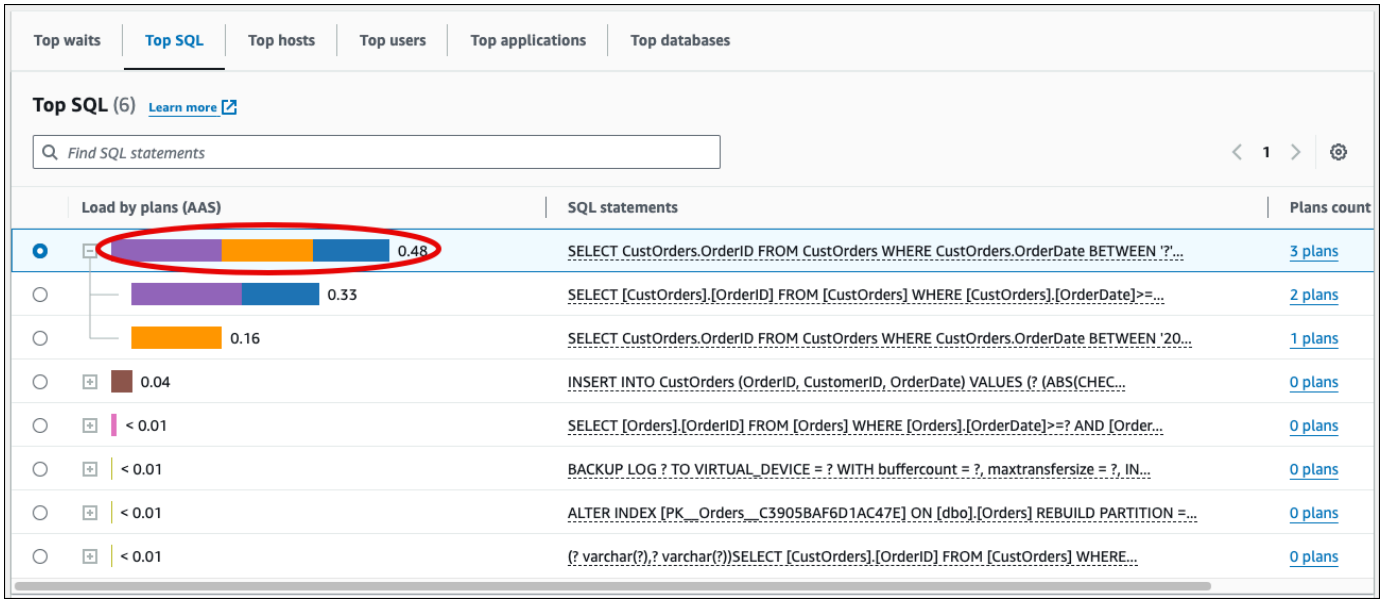
5. 下部にある [トップ SQL] タブまでスクロールします。

   次の例では、上位の SQL ダイジェストに 3 つのプランがあります。SQL ステートメントに疑問符が付いている場合は、そのステートメントがダイジェストであることを示しています。SQL ステートメント全体を表示するには、[SQL ステートメント] 列の値を選択します。



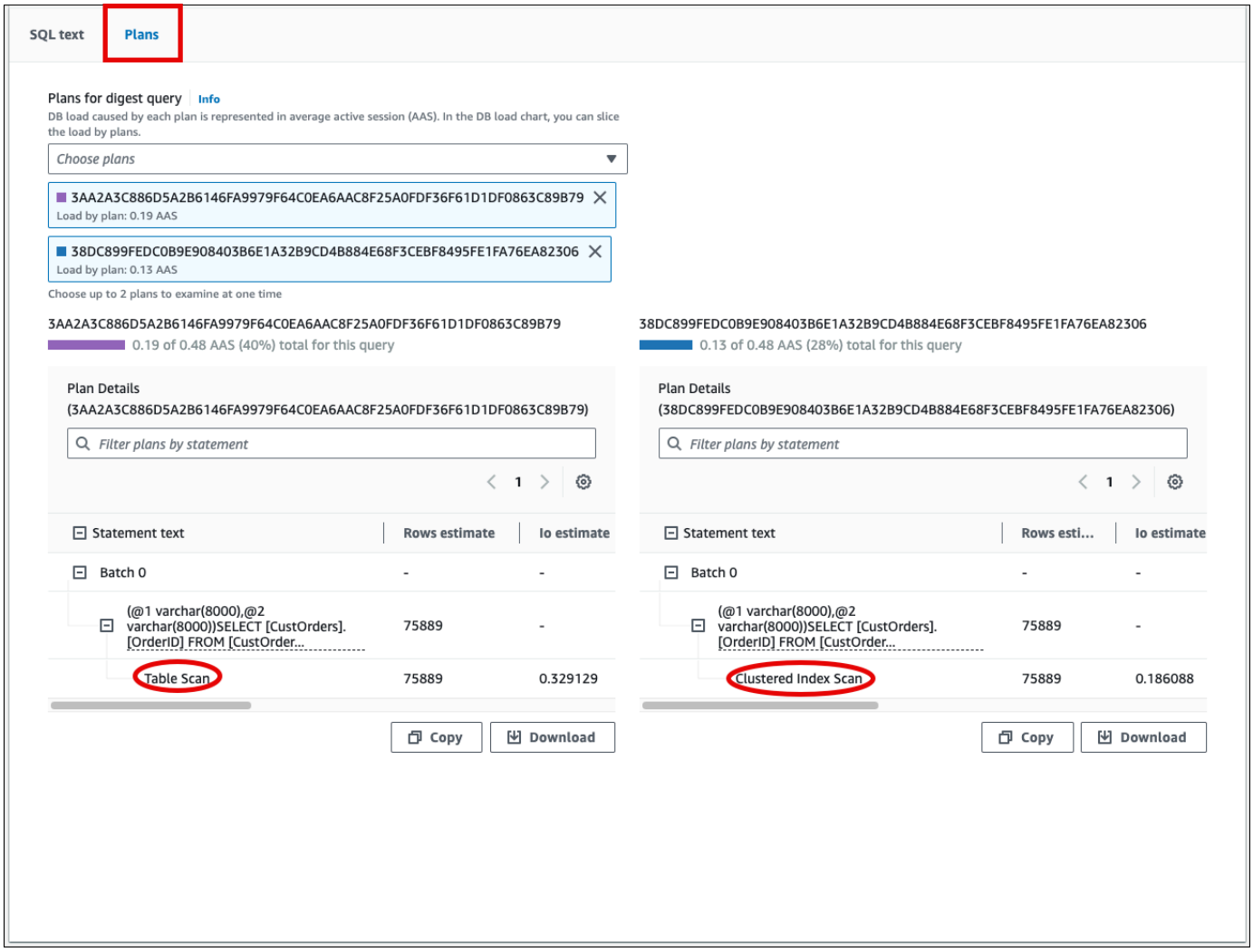
6. ダイジェストを選択して、そのコンポーネントステートメントを展開します。

   以下の例では、SELECT ステートメントがダイジェストクエリです。ダイジェスト内のコンポーネントクエリでは、3 つの異なる実行プランを使用します。プランに付けられた色は、データベースの負荷チャートに対応しています。



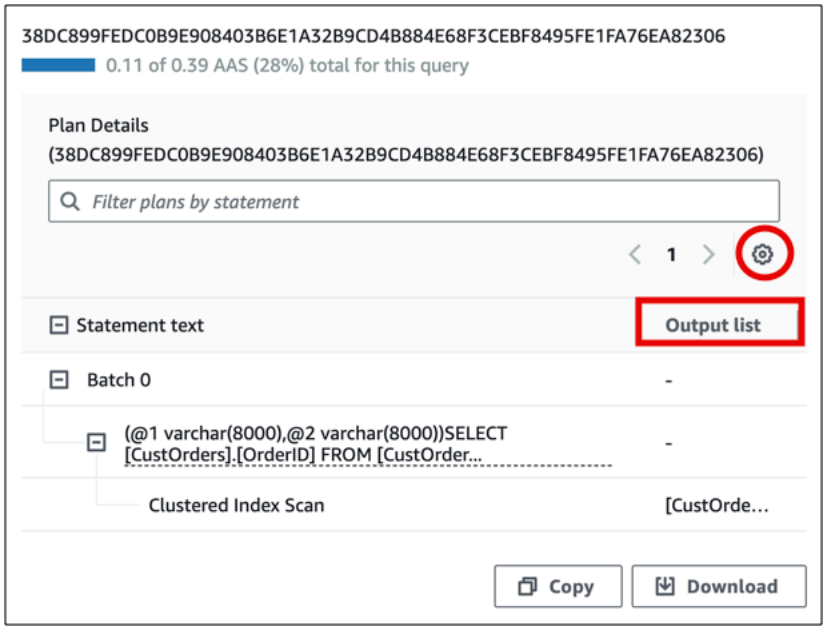
7. 下にスクロールし、[ダイジェストクエリのプラン] リストから、比較する 2 つの [プラン] を選択します。

   1 つのクエリについて、一度に 1 つまたは 2 つのプランを表示できます。次のスクリーンショットは、ダイジェスト内の 2 つのプランを比較しています。次の例では、このダイジェストクエリを実行する平均アクティブセッションの 40% が左側のプランを使用し、28% が右側のプランを使用しています。



前の例では、プラン間に重要な違いがあります。左側のプランのステップ 2 ではテーブルスキャンを使用しているのに対し、右側のプランではクラスター化インデックススキャンを使用しています。行数が多いテーブルでは、ほとんどの場合、1 行を取得するクエリはクラスター化インデックススキャンより高速です。

8. (オプション) [プランの詳細] テーブルの [設定] アイコンを選択して、列の表示と順序をカスタマイズします。次のスクリーンショットでは、[プランの詳細] テーブルに [出力リスト] 列が 2 番目の列として表示されています。



9. (オプション) [コピー] をクリックし、クリップボードにプランをコピーします。あるいは、[ダウンロード] により、ハードドライブにプランを保存します。

> **Note**
>
> Performance Insights は、階層ツリーテーブルを使用して推定実行プランを表示します。このテーブルには、各ステートメントの部分的な実行情報が含まれています。[プランの詳細] テーブルの列の詳細については、SQL Server ドキュメントの「SET SHOWPLAN_ALL」を参照してください。推定実行プランの実行情報全体を表示するには、[ダウンロード] を選択してプランをダウンロードし、これを SQL Server Management Studio にアップロードします。SQL Server Management Studio を使用して推定実行プランを表示する方法の詳細については、SQL Server ドキュメントの「Display an Estimated Execution Plan」を参照してください。

Performance Insights のプロアクティブ推奨事項の表示

Amazon RDS Performance Insights は特定のメトリクスを監視し、指定されたリソースで潜在的に問題と見なされる可能性があるレベルを分析することで自動的にしきい値を作成します。新しいメトリクス値が事前定義されたしきい値を一定期間にわたって超えた場合に、Performance Insights はプロアクティブ推奨事項を生成します。この推奨事項は、将来のデータベースパフォーマンスへの影響を防ぐのに役立ちます。これらのプロアクティブ推奨事項を受け取るには、有料利用枠の保持期間を使用して Performance Insights を有効にする必要があります。

Performance Insights をオンにする方法の詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。Performance Insights の料金とデータ保持については、「Performance Insights の料金とデータ保持」を参照してください。

プロアクティブ推奨事項でサポートされているリージョン、DB エンジン、インスタンスクラスについては、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

プロアクティブ推奨事項の詳細な分析と推奨される調査は、推奨事項の詳細ページで確認できます。

推奨事項の詳細については、「Amazon RDS の推奨事項」を参照してください。

プロアクティブ推奨事項の詳細な分析を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、次のいずれかを実行します。
   - [レコメンデーション] を選択します。

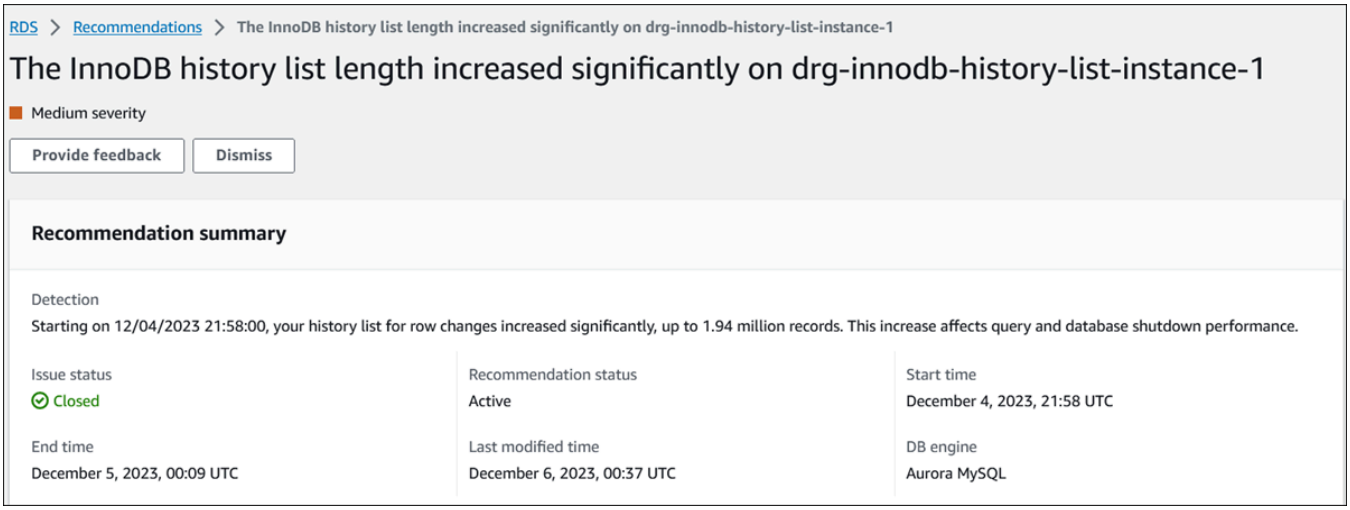
     [レコメンデーション] ページには、アカウント内のすべてのリソースの重要度でソートされた推奨事項のリストが表示されます。
   - [データベース] を選択し、データベースページでリソースの [レコメンデーション] を選択します。

     [レコメンデーション] タブには、選択したリソースの推奨事項とその詳細が表示されます。
3. プロアクティブ推奨事項を検索し、[詳細の表示] を選択します。

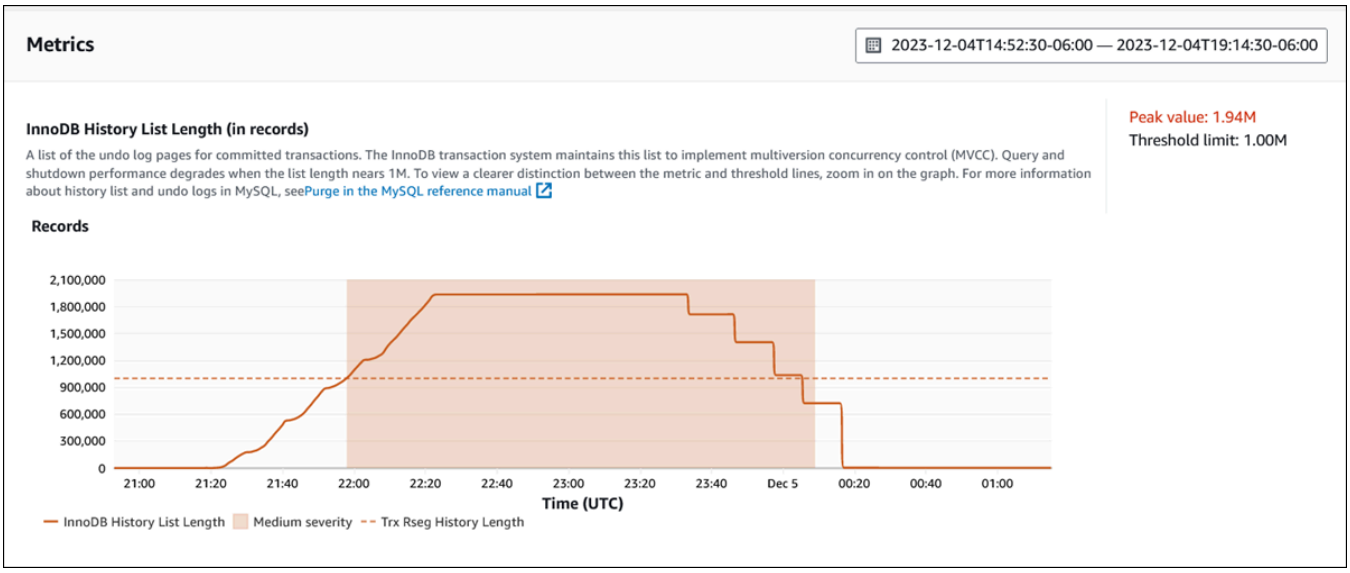
   推奨事項の詳細ページが表示されます。タイトルには、影響を受けたリソースの名前と、検出された問題および重要度が表示されます。

   推奨事項の詳細ページのコンポーネントは次のとおりです。

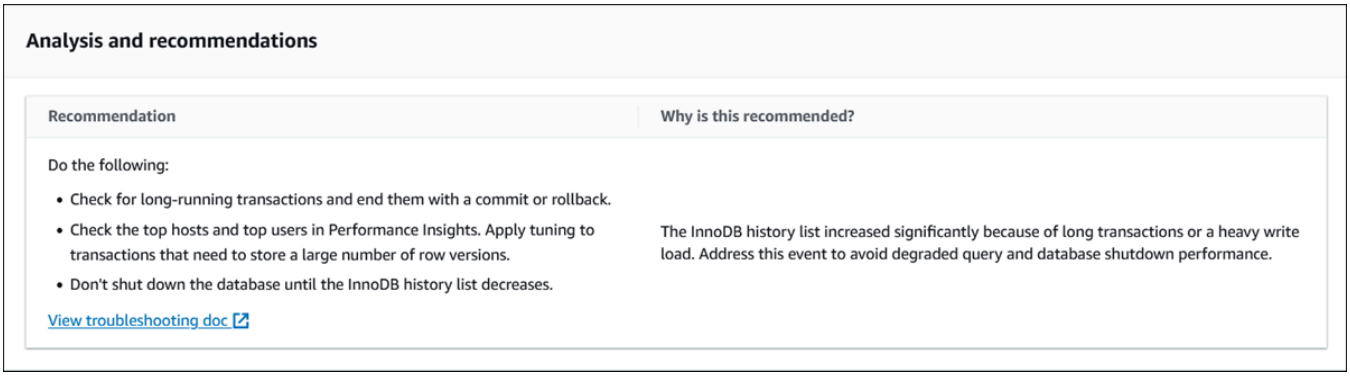
   - [推奨事項の概要] – 検出された問題、推奨事項と問題のステータス、問題の開始時刻と終了時刻、推奨事項の変更時刻、エンジンタイプ。



- [メトリクス] – 検出された問題のグラフ。各グラフには、リソースのベースライン動作によって決まるしきい値と、問題の開始時刻から報告されたメトリクスのデータが表示されます。



- [分析と推奨事項] — 推奨事項と示唆された推奨事項の理由。



問題の原因を確認し、示唆された推奨アクションを実行して問題を解決するか、[閉じる] を選択して推奨事項を却下できます。

Performance Insights API for Amazon RDS によるメトリクスの取得

Performance Insights がオンになっている場合、API はインスタンスのパフォーマンスを可視化します。Amazon CloudWatch Logs は、AWS のサービスをモニタリングしたメトリクスの信頼性のある提供元です。

Performance Insightsは、平均アクティブ・セッション(AAS)として測定されるデータベースロードのドメイン固有のビューを提供します。このメトリクスはAPI利用者には2次元時系列データセットのように見えます。データの時間ディメンションは、クエリされた時間範囲内の各時点のDB口ード・データを提供します。各時点で、その時点で計測された SQL、Wait-event、User、Host などのリクエストされたディメンションに関する負荷全体が分解されます。

Amazon RDS Performance Insights では、Amazon RDS DB インスタンス をモニタリングし、データベースパフォーマンスの分析とトラブルシューティングを行うことができます。Performance Insights は、AWS マネジメントコンソール で表示することができます。また、Performance Insights では独自のデータをクエリできるように、パブリック API も提供されています。API を使用して、次を実行できます。

- データベースにデータをオフロードする
- Performance Insights データを既存のモニタリングダッシュボードに追加する
- モニタリングツールを構築する

Performance Insights API を使用するには、いずれかの Amazon RDS DB インスタンスで Performance Insights を有効にします。Performance Insights の有効化については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。Performance Insights API の詳細については、「Amazon RDS Performance Insights API リファレンス」を参照してください。

Performance Insights API は、以下のオペレーションを提供します。

Performance Insights でのアクション	AWS CLI コマンド	説明
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	DB インスタンスの特定の期間のパフォーマンス分析レポートを作成します。結果は、レポートの固有識別子である AnalysisReportId です。
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	パフォーマンス分析レポートを削除します。
DescribeDimensionKeys	aws pi describe-dimension-keys	特定の期間に、メトリクスの上位 N 個のディメンションキーを取得します。
GetDimensionKeyDetails	aws pi get-dimension-key-details	DB インスタンスまたはデータソースの指定されたディメンショングループの属性を取得します。例えば、SQL ID を指定し、ディメンションの詳細が使用可能な場合、GetDimensionKeyDetails は、この ID に関連付けられているディメンション db.sql.statement の全文を取得します。このオペレーションは、GetResourceMetrics および DescribeDimensionKeys が大きな SQL ステートメントテキストの取得をサポートしないため、便利です。

Performance Insights でのアクション	AWS CLI コマンド	説明
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	レポートのインサイトを含むレポートを取得します。結果には、レポートのステータス、レポート ID、レポート時間の詳細、インサイト、および推奨事項が含まれます。
GetResourceMetadata	aws pi get-resource-metadata	さまざまな機能に関するメタデータを取得します。例えば、メタデータにより、特定の DB インスタンスで何等かの機能が有効化されているか無効化されているかを、示すことができます。
GetResourceMetrics	aws pi get-resource-metrics	期間中、データソースのセットに Performance Insights のメトリクスを取得します。特定のディメンショングループおよびディメンションを提供し、各グループの集約とフィルタリング条件を提供することができます。
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	指定したインスタンスで、指定したメトリクスタイプごとにクエリできるディメンションを取得します。
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	DB インスタンスを指定しながら、指定されたメトリクスタイプでクエリが可能なメトリクスをすべて取得します。

Performance Insights でのアクション	AWS CLI コマンド	説明
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	DB インスタンスについて入手可能なすべての分析レポートを取得します。レポートは、各レポートの開始時間に基づいて一覧表示されます。
ListTagsForResource	aws pi list-tags-for-resource	リソースに追加されたすべてのメタデータタグを一覧表示します。リストには、タグの名前と値が含まれます。
TagResource	aws pi tag-resource	Amazon RDS リソースにメタデータタグを追加します。タグには名前と値が含まれます。
UntagResource	aws pi untag-resource	メタデータタグをリソースから削除します。

Performance Insights の時系列メトリクスの取得と AWS CLI の例の詳細については、以下のトピックを参照してください。

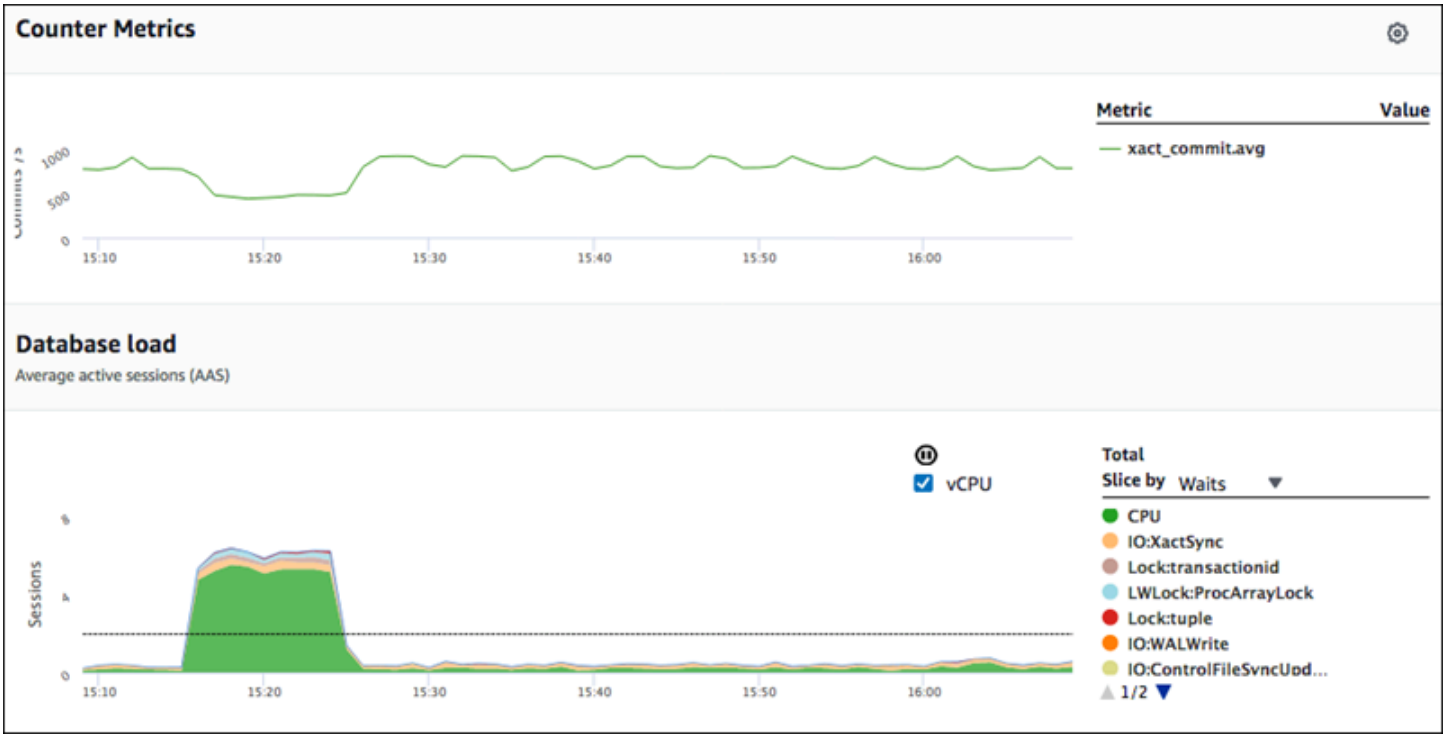
トピック

- Performance Insights の時系列メトリクスの取得
- AWS CLIPerformance Insights での の例

Performance Insights の時系列メトリクスの取得

`GetResourceMetrics` オペレーションでは、1 つ以上の時系列メトリクスを Performance Insights データから取得します。`GetResourceMetrics` には、メトリクスおよび期間が必要であり、データポイントのリストを含むレスポンスが返ります。

例えば、AWS マネジメントコンソール は、次のイメージのように、[カウンターメトリクス] チャートと [データベースロード] チャートの入力に `GetResourceMetrics` を使用します。



GetResourceMetrics によって返るメトリクスはすべて、db.load の例外を除き、スタンダードの時系列メトリクスです。このメトリクスは、[データベースロード] グラフに表示されます。この db.load メトリクスは、ディメンションと呼ばれるサブコンポーネントに分割できるため、他の時系列メトリクスとは異なります。前のイメージでは、db.load は分割され、db.load を構成する待機状態によってグループ化されています。

> **Note**
>
> GetResourceMetrics は、db.sampleload メトリクスを返すこともできますが、通常 db.load メトリクスが適切です。

GetResourceMetrics により返されるカウンターメトリクスに関する情報は、「Performance Insights カウンターメトリクス」を参照してください。

以下の計算は、メトリクスにサポートされています。

- 平均 - 期間中のメトリクスの平均値。.avg をメトリクス名に追加します。
- 最小 - 期間中のメトリクスの最小値。.min をメトリクス名に追加します。
- 最大 - 期間中のメトリクスの最大値。.max をメトリクス名に追加します。
- 合計 - 期間中のメトリクス値の合計。.sum をメトリクス名に追加します。

- サンプル数 - 期間中にメトリクスが収集された回数。`.sample_count` をメトリクス名に追加します。

例えば、メトリクスが 300 秒 (5 分) 収集され、メトリクスが 1 分に 1 回収集されたものと見なします。毎分の値は、1、2、3、4、5 です。この場合、以下の計算が返されます。

- 平均 - 3
- 最小 - 1
- 最大 - 5
- 合計 - 15
- サンプル数 - 5

`get-resource-metrics` AWS CLI コマンドの使用の詳細については、「`get-resource-metrics`」を参照してください。

`--metric-queries` オプションでは、結果を取得する 1 つ以上のクエリを指定します。各クエリは、必須の `Metric` と、オプションの `GroupBy` および `Filter` パラメータから構成されます。`--metric-queries` オプションの指定の例を次に示します。

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLIPerformance Insights での の例

以下のセクションでは、Performance Insights の AWS Command Line Interface (AWS CLI) の詳細と AWS CLI の使用例について説明します。

トピック

- Performance Insights の AWS CLI の組み込みヘルプ
- カウンターメトリクスの取得

- 上位の待機イベントに関する DB 平均負荷の取得
- 上位の SQL に関する DB 平均負荷の取得
- SQL によってフィルタリングされた平均 DB ロードの取得
- SQL ステートメントの全文の取得
- 一定期間のパフォーマンス分析レポートの作成
- パフォーマンス分析レポートの取得
- DB インスタンスのすべてのパフォーマンス分析レポートを一覧表示する
- パフォーマンス分析レポートの削除
- パフォーマンス分析レポートにタグを追加する
- パフォーマンス分析レポートのすべてのタグを一覧表示する
- パフォーマンス分析レポートからタグを削除する

Performance Insights の AWS CLI の組み込みヘルプ

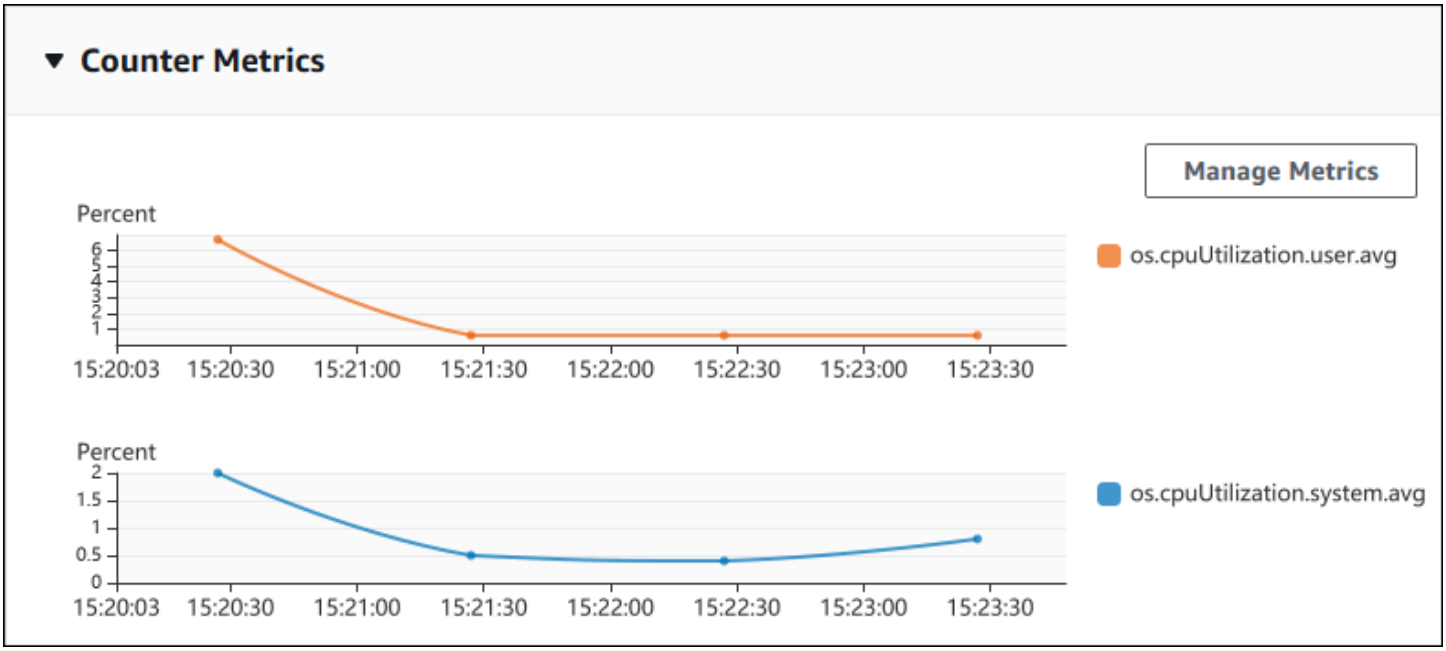
Performance Insights は、AWS CLI を使用して表示することができます。Performance Insights の AWS CLI コマンドのヘルプを表示するには、コマンドラインで次のように入力します。

```
aws pi help
```

AWS CLI がインストールされていない場合は、「AWS CLI ユーザーガイド」の「AWS CLI のインストール」でインストールの方法を確認してください。

カウンターメトリクスの取得

以下のスクリーンショットは、AWS マネジメントコンソール における 2 つのカウンターメトリクスグラフを示します。



以下の例では、2 つのカウンターメトリクスグラフを生成するために AWS マネジメントコンソールで使用するデータと同じデータを生成する方法を示します。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

また、コマンドを作成しやすくするために、--metrics-query オプションにファイルを指定します。以下の例では、このオプション用に query.json と呼ばれるファイルを使用します。ファイルの内容は次のとおりです。

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

ファイルを使用するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

前述の例では、各オプションに次の値を指定します。

- --service-type - RDS for Amazon RDS
- --identifier - DB インスタンスのリソース ID
- --start-time および --end-time - クエリを実行する期間の ISO 8601 DateTime 値 (サポートされている複数の形式)

クエリは 1 時間の範囲で実行されます。

- `--period-in-seconds` - `60` (1 分ごとのクエリ)
- `--metric-queries` - 2 つのクエリの配列。それぞれ 1 つのメトリクスに対して使用されます。

  メトリクス名ではドットを使用してメトリクスを有用なカテゴリに分類します。最終の要素は関数になります。この例では、関数は、クエリの `avg` です。Amazon CloudWatch と同様に、サポートされている関数は、`min`、`max`、`total`、および `avg` です。

レスポンスは次の例のようになります。

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
```

```
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

レスポンスには、Identifier、AlignedStartTime、AlignedEndTime があります。--period-in-seconds 値が 60 の場合、スタート時間および終了時間は、時間 (分) に調整されます。--period-in-seconds が 3600 の場合、スタート時間および終了時間は、時間 (時) に調整されます。

レスポンスの MetricList には、多数のエントリを含み、それぞれに Key および DataPoints エントリがあります。DataPoint にはそれぞれ、Timestamp および Value を含みます。クエリは 1 分ごとのデータが 1 時間以上実行されるため、Datapoints の各リストには、60 個のデータポイントがあります。これには、Timestamp1/Minute1 や Timestamp2/Minute2 から、Timestamp60/Minute60 まで含まれます。

クエリは 2 つの異なるカウンターメトリクスを対象としているため、レスポンス MetricList には 2 つの要素があります。

上位の待機イベントに関する DB 平均負荷の取得

以下の例は、スタックされたエリアチャートを生成するために AWS マネジメントコンソール で使用されるのと同じクエリです。この例では、上位 7 つの待機イベントに応じて負荷を分割し、最後の 1 時間で db.load.avg を取得します。コマンドは カウンターメトリクスの取得 と同じコマンドです。ただし、query.json ファイルには、次の内容が含まれます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
```

```
]
```

以下のコマンドを実行してください。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

この例では、上位 7 つの待機イベントのうち db.load.avg と GroupBy のメトリクスを指定しています。この例の有効な値の詳細については、Performance Insights の API リファレンスの「DimensionGroup」を参照してください。

レスポンスは次の例のようになります。

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
```

```
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

このレスポンスでは、MetricList の 8 つのエントリがあります。合計の db.load.avg のエントリが 1 つあり、上位 7 つの待機イベントのいずれかに従って分割された db.load.avg のエントリが 7 つあります。初期の例とは異なり、グループ化ディメンションがあったため、メトリクスのグループ化ごとに 1 つのキーが必要です。基本的なカウンターメトリクスのユースケースのように、メトリクスごとに 1 つのキーのみ使用することはできません。

上位の SQL に関する DB 平均負荷の取得

以下の例では、上位 10 個の SQL ステートメント別に db.wait_events をグループ化します。SQL ステートメントには 2 つの異なるグループがあります。

- db.sql - SQL のフルステートメント (例:select * from customers where customer_id = 123)
- db.sql_tokenized - トークン分割された SQL ステートメント select * from customers where customer_id = ?()

データベースのパフォーマンスを分析するときは、パラメータが異なるだけの SQL ステートメントを 1 つの論理的な項目として検討すると便利です。そのため、クエリを実行する際、db.sql_tokenized を使用することができます。ただし、特に Explain Plan を確認したい場合は、パラメータ付きの完全な SQL ステートメントと、db.sql によるクエリのグループ化を調べる方が便利な場合があります。トークン分割化された SQL と完全 SQL の間には親子関係があり、複数の完全 SQL (子) が同じトークン分割化された SQL (親) の下にグループ化されています。

この例のコマンドは、上位の待機イベントに関する DB 平均負荷の取得 のコマンドに似ています。ただし、query.json ファイルには、次の内容が含まれます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

次の例では db.sql_tokenized を使用しています。

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
```

```
    --start-time 2018-10-29T00:00:00Z \
    --end-time   2018-10-30T00:00:00Z \
    --period-in-seconds 3600 \
    --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
    --service-type RDS ^
    --identifier db-ID ^
    --start-time 2018-10-29T00:00:00Z ^
    --end-time   2018-10-30T00:00:00Z  ^
    --period-in-seconds 3600 ^
    --metric-queries file://query.json
```

この例では、1 時間の間隔 (秒) で 24 時間以上のクエリを実行します。

この例では、上位 7 つの待機イベントのうち db.load.avg と GroupBy のメトリクスを指定しています。この例の有効な値の詳細については、Performance Insights の API リファレンスの「DimensionGroup」を参照してください。

レスポンスは次の例のようになります。

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
        },
        {
```

```
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

このレスポンスの MetricList には 11 のエントリがあり (合計が 1 つと、トークン分割された上位 10 項目の SQL)、各エントリには、1 時間あたり 24 の DataPoints があります。

トークン分割された SQL の場合は、各ディメンションリストに 3 つのエントリがあります。

- db.sql_tokenized.statement - トークン分割された SQL ステートメント。
- db.sql_tokenized.db_id - SQL の参照に使用されていたネイティブデータベース ID、または Performance Insights によって生成される合成 ID (ネイティブデータベース ID が利用できない場合)。この例では、pi-2372568224 合成 ID が返ります。
- db.sql_tokenized.id - Performance Insights 内のクエリの ID。

  AWS マネジメントコンソール で、この ID はサポート ID と呼ばれます。ID は、データベースに関する問題のトラブルシューティングに役立つ、AWS サポートが調査できるデータであるため、この名前が付けられています。AWS は、データのセキュリティとプライバシーを非常に重視しており、ほとんどすべてのデータが AWS KMS キーで暗号化されて保存されます。そのため、このデータを AWS 内で見ることはできません。前の例では、tokenized.statement と tokenized.db_id の両方が暗号化されて保存されます。データベースに問題がある場合は、AWS サポートがサポート ID を参照して問題を解決できるようお手伝いします。

クエリを実行する際、Group で GroupBy を指定した方が便利な場合があります。ただし、返るデータを詳細に制御できるように、ディメンションのリストを指定します。例えば、必要なデータが db.sql_tokenized.statement のみの場合は、Dimensions 属性を query.json ファイルに追加することができます。

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

SQL によってフィルタリングされた平均 DB ロードの取得

上のイメージは、特定のクエリが選択されていることを示しています。上位の平均アクティブセッションのスタックされたエリアグラフはそのクエリを対象としています。クエリは、依然として上位 7 つの全体的な待機イベントを対象としていますが、レスポンスの値はフィルタリングされます。フィルタでは、特定のフィルタに一致するセッションのみが考慮されます。

この例に対応する API クエリは、上位の SQL に関する DB 平均負荷の取得 のコマンドに似ています。ただし、query.json ファイルには、次の内容が含まれます。

```
[
```

```
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Linux、macOS、Unix の場合:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Windows の場合:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

レスポンスは次の例のようになります。

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
```

```
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
```

```
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
```

```
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

このレスポンスでは、query.json ファイルで指定されているトークン分割化された SQL AKIAIOSFODNN7EXAMPLE の割合に従って、値はすべてフィルタリングされます。キーは、フィルタなしのクエリとは異なる順序で表示されることもあります。これは、フィルタ処理された SQL に影響を与えるのは上位 5 つの待機イベントであるためです。

SQL ステートメントの全文の取得

次の例では、DB インスタンス db-10BCD2EFGHIJ3KL4M5NO6PQRS5 の SQL ステートメントの全文を取得します。`--group` は `db.sql` であり、`--group-identifier` は `db.sql.id` です。この例では、*my-sql-id* は、`pi get-resource-metrics` または `pi describe-dimension-keys` を呼び出して取得した SQL ID を表します。

以下のコマンドを実行してください。

Linux、macOS、Unix の場合:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Windows の場合:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

この例では、ディメンションの詳細を使用できます。したがって、Performance Insights は、SQL ステートメントを切り捨てることなく、その全文を取得します。

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

一定期間のパフォーマンス分析レポートの作成

次の例では、db-loadtest-0 データベースの 1682969503 開始時間と 1682979503 終了時間を使用してパフォーマンス分析レポートを作成します。

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

レスポンスは、レポートの一意識別子 report-0234d3ed98e28fb17 です。

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

パフォーマンス分析レポートの取得

次の例では、report-0d99cc91c4422ee61 レポートについて分析レポートの詳細を取得します。

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

レスポンスには、レポートのステータス、ID、時間の詳細、およびインサイトが表示されます。

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

DB インスタンスのすべてのパフォーマンス分析レポートを一覧表示する

次の例は、db-loadtest-0 データベースについて入手可能なすべてのパフォーマンス分析レポートを一覧表示します。

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

レスポンスには、すべてのレポートがレポートの ID、ステータス、および時間の詳細とともに一覧表示されます。

```
{
```

```
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

パフォーマンス分析レポートの削除

次の例では、db-loadtest-0 データベースの分析レポートを削除します。

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

パフォーマンス分析レポートにタグを追加する

次の例では、キー name と値 test-tag のタグを report-01ad15f9b88bcbd56 レポートに追加します。

```
aws pi tag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

パフォーマンス分析レポートのすべてのタグを一覧表示する

次の例では、report-01ad15f9b88bcbd56 レポートのすべてのタグを一覧表示します。

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

レスポンスには、レポートに追加されたすべてのタグの値とキーが一覧表示されます。

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

パフォーマンス分析レポートからタグを削除する

次の例では、report-01ad15f9b88bcbd56 レポートから name タグを削除します。

```
aws pi untag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
```

```
--tag-keys name \
--region us-west-2
```

タグを削除した後、`list-tags-for-resource` API を呼び出しても、このタグは一覧に表示されません。

AWS CloudTrail を使用した Performance Insights 呼び出しのログ記録

Performance Insights は AWS CloudTrail サービスと連携して動作します。このサービスは、Performance Insights でユーザー、ロール、または AWS のサービスによって実行されたアクションを記録します。CloudTrail は、Performance Insights のすべての API コールをイベントとしてキャプチャします。このキャプチャには、Amazon RDS コンソールからのコールと、Performance Insights API オペレーションへのコードコールが含まれます。

証跡を作成すると、Performance Insights のイベントも含めて、Amazon S3 バケットへの CloudTrail イベントの継続的配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集されたデータを使用して、多くの情報を判断できます。この情報には、Performance Insights に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時が含まれます。追加の詳細も含まれています。

CloudTrail の詳細については、AWS CloudTrail ユーザーガイドを参照してください。

CloudTrail での Performance Insights 情報の使用

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。Performance Insights でアクティビティが発生すると、そのアクティビティは、CloudTrail イベントとして AWS の他のサービスのイベントとともに、CloudTrail コンソールの [イベント履歴] に記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、AWS CloudTrail ユーザーガイドの「CloudTrail イベント履歴でのイベントの表示」を参照してください。

Performance Insights のイベントなど、AWS アカウントのイベントを継続的に記録する場合は、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべての AWS リージョンに証跡が適用されます。追跡では、AWS パーティション内のすべての AWS リージョンからのイベントをログに記録し、指定した Simple Storage Service (Amazon S3)バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づいて対応するた

め、他の AWS サービスを構成できます。詳細については、AWS CloudTrail ユーザーガイドの次のトピックを参照してください。

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての Performance Insights オペレーションは CloudTrail によってログに記録されます。また、Performance Insights API リファレンスに記載されています。例えば、`DescribeDimensionKeys` オペレーションと `GetResourceMetrics` オペレーションへのコールに伴って、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「[CloudTrail userIdentity 要素」を参照してください。

Performance Insights のログファイルのエントリ

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように設定できます。CloudTrail ログファイルには、1 つ以上のログエントリがあります。イベントは、任意の送信元からの単一のリクエストを表します。各イベントには、リクエストされたオペレーション、オペレーションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`GetResourceMetrics` オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Performance Insights API とインターフェイス VPC エンドポイント (AWS PrivateLink)

AWS PrivateLink を使用して、VPC と Amazon RDS Performance Insights 間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのように Performance Insights にアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても Performance Insights にアクセスできます。

このプライベート接続を確立するには、AWS PrivateLink を利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、Performance Insights 宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については「AWS PrivateLink ガイド」の「Access AWS のサービス through AWS PrivateLink」を参照してください。

Performance Insights に関する考慮事項

Performance Insights のインターフェイスエンドポイントを設定する前に、「AWS PrivateLink ガイド」の「考慮事項」を確認してください。

Performance Insights は、インターフェイスエンドポイントを介したすべての API アクションの呼び出しをサポートしています。

デフォルトでは、Performance Insights への完全なアクセスは、インターフェイスエンドポイント経由で許可されます。インターフェイスエンドポイントを介した Performance Insights へのトラフィックを制御するには、セキュリティグループをエンドポイントネットワークインターフェイスに関連付けます。

可用性

Performance Insights API は現在、Performance Insights をサポートする AWS リージョン の VPC エンドポイントをサポートしています。Performance Insights の可用性の詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン 」を参照してください。

Performance Insights のインターフェイスエンドポイントの作成

Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して、Performance Insights のインターフェイスエンドポイントを作成できます。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

以下のサービス名を使用して、Performance Insights のインターフェイスエンドポイントを作成します。

インターフェイスエンドポイントのプライベート DNS を有効にすると、デフォルトのリージョン DNS 名を使用して Performance Insights への API 要求を行うことができます。例えば、`pi.us-east-1.amazonaws.com`。

Performance Insights API の VPC エンドポイントポリシーの作成

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイントを介した Performance Insights へのフルアクセスが許可されています。VPC から Performance Insights に許可されるアクセスを制御するには、カスタムエンドポイントポリシーをインターフェイスエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドのControl access to services using endpoint policies (エンドポイントポリシーを使用してサービスへのアクセスをコントロールする)を参照してください。

例: Performance Insights アクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。このポリシーをインターフェイスエンドポイントにアタッチすると、すべてのリソースのすべてのプリンシパルについて、リストされている Performance Insights アクションへのアクセス権を付与します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "pi:CreatePerformanceAnalysisReport",
            "pi:DeletePerformanceAnalysisReport",
            "pi:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

例: 指定した AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

以下の VPC エンドポイントポリシーは、AWS アカウント 123456789012 からリソースへのエンドポイントを使用したすべてのアクセスを拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Performance Insights の IP アドレス指定

IP アドレスは、VPC 内のリソースの相互通信と、インターネット経由でのリソースとの通信を可能にします。Performance Insights は、IPv4 と IPv6 の両方のアドレス指定プロトコルをサポートしています。デフォルトでは、Performance Insights と Amazon VPC は IPv4 アドレス設定プロトコルを使用します。この動作をオフにすることはできません。VPC の作成時には、IPv4 CIDR ブロック (プライベート IPv4 アドレスの範囲) を指定する必要があります。

オプションで、IPv6 CIDR ブロックを VPC とサブネットに割り当て、そのブロックからサブネットの RDS リソースに IPv6 アドレスを割り当てることができます。IPv6 プロトコルのサポートにより、サポートされる IP アドレスの数が増えます。IPv6 プロトコルを使用することで、インターネットの今後の成長に十分なアドレスを確保できます。新規および既存の RDS リソースは、VPC 内で IPv4 アドレスと IPv6 アドレスを使用できます。アプリケーションの異なる部分で使用される 2 つのプロトコル間のネットワークトラフィックの設定、保護、および変換を行うと、運用上のオーバーヘッドが発生する可能性があります。Amazon RDS リソースについては IPv6 プロトコルを標準化して、ネットワーク構成を簡素化できます。サービスエンドポイントとクォータの詳細については、「Amazon リレーショナルデータベースサービスエンドポイントとクォータ」を参照してください。

Amazon RDS IP アドレス指定の詳細については、「Amazon RDS IP アドレス指定」を参照してください。

Amazon DevOps Guru for Amazon RDS でパフォーマンスの異常を分析する

Amazon DevOps Guru は、開発者およびオペレーターがアプリケーションのパフォーマンスと可用性を向上させるためのフルマネージド型オペレーションサービスです。DevOps Guru は、運用上の問題の特定に関連するタスクをオフロードし、アプリケーションを改善するための推奨事項を迅速に実装できるようにします。詳細については、「Amazon DevOps Guru ユーザーガイド」の「Amazon DevOps Guru とは」を参照してください。

DevOps Guru は、すべての Amazon RDS DB エンジンの既存の運用上の問題を検出し、分析し、推奨を行います。DevOps Guru for RDS は、 RDS for PostgreSQL データベースの Performance Insights メトリクスに機械学習を適用することで、この機能を拡張します。これらのモニタリング機能により、DevOps Guru for RDS はパフォーマンスのボトルネックを検出して診断し、具体的な修正措置を推奨できます。DevOps Guru for RDS は、 RDS for PostgreSQL データベースで問題が発生する前に問題条件を検出することもできます。

これらの推奨事項を RDS コンソールで表示できるようになりました。詳細については、「Amazon RDS の推奨事項」を参照してください。

次の動画は DevOps Guru for RDS の概要です。

この主題の詳細については、Amazon DevOps Guru for RDS の内部を参照してください。

トピック

- DevOps Guru for RDS の利点
- DevOps Guru for RDSはどのように機能しますか
- RDS 用の DevOps Guru のセットアップ

DevOps Guru for RDS の利点

RDS for PostgreSQL データベースを担当していて、そのデータベースに影響を与えるイベントやリグレッションの発生を知らないことがあります。問題を知っても、なぜそれが発生しているのか、どう対処すべきかわからないこともあります。データベース管理者 (DBA) に問い合わせたり、サードパーティーツールに頼ったりするのではなく、DevOps Guru for RDS のレコメンデーションに従ってください。

DevOps Guru for RDS の詳細な分析により、次の利点が得られます。

高速診断

DevOps Guru for RDS は、データベースのテレメトリを継続的にモニタリングおよび分析します。Performance Insights、拡張モニタリング、および Amazon CloudWatch は、データベースインスタンスのテレメトリーデータを収集します。DevOps Guru for RDS は、統計的な機械学習の技術を使用してこのデータをマイニングし、異常を検出します。テレメトリーデータの詳細については、、Amazon RDS ユーザーガイドの「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」および「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

高速解像度

各異常はパフォーマンスの問題を特定し、調査または修正措置の方法を提案します。例えば、DevOps Guru for RDS では、特定の待機イベントの調査をお勧めすることがあります。または、データベース接続数を制限するよう、アプリケーションプールの設定のチューニングをお勧めすることもあります。これらのレコメンデーションに基づいて、マニュアルでトラブルシューティングを実行するよりも迅速にパフォーマンスの問題を解決できます。

事前対応型インサイト

DevOps Guru for RDS は、リソースからのメトリクスを使用して、問題となる可能性のある動作を大きな問題になる前に検出します。例えば、データベースが使用するディスク上の一時テーブルの数が増え、パフォーマンスに影響を与え始めたことを検出できます。その場合、DevOps Guru は問題が大きくなる前に対処するのに役立つ推奨事項を提供します。

Amazon エンジニアの深い知識と機械学習

パフォーマンス問題を検出し、ボトルネックの解決を支援するために、DevOps Guru for RDS は機械学習 (ML) と高度な数式に依存しています。Amazon データベースエンジニアは、数十万のデータベース管理の長年の経験をカプセル化した DevOps Guru for RDS の知見の開発に貢献しました。この集合的な知識を活かすことで、DevOps Guru for RDS はベストプラクティスを伝えることができます。

DevOps Guru for RDSはどのように機能しますか

DevOps Guru for RDS は、Amazon RDS Performance Insights から RDS for PostgreSQL データベースに関するデータを収集します。最も重要なメトリクスはDBLoadです。DevOps Guru for RDS は、Performance Insights メトリクスを消費し、機械学習を使用して分析し、ダッシュボードにインサイトを公開します。

インサイトは、DevOps Guru によって検出された関連する異常のコレクションです。

DevOps Guru for RDS では、異常とは、 RDS for PostgreSQL データベースの通常のパフォーマンスから逸脱したパターンのことです。

事前対応型インサイト

プロアクティブインサイトでは、問題のある動作を発生前に知ることができます。異常と共に推奨事項と関連メトリクスも含まれるため、 RDS for PostgreSQL データベースの問題が大きな問題になる前に対処できます。これらのインサイトは、DevOps Guru ダッシュボードに発行されます。

例えば、DevOps Guru は、RDS for PostgreSQL データベースがディスク上に多数の一時テーブルを作成していることを検出する場合があります。対処しなければ、この傾向はパフォーマンス問題につながる可能性があります。各プロアクティブインサイトには、是正措置に関する推奨事項と、Amazon DevOps Guru のプロアクティブインサイトによる RDS for PostgreSQL のチューニング 内の関連トピックへのリンクが含まれています。詳細については、「Amazon DevOps Guru ユーザーガイド」の「Working with insights in DevOps Guru」を参照してください。

事後対応型インサイト

リアクティブインサイトは、異常な動作を発生時に識別します。DevOps Guru for RDS が RDS for PostgreSQL DB インスタンスでパフォーマンス問題を発見すると、DevOps Guru ダッシュボードにリアクティブインサイトが発行されます。詳細については、「Amazon DevOps Guru ユーザーガイド」の「Working with insights in DevOps Guru」を参照してください。

因果異常

因果異常は、リアクティブインサイト内のトップレベルの異常です。データベースロード (DB ロード)DevOps Guru for RDS の因果異常です。

異常は、深刻度レベルを高、中、低のいずれかに割り当てて、パフォーマンスへの影響を測定します。詳細については、「Amazon DevOps Guru ユーザーガイド」の「DevOps Guru for RDS の主要な概念」を参照してください。

DevOps Guru が DB インスタンスで現在の異常を検出すると、RDS コンソールの[Databases] (データベース) ページにアラートが表示されます。コンソールは、過去 24 時間に発生した異常についても警告します。RDS コンソールから異常ページに移動するには、アラートメッセージ内のリンクを選択します。RDS コンソールでは、 RDS for PostgreSQL DB インスタンスのページでもアラートが表示されます。

コンテキスト異常

コンテキスト異常は、事後対応型インサイトに関連するデータベースロード (DB ロード)での所見です。各コンテキスト異常は、調査が必要な特定の RDS for PostgreSQL のパフォーマンス問題を記述します。例えば、DevOps Guru for RDS は、CPU 容量の増加を検討したり、DB ロードに寄与している待機イベントを調査するよう推奨することがあります。

> **Important**
>
> 本稼働インスタンスの修正前に、各変更の影響を完全に把握できるように、テストインスタンスでの変更のテストをお勧めします。このようにして、変更の影響を理解します。

詳細については、Amazon DevOps Guru ユーザーガイドの「Analyzing anomalies in Amazon RDS」(Amazon RDS の異常を分析する) を参照してください。

RDS 用の DevOps Guru のセットアップ

DevOps Guru for Amazon RDS が RDS for PostgreSQL データベースのインサイトを発行できるようにするには、以下のタスクを実行します。

トピック

- DevOps Guru for RDS の IAM アクセスポリシーの設定
- RDS for PostgreSQL DB インスタンスの Performance Insights をオンにする
- DevOps Guru をオンにしてリソースカバレッジを指定する

DevOps Guru for RDS の IAM アクセスポリシーの設定

DevOps Guru からのアラートを RDS コンソールに表示するには、AWS Identity and Access Management (IAM) ユーザーまたはロールに次のいずれかのポリシーが必要です。

- AWS マネージドポリシー `AmazonDevOpsGuruConsoleFullAccess`
- AWS 管理ポリシー `AmazonDevOpsGuruConsoleReadOnlyAccess` および次のいずれかのポリシーが必要です。
  - AWS マネージドポリシー `AmazonRDSFullAccess`
  - `pi:GetResourceMetrics` と `pi:DescribeDimensionKeys` を含むカスタマー管理ポリシー

詳細については、「Performance Insights 用のアクセスポリシーの設定」を参照してください。

RDS for PostgreSQL DB インスタンスの Performance Insights をオンにする

DevOps Guru for RDS は、データの Performance Insights に依存しています。Performance Insights がなければ、DevOps Guru は異常を公開しますが、詳細な分析と推奨事項は含まれません。

RDS for PostgreSQL DB インスタンスを作成または変更するときに、Performance Insights をオンにすることができます。詳細については、「Amazon RDS の Performance Insights の有効化と無効化」を参照してください。

DevOps Guru をオンにしてリソースカバレッジを指定する

DevOps Guru をオンにして、次のいずれかの方法で RDS for PostgreSQL データベースをモニタリングできます。

トピック

- RDS コンソールで DevOps Guru をオンにする
- RDS for PostgreSQL リソースを DevOps Guru コンソールに追加する
- CloudFormation を使用して RDS for PostgreSQL リソースを追加する

RDS コンソールで DevOps Guru をオンにする

Amazon RDS コンソールで複数のパスを取得して DevOps Guru をオンにすることができます。

トピック

- RDS for PostgreSQL データベースを作成するときに DevOps Guru をオンにする
- 通知バナーから DevOps Guru をオンにする
- DevOps Guru をオンにしたときのアクセス許可エラーへの応答

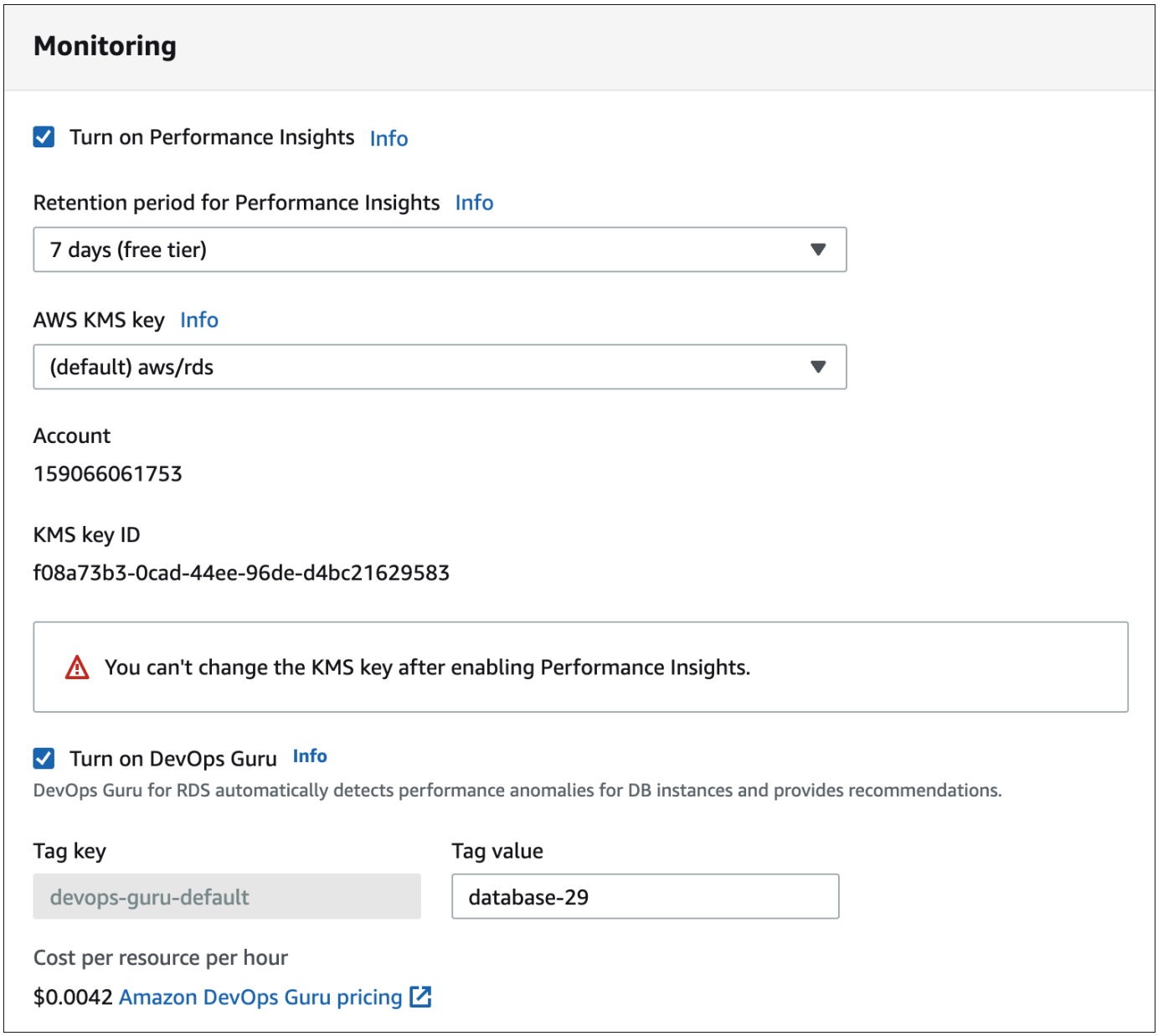
RDS for PostgreSQL データベースを作成するときに DevOps Guru をオンにする

作成ワークフローには、データベースの DevOps Guru カバレッジを有効にする設定が含まれています。本番稼働用テンプレートを選択した場合、この設定はデフォルトでオンになっています。

RDS for PostgreSQL データベースを作成するときに DevOps Guru をオンにするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. 「DB インスタンスの作成」で、モニタリング設定を選択する手順まで (ただしその手順は含まない) の手順を行います。
3. [Monitoring] (モニタリング) で、[Turn on Performance Insights] (Performance Insights をオンにする) を選択します。DevOps Guru for RDS で、パフォーマンスの異常の詳細な分析を提供するには、[Performance Insights] (パフォーマンスインサイト) をオンにする必要があります。
4. [Turn on DevOps Guru] (DevOps Guru をオンにする) を選択します。



5. DevOps Guru がそれをモニタリングできるように、データベースのタグを作成します。以下の操作を実行します。
   - [Tag key] (タグキー) のテキストフィールドで、**`Devops-Guru-`** で始まる名前を入力します。

- [Tag value] (タグ値) のテキストフィールドで、任意の値を入力します。例えば、 RDS for PostgreSQL データベースの名前として **rds-database-1** と入力した場合、タグ値として **rds-database-1** も入力することができます。

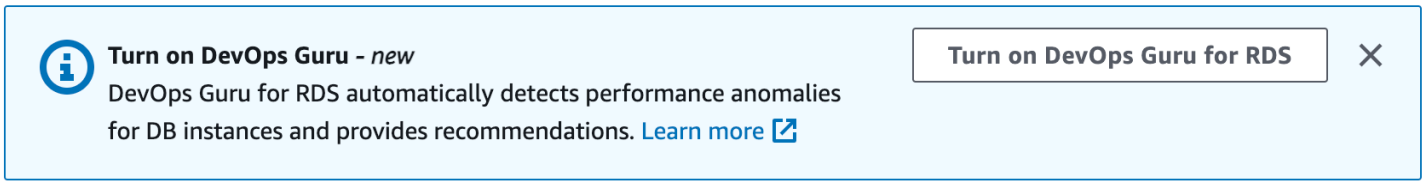
タグの詳細については、「Amazon DevOps Guru ユーザーガイド」の「タグを使用して DevOps Guru アプリケーションのリソースを特定する」を参照してください。

6. 「DB インスタンスの作成」の残りのステップを行います。

通知バナーから DevOps Guru をオンにする

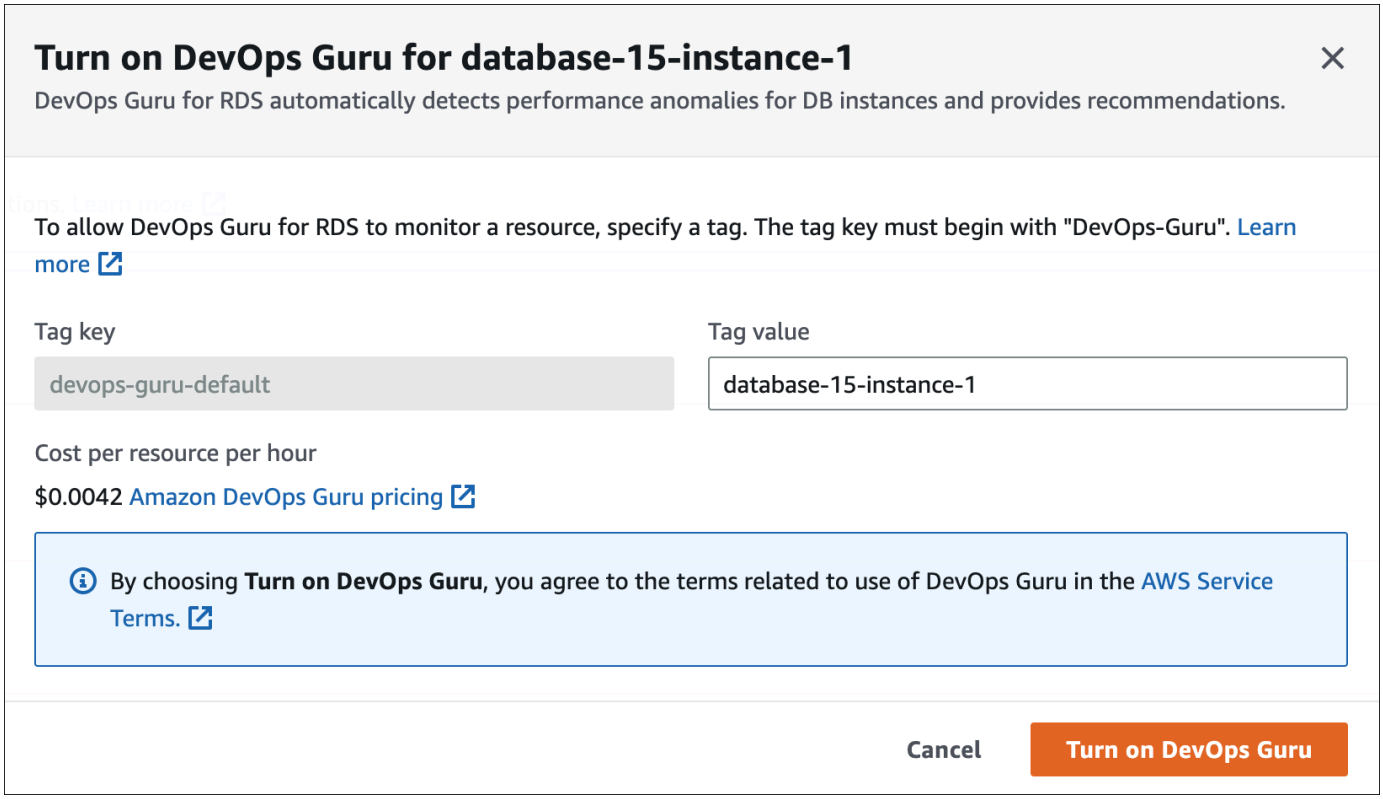
リソースが DevOps Guru の対象外である場合、Amazon RDS は次の場所のバナーで通知します。

- DB クラスターインスタンスの[Monitoring] (モニタリング) タブ
- Performance Insights ダッシュボード



RDS for PostgreSQL データベースについて DevOps Guru をオンにするには

1. バナーで、[Turn on DevOps Guru for RDS] (DevOps Guru for RDS をオンにする) を選択します。
2. タグのキー名と値を入力します。タグの詳細については、「Amazon DevOps Guru ユーザーガイド」の「タグを使用して DevOps Guru アプリケーションのリソースを特定する」を参照してください。



3. [DevOps Guru をオンにする] を選択します。

DevOps Guru をオンにしたときのアクセス許可エラーへの応答

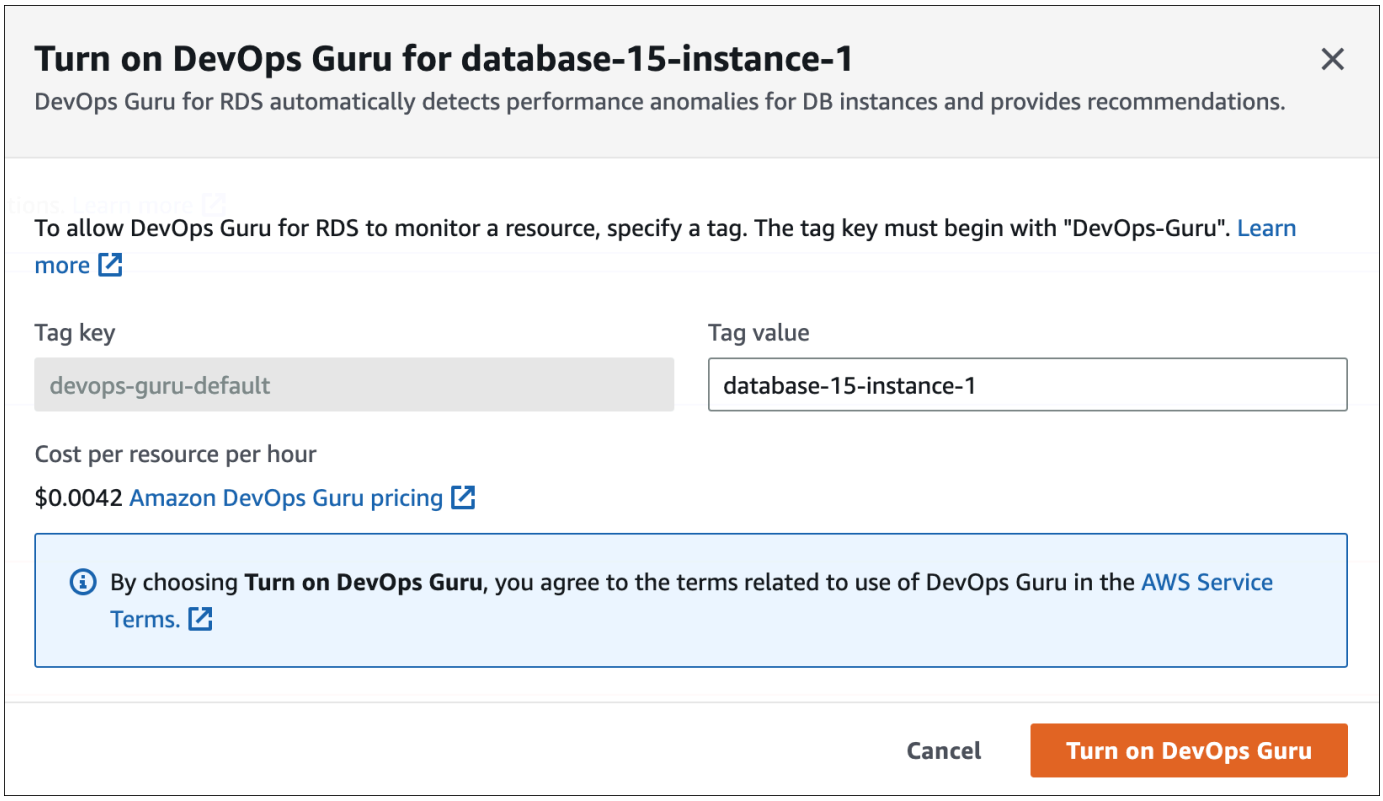
データベースを作成するときに RDS コンソールから DevOps Guru をオンにすると、RDS にアクセス許可がないことについて次のバナーが表示されることがあります。

アクセス許可エラーに応答するには

1. IAM ユーザーまたはロールにユーザー管理ロール `AmazonDevOpsGuruConsoleFullAccess` を付与します。詳細については、「DevOps Guru for RDS の IAM アクセスポリシーの設定」を参照してください。
2. RDS コンソールを開きます。
3. ナビゲーションペインで、[Performance Insights] を選択します。
4. 先ほど作成したクラスターで DB インスタンスを選択します。
5. スイッチを選択して、[DevOps Guru for RDS] をオンにします。

DevOps Guru for RDS

6. タグ値を選択します。詳細については、「Amazon DevOps Guru ユーザーガイド」の「タグを使用して DevOps Guru アプリケーションのリソースを特定する」を参照してください。



7. [DevOps Guru をオンにする] を選択します。

RDS for PostgreSQL リソースを DevOps Guru コンソールに追加する

DevOps Guru コンソールで DevOps Guru リソースカバレッジを指定できます。「Amazon DevOps Guru ユーザーガイド」の「DevOps Guru リソースカバレッジを指定する」で説明されている手順に従います。分析リソースを編集する場合は、以下のオプションのいずれかを選択します。

- RDS for PostgreSQL データベースも含め、AWS アカウントとリージョンでサポートされているすべてのリソースを分析するには、[すべてのアカウントリソース] を選択します。
- 選択したスタック内の RDS for PostgreSQL データベースを分析するには、[CloudFormation スタック] を選択します。詳細については、「Amazon DevOps Guru ユーザーガイド」の「AWS CloudFormation スタックを使用して DevOps Guru アプリケーション内のリソースを識別する」を参照してください。

- タグ付けした RDS for PostgreSQL データベースを分析するには、[タグ] を選択します。詳細については、「Amazon DevOps Guru ユーザーガイド」の「タグを使用して DevOps Guru アプリケーションのリソースを特定する」を参照してください。

詳細については、「Amazon DevOps Guru ユーザーガイド」の 「Amazon DevOps Guru の有効化」を参照してください。

CloudFormation を使用して RDS for PostgreSQL リソースを追加する

タグを使用して、 RDS for PostgreSQL リソースのカバレッジを CloudFormation テンプレートに追加できます。次の手順では、 RDS for PostgreSQL DB インスタンスと DevOps Guru スタックの両方についての CloudFormation テンプレートがあることを前提としています。

CloudFormation タグを使用して RDS for PostgreSQL DB インスタンスを指定するには

1. DB インスタンスの CloudFormation テンプレートで、キーと値のペアを使用してタグを定義します。

   次の例では、値 `my-db-instance1` を RDS for PostgreSQL DB インスタンスの `Devops-guru-cfn-default` に割り当てます。

   ```
   MyDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBInstanceIdentifier: my-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-db-instance1
   ```

2. DevOps Guru スタックの CloudFormation テンプレートで、リソースコレクションフィルターに同じタグを指定します。

   次の例では、タグ値 `my-db-instance1` を持つリソースをカバーするように DevOps Guru を設定します。

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
           - AppBoundaryKey: "Devops-guru-cfn-default"
   ```

```
          TagValues:
          - "devopsguru-my-db-instance1"
```

次の例では、アプリケーション境界 Devops-guru-cfn-default 内のすべてのリソースを対象としています。

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

詳細については、「CloudFormation ユーザーガイド」の「AWS::DevOpsGuru::ResourceCollection」と「AWS::RDS::DBInstance」を参照してください。

拡張モニタリングを使用した OS メトリクスのモニタリング

Enhanced Monitoring を使用すると、DB インスタンスのオペレーティングシステムをリアルタイムでモニタリングできます。さまざまなプロセスまたはスレッドが CPU をどのように使用しているかを確認するには、Enhanced Monitoring メトリクスが役立ちます。

トピック

- Enhanced Monitoring の概要
- 拡張モニタリングの設定と有効化
- RDS コンソールでの OS メトリクスの表示
- CloudWatch Logs を使用した OS メトリクスの表示

Enhanced Monitoring の概要

Amazon RDS には、DB インスタンスが実行されているオペレーティングシステム (OS) のリアルタイムのメトリクスが用意されています。RDS DB インスタンスのすべてのシステムメトリクスとプロセス情報をコンソールに表示できます。各インスタンスでモニタリングするメトリクスを管理し、要件に応じてダッシュボードをカスタマイズできます。拡張モニタリングメトリクスの説明については、「拡張モニタリングの OS メトリクス」を参照してください。

RDS は、拡張モニタリングのメトリクスを Amazon CloudWatch Logs アカウントに配信します。CloudWatch Logs から CloudWatch のメトリクスフィルタを作成し、CloudWatch ダッシュボードにグラフを表示できます。選択したモニタリングシステムで CloudWatch Logs からの拡張モニタリング JSON 出力を使用できます。詳細については、Amazon RDS に関するよくある質問の「拡張モニタリング」を参照してください。

トピック

- 拡張モニタリングの可用性
- CloudWatch と拡張モニタリングのメトリクスの相違点
- Enhanced Monitoring メトリクスの保持
- 拡張モニタリングのコスト

拡張モニタリングの可用性

拡張モニタリングは、次のデータベースエンジンに使用できます。

- Db2
- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

CloudWatch と拡張モニタリングのメトリクスの相違点

ハイパーバイザーは、仮想マシン (VM) を作成して実行します。ハイパーバイザーを使用すると、メモリと CPU を仮想的に共有することで、1 つのインスタンスで複数のゲスト VM をサポートできます。CloudWatch は DB インスタンスのハイパーバイザーから CPU 使用率のメトリクスを収集します。対照的に、Enhanced Monitoring は DB インスタンス上のエージェントからメトリクスを収集します。

ハイパーバイザーレイヤーで少量の処理が実行されるため、CloudWatch と Enhanced Monitoring 測定値の間に違いが見つかることがあります。DB インスタンスがより小さなインスタンスクラスを使用している場合、その違いはより大きくなる可能性があります。このシナリオでは、1 つの物理インスタンス上のハイパーバイザー層によってより多くの仮想マシン (VM) が管理されている可能性があります。

拡張モニタリングメトリクスの説明については、「拡張モニタリングの OS メトリクス」を参照してください。Amazon CloudWatch メトリクスの詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

Enhanced Monitoring メトリクスの保持

デフォルトでは、Enhanced Monitoring メトリクスは CloudWatch Logs で 30 日間保存されます。この保持期間は、通常の CloudWatch メトリクスとは異なります。

メトリクスが CloudWatch Logs に保存される時間の長さを変更するには、CloudWatch コンソールの `RDSOSMetrics` ロググループの保存期間を変更します。詳細については、Amazon CloudWatch Logs User Guideの「CloudWatch ログでのログデータ保管期間の変更」を参照してください。

拡張モニタリングのコスト

拡張モニタリングのメトリクスは、CloudWatch メトリクスではなく CloudWatch Logs に保存されます。拡張モニタリングのコストは次の要因によって異なります。

- 拡張モニタリングの課金は、Amazon CloudWatch Logs に指定されているデータ転送とストレージの量を超えた場合にのみ発生します。CloudWatch Logs のデータ転送料金とストレージ料金に基づいて料金が決まります。
- RDS インスタンスに対して転送される情報の量は、拡張モニタリング機能に対して定義された詳細度に正比例します。モニタリング間隔を短くすると、OS メトリクスのレポート回数が増え、モニタリングコストが高くなります。コストを管理するには、アカウント内のインスタンスごとに異なる詳細度を設定します。
- 拡張モニタリングの使用コストは、拡張モニタリングが有効になっている各 DB インスタンスに適用されます。多数の DB インスタンスをモニタリングすると、少数の DB インスタンスをモニタリングするよりもコストが高くなります。
- 複数のコンピューティング集中型のワークロードをサポートする DB インスタンスでは、レポートする OS プロセスアクティビティが増え、拡張モニタリングのコストがより高くなります。

料金の詳細については、「Amazon CloudWatch の料金」を参照してください。

拡張モニタリングの設定と有効化

拡張モニタリングを使用するには、IAM ロールを作成し、拡張モニタリングを有効にする必要があります。

トピック

- 拡張モニタリング用の IAM ロールの作成
- 拡張モニタリングのオンとオフを切り替える
- 「混乱した代理」問題からの保護

拡張モニタリング用の IAM ロールの作成

拡張モニタリングには、CloudWatch Logs に OS メトリクスの情報を送るためのアクセス権限が必要です。AWS Identity and Access Management (IAM) ロールを使用して、Enhanced Monitoring に必要なアクセス許可を付与します。このロールは、拡張モニタリングを有効にする際に作成することも、事前に作成しておくこともできます。

トピック

- 拡張モニタリングを有効にしたときの IAM ロールの作成
- 拡張モニタリングを有効にする前の IAM ロールの作成

拡張モニタリングを有効にしたときの IAM ロールの作成

RDS コンソールで拡張モニタリングを有効にすると、Amazon RDS は必要な IAM ロールを作成できます。ロールの名前は `rds-monitoring-role` です。RDS は、指定済み DB インスタンス、リードレプリカ、またはマルチ AZ DB クラスターに対してこのロールを使用します。

拡張モニタリングを有効にするときに、IAM ロールを作成するには

1. 拡張モニタリングのオンとオフを切り替える の手順を行います。
2. ロールを選択する手順で、[モニタリングロール] を [デフォルト] に設定します。

拡張モニタリングを有効にする前の IAM ロールの作成

拡張モニタリングを有効にする前に、必要なロールを作成できます。拡張モニタリングを有効にする場合は、新しいロールの名前を指定します。AWS CLI または RDS API を使用して拡張モニタリングを有効にする場合は、この必要なロールを作成する必要があります。

拡張モニタリングを有効にするユーザーには、`PassRole` アクセス許可を付与する必要があります。詳細については、IAM ユーザーガイドの「AWS サービスにロールを渡すアクセス許可をユーザーに許可する」の「例 2」を参照してください。

Amazon RDS 拡張モニタリング用の IAM ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com) を開きます。
2. ナビゲーションペインで [ロール] を選択します。
3. [ロールの作成] を選択します。
4. [AWS のサービス] タブを選択し、サービスのリストから [RDS] を選択します。
5. [RDS - Enhanced Monitoring] (RDS - 拡張モニタリング)、[Next] (次へ) の順に選択します。
6. [Permissions policies] (アクセス許可ポリシー) に [AmazonRDSEnhancedMonitoringRole] が表示されていることを確認し、[Next] (次へ) を選択します。
7. [ロール名] に、ロールの名前を入力します。例えば、「**`emaccess`**」と入力します。

   ロールの信頼されたエンティティは、AWS サービスmonitoring.rds.amazonaws.com です。
8. [ロールの作成] を選択してください。

拡張モニタリングのオンとオフを切り替える

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して拡張モニタリングを管理できます。DB インスタンスごとに、メトリクス収集の詳細度を別々に設定できます。

コンソール

DB インスタンス、マルチ AZ DB クラスター、もしくはリードレプリカの作成時、または DB インスタンスもしくはマルチ AZ DB クラスターの変更時に、拡張モニタリングをオンにすることができます。拡張モニタリングをオンにするように DB インスタンスを変更した場合は、変更を有効にするために DB インスタンスを再起動する必要はありません。

RDS コンソールの [データベース] ページで、次のいずれかのアクションを行うと、拡張モニタリングをオンにすることができます。

- DB インスタンスまたはマルチ AZ DB クラスターを作成する – [データベースを作成] を選択します。
- [リードレプリカの作成] – [アクション]、[リードレプリカの作成] の順にクリックします。
- [DB インスタンスまたはマルチ AZ DB クラスターを変更する] – [変更] を選択します。

RDS コンソールで拡張モニタリングのオンとオフを切り替えるには

1. [その他の設定] までスクロールします。
2. [モニタリング] で、DB インスタンス、またはリードレプリカに対し、[拡張モニタリングを有効にする] をクリックします。
3. [ロールの作成] プロパティで、Amazon CloudWatch Logs との通信を Amazon RDS に許可するために作成した IAM ロールに設定するか、[デフォルト] を選択して、RDS によって `rds-monitoring-role` という名前でロールが作成されるようにします。

> **Note**
>
> アカウントでロールマネージャーが有効になっている場合、Amazon RDS は自動的にロールをアタッチします。また、ここで説明するロールオプションは [カスタマイズ] オプションに置き換えられます。別のロールを使用するには、[カスタマイズ] を選択します。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

4. [詳細度] プロパティを、DB インスタンス、またはリードレプリカのメトリクスが収集される間隔 (秒単位) に設定します。[詳細度] プロパティは、1、5、10、15、30、60 のいずれかの値に設定できます。

   RDS コンソールは最速で 5 秒ごとに更新されます。RDS コンソールで詳細度を 1 秒に設定しても、メトリクスはやはり 5 秒ごとに更新されます。1 秒ごとにメトリクスの更新を取得するには、CloudWatch Logs を使用します。

AWS CLI

AWS CLI を使用して拡張モニタリングをオンにするには、次のコマンドで `--monitoring-interval` オプションを `0` 以外の値に設定し、`--monitoring-role-arn` オプションを 拡張モニタリング用の IAM ロールの作成 で作成したロールに設定します。

- create-db-instance
- create-db-instance-read-replica
- modify-db-instance
- create-db-cluster (マルチ AZ DB クラスター)
- modify-db-cluster (マルチ AZ DB クラスター)

`--monitoring-interval` オプションにより、拡張モニタリングのメトリクスが収集される時間点の間隔 (秒単位) が指定されます。オプションの有効な値は、`0`、`1`、`5`、`10`、`15`、`30`、および `60` です。

AWS CLI を使用して拡張モニタリングをオフにするには、これらのコマンドで `--monitoring-interval` オプションを `0` に設定します。

Example

次の例では、DB インスタンスの拡張モニタリングをオンに切り替えています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

次の例では、マルチ AZ DB クラスターの拡張モニタリングをオンに切り替えています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

RDS API

RDS API を使用して拡張モニタリングをオンにするには、MonitoringInterval パラメータに 0 以外の値を設定し、MonitoringRoleArn パラメータを 拡張モニタリング用の IAM ロールの作成 で作成したロールに設定します。次のアクションでこれらのパラメータを設定します。

- CreateDBInstance
- CreateDBInstanceReadReplica
- ModifyDBInstance
- CreateDBCluster (マルチ AZ DB クラスター)
- ModifyDBCluster (マルチ AZ DB クラスター)

MonitoringInterval パラメータにより、拡張モニタリングのメトリクスが収集される時間点の間隔 (秒単位) が指定されます。有効な値は、0、1、5、10、15、30、60 です。

RDS API を使用して拡張モニタリングをオフにするには、MonitoringInterval に 0 を設定します。

「混乱した代理」問題からの保護

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。AWS では、サービス間でのなりすましによって、混乱した代理問題が発生する場合があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。詳細については、「「混乱した代理」問題」を参照してください。

リソースに関して、Amazon RDS から別のサービスに付与できるアクセス許可を制限する場合は、拡張モニタリングロール用の信頼ポリシー内で、aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用することをお勧めします。これら両方のグローバル条件コンテキストキーを使用する場合、アカウント ID は同じものを使用する必要があります。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して aws:SourceArn グローバル条件コンテキストキーを使用することです。Amazon RDS の場合は、aws:SourceArn に arn:aws:rds:*Region*:*my-account-id*:db:*dbname* を設定します。

次の例では、「混乱した代理」問題を防ぐために、信頼ポリシー内で aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストを使用しています。

JSON

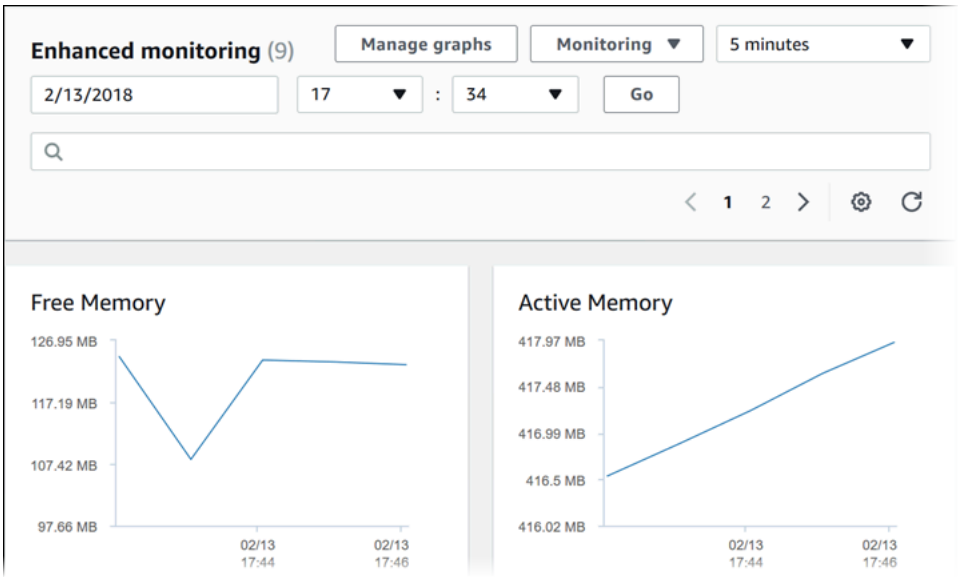
```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-123456789012:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-123456789012"
        }
      }
    }
  ]
}
```

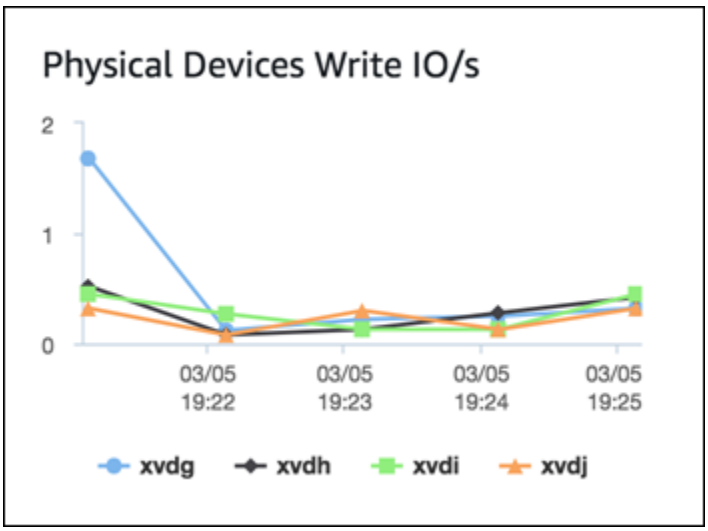
RDS コンソールでの OS メトリクスの表示

拡張モニタリングによってレポートされた OS のメトリクスを RDS コンソールで表示するには、[モニタリング] の [拡張モニタリング] を選択します。

次の例は、拡張モニタリングのページを示しています。拡張モニタリングメトリクスの説明については、「拡張モニタリングの OS メトリクス」を参照してください。



一部の DB インスタンスでは、DB インスタンスのデータストレージボリュームに複数のディスクを使用します。このような DB インスタンスの [物理デバイス] グラフには、それぞれのディスクのメトリクスが表示されます。例えば、次のグラフには、4 つのディスクのメトリクスが表示されています。

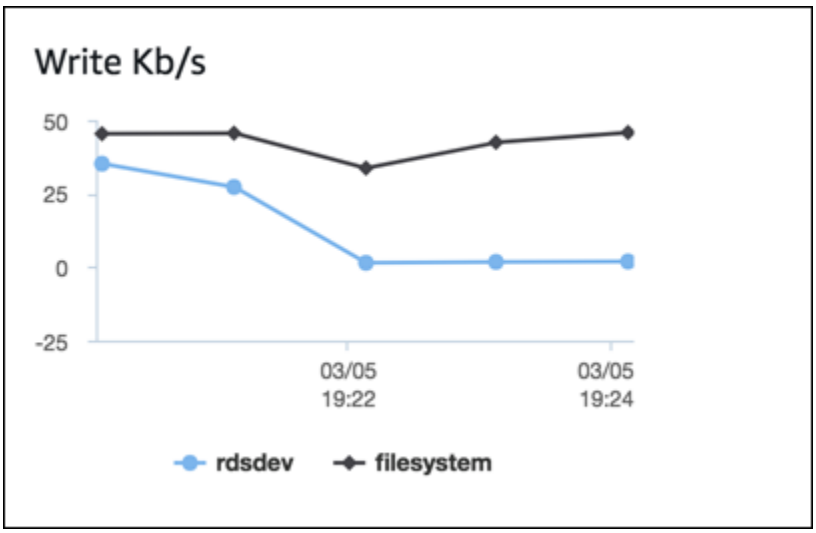


> **Note**
>
> 現在、[物理デバイス] グラフは、Microsoft SQL Server DB インスタンスでは使用できません。

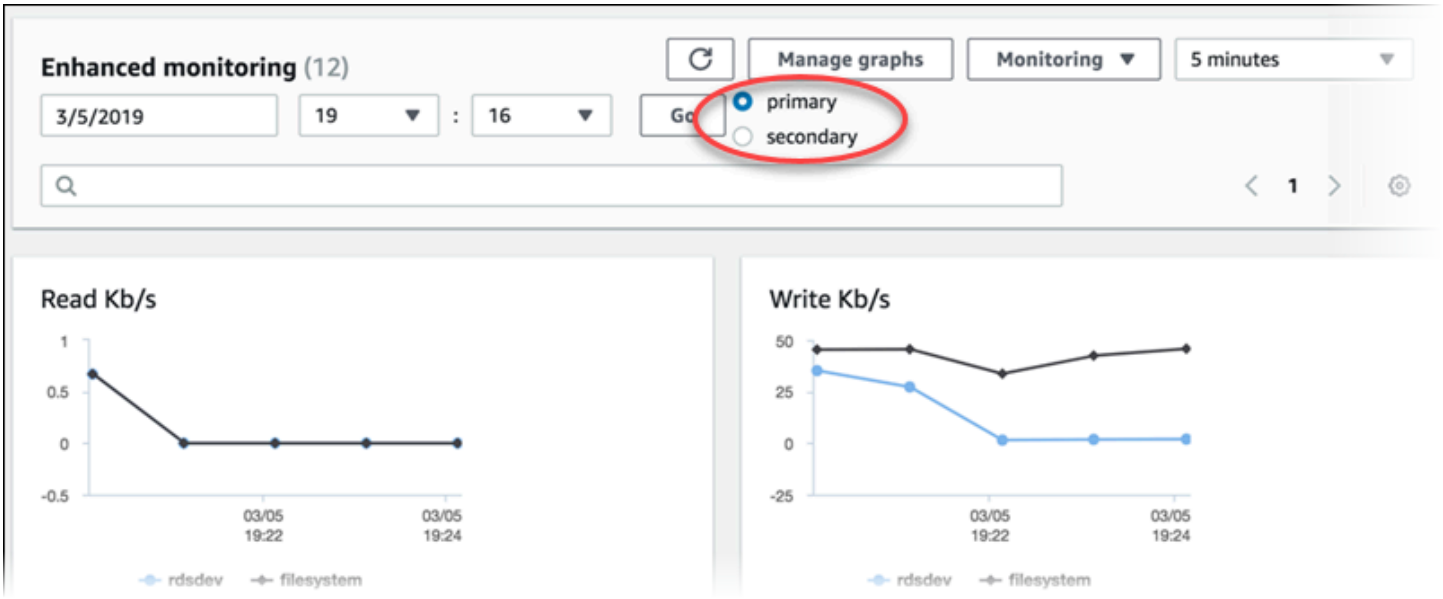
集約された [ファイルシステム] に関するグラフを表示する場合、(すべてのデータベースファイルとログが保存されている) `rdsfilesys/rdsdbdata*` ファイルシステムに [rdsdbdata*] デバイスが関連付けられます。[rootfilesys] デバイスは / ファイルシステム (root と呼ばれる) に関連付けられています。ここで、オペレーティングシステムに関するファイルは保存されます。追加のストレージボリュームを使用する場合は、ボリューム固有の情報の `rdsdbdata2`、`rdsdbdata3`、および `rdsdbdata4` ボリュームメトリクスを表示します。

集約された [ディスク I/O] グラフを表示している場合、[rdsdbdata] デバイスはプライマリ / `rdsdbdata` ストレージボリュームに関連付けられます。追加のストレージボリュームを使用する場合は、ボリューム固有の情報の `rdsdbdata2`、`rdsdbdata3`、および `rdsdbdata4` ボリュームメトリクスを表示します。[filesystem] デバイスは /ファイルシステム (root と呼ばれる) に関連付けられています。ここで、オペレーティングシステムに関するファイルは保存されます。

デバイス名 [rdsdev] は廃止されました。[rdsdev] デバイスはプライマリ `/rdsdbdata` ストレージボリュームにのみ関連し、追加のストレージボリュームからのメトリクスは含まれません。



DB インスタンスがマルチ AZ 配置の場合は、プライマリ DB インスタンスとそのマルチ AZ スタンバイレプリカの OS メトリクスを表示できます。[拡張モニタリング] ビューで、[プライマリ] を選択してプライマリ DB インスタンスの OS メトリクスを表示するか、[セカンダリ] を選択してスタンバイレプリカの OS メトリクスを表示します。

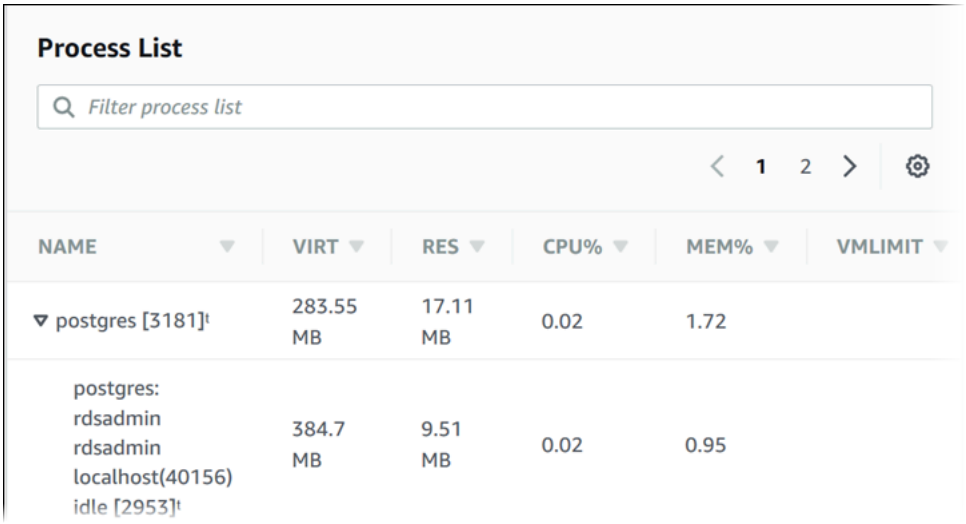


マルチ AZ 配置については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

> **Note**
>
> 現在、マルチ AZ スタンバイレプリカの OS メトリクスを表示することは、MariaDB インスタンスではサポートされていません。

DB インスタンスで実行中のプロセスの詳細を確認する場合は、[モニタリング] の [OS プロセスリスト] を選択します。

[プロセスリスト] ビューは次のように表示されます。



[プロセスリスト] ビューに表示される拡張モニタリングのメトリクスは以下のように整理されます。

- [RDS child processes] – DB インスタンスをサポートする RDS プロセス (Amazon Aurora DB クラスターの場合は `aurora`、MySQL DB インスタンスの場合は `mysqld` など) の概要を表示します。プロセスのスレッドは親プロセスの下にネストされて表示されます。プロセスのスレッドには CPU 使用率のみが表示されます。他のメトリクスはプロセスのすべてのスレッドで同じであるためです。コンソールには最大 100 個のプロセスとスレッドが表示されます。結果は、プロセスとスレッドを消費している上位の CPU とメモリの組み合わせです。プロセスとスレッドが 50 個よりも多い場合、コンソールではカテゴリ別に上位 50 個の消費元が表示されます。この表示は、パフォーマンスに最大の影響を与えているプロセスを特定するために役立ちます。
- [RDS プロセス] – RDS DB インスタンスをサポートするために必要な RDS 管理エージェント、診断モニタリングプロセス、その他の AWS プロセスによって使用されているリソースの概要を表示します。
- [OS processes] – 一般的にパフォーマンスに最小の影響を与えているカーネルとシステムプロセスの概要を表示します。

各プロセスに対して表示される項目は次のとおりです。

- VIRT – プロセスの仮想サイズを表示します。
- RES – プロセスが使用する実際の物理メモリを表示します。
- [CPU%] – プロセスで使用されている合計 CPU 帯域幅のパーセンテージを表示します。

- [MEM%] – プロセスで使用されている合計メモリのパーセンテージを表示します。

RDS コンソールに表示するモニタリングデータは、Amazon CloudWatch Logs から取得されます。また、DB インスタンスのメトリクスも CloudWatch Logs からログストリームとして取得できます。詳細については、「CloudWatch Logs を使用した OS メトリクスの表示」を参照してください。

以下の実行中は拡張モニタリングメトリクスは返されません:

- DB インスタンスのフェイルオーバー。
- DB インスタンスのインスタンスクラスの変更 (コンピューティングのスケール)。

拡張モニタリングのメトリクスは DB インスタンスの再起動中も返されます。これはデータベースエンジンのみが再起動するためです。オペレーティングシステムのメトリクスは、引き続き報告されます。

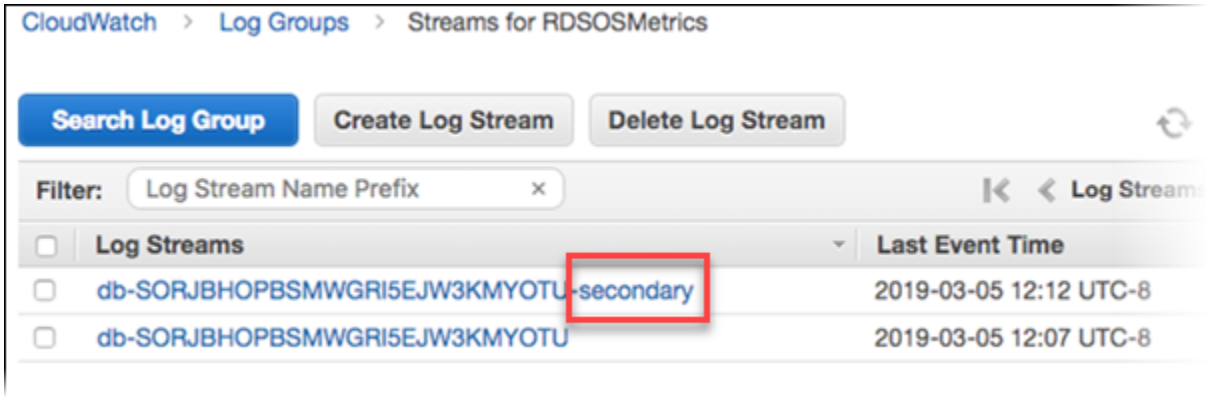
CloudWatch Logs を使用した OS メトリクスの表示

DB インスタンスまたはマルチ AZ DB クラスターの拡張モニタリングを有効にした後、CloudWatch Logs を使用してそのメトリクスを表示できます。各ログストリームは、モニタリング中の 1 つの DB インスタンスまたは DB クラスターを表します。ログストリーム識別子は DB インスタンスまたは DB クラスターのリソース識別子 (`DbiResourceId`) です。

拡張モニタリングのログデータを表示するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 必要に応じて、DB インスタンスまたはマルチ AZ DB クラスターが存在する AWS リージョンを選択します。詳細については、Amazon Web Services 全般のリファレンスの「リージョンとエンドポイント」を参照してください。
3. ナビゲーションペインで [ログ] を選択します。
4. ロググループのリストから [RDSOSMetrics] を選択します。

   マルチ AZ DB インスタンス配置では、名前に `-secondary` が追加されたログファイルは、マルチ AZ スタンバイレプリカ用です。



5. ログストリームのリストから、表示するログストリームを選択します。

Amazon RDS のメトリクスリファレンス

このリファレンスでは、Amazon CloudWatch、Performance Insights、および Enhanced Monitoring に関する Amazon RDS メトリクスが説明されています。

トピック

- Amazon RDS の Amazon CloudWatch メトリクス
- Amazon RDS の Amazon CloudWatch ディメンション
- Amazon RDS Performance Insights の Amazon CloudWatch メトリクス
- Performance Insights カウンターメトリクス
- Performance Insights の SQL 統計
- 拡張モニタリングの OS メトリクス

Amazon RDS の Amazon CloudWatch メトリクス

Amazon CloudWatch メトリクスは、Amazon RDS インスタンスとクラスターのパフォーマンスとヘルスに関するインサイトを提供し、システム動作をモニタリングしてデータ主導の意思決定を行うことができます。これらのメトリクスは、リソースの使用率、データベースアクティビティ、運用効率を追跡するのに役立ち、インスタンスのパフォーマンスを可視化します。

このリファレンスでは、Amazon RDS で使用できる特定のメトリクスの概要と、データベースのパフォーマンスの最適化、問題のトラブルシューティング、高可用性の確保のためにそれらのメトリクスを解釈して使用する方法について説明します。

Amazon RDS は、`AWS/RDS` および `AWS/Usage` 名前空間で、Amazon CloudWatch にメトリクスを発行します。

トピック

- Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス
- Amazon RDS の Amazon CloudWatch 使用状況メトリクス

Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス

Amazon CloudWatch の `AWS/RDS` 名前空間には、次のインスタンスレベルのメトリクスが含まれます。

> **Note**
>
> Amazon RDS コンソールには、Amazon CloudWatch に送信された単位とは異なる単位でメトリクスが表示される場合があります。例えば、Amazon RDS コンソールにはメトリクスがメガバイト (MB) で表示されますが、Amazon CloudWatch にはメトリクスはバイト単位で送信されます。

メトリクス	説明	Applies to	単位
`BinLogDiskUsage`	バイナリログが占めるディスク容量。リードレプリカを含む MySQL および MariaDB インスタンスで自動バックアップが有効になっている場合、バイナリログが作成されます。	MariaDB MySQL	バイト
`BurstBalance`	汎用 SSD (gp2) のバーストバケット I/O クレジットの利用可能パーセント。	すべて	割合 (%)
`CheckpointLag`	直近のチェックポイントからの時間。		[秒]
`ConnectionAttempts`	成功したかどうかを問わず、インスタンスへの接続を試行した回数。	MySQL	カウント
`CPUUtilization`	CPU 使用率。	すべて	割合 (%)
`CPUCreditUsage`	CPU 使用率に関してインスタンスで消費される CPU クレジットの数。1 CPUクレジットは、1 分間 100% の使用率で実行される 1 つの vCPU、または vCPU、使用率、時間の同等の組み合わせに相当します。例えば、2 分間 50% の使用率で実行されている 1 つの vCPU、または 2 分間 25% の使用率で実行されている 2 つの vCPU があるとします。		クレジット (vCPU 分)

メトリクス	説明	Applies to	単位
	このメトリクスは、db.t2、db.t3、および db.t4g インスタンスにのみ適用されます。 **Note** T DB インスタンスクラスは、開発サーバーおよびテストサーバー、またはその他の本稼働以外のサーバーにのみ使用することをお勧めします。T インスタンスクラスの詳細については、「DB インスタンスクラスタイプ」を参照してください。 CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。5 分を超える期間を指定する場合は、Average 統計の代わりに Sum 統計を使用します。		

メトリクス	説明	Applies to	単位
CPUCredit Balance	インスタンスが起動または開始後に蓄積した獲得 CPU クレジットの数。T2 スタンダードの場合、CPUCredit Balance には蓄積された起動クレジットの数も含まれます。 クレジットは獲得後にクレジット残高に蓄積され、消費されるとクレジット残高から削除されます。クレジット残高にはインスタンスサイズによって決まる上限があります。制限に到達すると、獲得された新しいクレジットはすべて破棄されます。T2 スタンダードの場合、起動クレジットは制限に対してカウントされません。 CPUCreditBalance のクレジットは、インスタンスがそのベースライン CPU 使用率を超えてバーストするために消費できます。 インスタンスが実行中の場合、CPUCreditBalance のクレジットは期限切れになりません。インスタンスが停止すると、CPUCredit Balance は保持されず、蓄積されたすべてのクレジットが失われます。 CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。 このメトリクスは、db.t2、db.t3、および db.t4g インスタンスにのみ適用されます。		クレジット (vCPU 分)

メトリクス	説明	Applies to	単位
	Note T DB インスタンスクラスは、開発サーバーおよびテストサーバー、またはその他の本稼働以外のサーバーにのみ使用することをお勧めします。T インスタンスクラスの詳細については、「DB インスタンスクラスタイプ」を参照してください。 起動クレジットは Amazon RDS でも Amazon EC2 と同じように機能します。詳細については、「Linux インスタンス向け Amazon Elastic Compute Cloud ユーザーガイド」の「起動クレジット」を参照してください。		
`CPUSurplusCreditBalance`	`CPUCreditBalance` 値がゼロの場合に unlimited インスタンスによって消費された余剰クレジットの数。 `CPUSurplusCreditBalance` 値は獲得した CPU クレジットによって支払われます。余剰クレジットの数が、24 時間にインスタンスが獲得できるクレジットの最大数を超えている場合、最大数を超えて消費された余剰クレジットに対しては料金が発生します。 CPU クレジットメトリクスは5 分間隔でのみ利用可能です。	すべて	クレジット (vCPU 分)

メトリクス	説明	Applies to	単位
CPUSurplusCreditsCharged	獲得 CPU クレジットにより支払われないために追加料金が発生した、消費された余剰クレジットの数。 消費された余剰クレジットは以下のいずれかの状況に当てはまると料金が発生します。 • 消費された余剰クレジットが、インスタンスが 24 時間に獲得できる最大クレジット数を超えている。最大数を越えて消費された余剰クレジットは時間の最後に課金されます。 • インスタンスが停止または終了した。 • インスタンスは unlimited から standard に切り替わります。 CPU クレジットメトリクスは5 分間隔でのみ利用可能です。	すべて	クレジット (vCPU 分)

メトリクス	説明	Applies to	単位
DatabaseConnections	データベースインスタンスへのクライアントネットワーク接続の数。 データベースセッションの数は、メトリクスが示す値よりも多くなることがあります。これは、メトリクス値には以下が含まれないためです。 • ネットワークへの接続がなくなっているものの、データベースによるクリーンアップ行われていないセッション • データベースエンジンが、固有の目的のために作成したセッション • データベースエンジンの並列実行機能によって作成されたセッション • データベースエンジンのジョブスケジューラによって作成されたセッション • Amazon RDS の接続	すべて	カウント
DiskQueueDepth	未処理のディスク I/O アクセス (読み取り/書き込みリクエスト) の数。	すべて	カウント
DiskQueueDepthLogVolume	未処理のログボリュームディスク I/O アクセス (読み取り/書き込みリクエスト) の数。	専用ログボリュームが有効になっている DB インスタンス	カウント

メトリクス	説明	Applies to	単位
EBSByteBalance%	RDS データベースのバーストバケットに残っているスループットクレジットの割合。このメトリクスは基本モニタリング専用です。 メトリクス値は、データベースファイルを含むボリュームのみではなく、ルートボリュームを含むすべてのボリュームのスループットに基づいています。 このメトリクスをサポートするインスタンスサイズを確認するには、「Amazon EC2 ユーザーガイド」 の「デフォルトで最適化された EBS」の表で、アスタリスク (*) の付いたインスタンスサイズを参照してください。Sum 統計はこのメトリクスに該当しません。	すべて	割合 (%)

メトリクス	説明	Applies to	単位
EBSIOBalance%	RDS データベースのバーストバケットに残っている I/O クレジットの割合。このメトリクスは基本モニタリング専用です。 メトリック値は、データベースファイルを含むボリュームのみではなく、ルートボリュームを含むすべてのボリュームの IOPS に基づいています。 このメトリクスをサポートするインスタンスサイズを確認するには、「Amazon EC2 ユーザーガイド」の「Amazon EBS 最適化インスタンスタイプ」を参照してください。Sum 統計は、このメトリクスに該当しません。 このメトリクスは BurstBalance とは異なります。このメトリクスの使用方法については、Amazon EBS で最適化されたインスタンスバースト機能によるアプリケーションのパフォーマンスの向上とコスト削減を参照してください。	すべて	割合 (%)
FailedSQLServerAgentJobsCount	直近 1 分間に失敗した Microsoft SQL Server エージェントジョブの数。	Microsoft SQL Server	1 分あたりのカウント

メトリクス	説明	Applies to	単位
`FreeableMemory`	使用可能な RAM の容量。 MariaDB、MySQL、Oracle、および PostgreSQL DB インスタンスの場合、このメトリクスは `MemAvailable` の `/proc/meminfo` フィールドの値を報告します。	すべて	バイト
`FreeLocalStorage`	使用できるローカルストレージスペースの量。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		バイト

メトリクス	説明	Applies to	単位
FreeLocalStoragePercent	使用できるローカルストレージスペースのパーセンテージ。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		割合 (%)
FreeStorageSpace	使用可能なストレージ領域の容量。	すべて	バイト
FreeStorageSpaceLogVolume	ログボリュームで使用可能なストレージスペースの容量。	専用ログボリュームが有効になっている DB インスタンス	バイト
IamDbAuthConnectionRequests	DB インスタンスへの IAM 認証を使用した接続リクエストの数。	すべて	カウント
MaximumUsedTransactionIDs	最大使用済みトランザクション ID 数。	PostgreSQL	カウント

メトリクス	説明	Applies to	単位
NetworkReceiveThroughput	モニタリングとレプリケーションに使用する顧客データベーストラフィックと Amazon RDS トラフィックの両方を含む、DB インスタンスの受信ネットワークトラフィック。	すべて	1 秒あたりのバイト数
NetworkTransmitThroughput	モニタリングとレプリケーションに使用する顧客データベーストラフィックと Amazon RDS トラフィックの両方を含む、DB インスタンスの送信ネットワークトラフィック。	すべて	1 秒あたりのバイト数
OldestLogicalReplicationSlotLag	Amazon RDS の遅延サイズは、ソースデータベースでトランザクションをコミットする時刻と、RDS がレプリカデータベースにトランザクションを適用する時刻の差を示します。	PostgreSQL	バイト
OldestReplicationSlotLag	受信した先行書き込み (WAL) データに関して最も遅延の長いレプリカの遅延サイズ。	PostgreSQL	バイト
ReadIOPS	1 秒あたりのディスク読み取り I/O オペレーションの平均回数。	すべて	1 秒あたりのカウント数

メトリクス	説明	Applies to	単位
ReadIOPSLocalStorage	1秒あたりのローカルストレージへのディスク読み取りI / O操作の平均数。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		1 秒あたりのカウント数
ReadIOPSLogVolume	ログボリュームに対して 1 秒あたりのディスク読み取り I/O オペレーションの平均回数。	専用ログボリュームが有効になっている DB インスタンス	1 秒あたりのカウント数
ReadLatency	1 回のディスク I/O 操作にかかる平均時間。	すべて	秒

メトリクス	説明	Applies to	単位
ReadLatencyLocalStorage	ローカルストレージのディスクI / O操作ごとにかかる平均時間。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		秒
ReadLatencyLogVolume	ログボリュームに対して 1 回のディスク I/O オペレーションにかかる平均時間。	専用ログボリュームが有効になっている DB インスタンス	秒
ReadThroughput	1 秒あたりのディスクからの平均読み取りバイト数。	すべて	1 秒あたりのバイト数

メトリクス	説明	Applies to	単位
ReadThroughputLocalStorage	ローカルストレージ用にディスクから読み取られた1秒あたりの平均バイト数。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		1 秒あたりのバイト数
ReadThroughputLogVolume	ログボリュームに対して 1 秒あたりのディスクからの平均読み取りバイト数。	専用ログボリュームが有効になっている DB インスタンス	1 秒あたりのバイト数

メトリクス	説明	Applies to	単位
ReplicaLag	リードレプリカの設定では、ソース DB インスタンスからリードレプリカ DB インスタンスまでの遅れ時間。MariaDB、Microsoft SQL Server、MySQL、Oracle、および PostgreSQL のリードレプリカに適用されます。 マルチ AZ の DB クラスターでは、ライター DB インスタンスの最新のトランザクションと、リーダー DB インスタンスで最後に適用されたトランザクションとの時間の差。		秒
ReplicationChannelLag	マルチソースレプリカ設定の場合、マルチソースレプリカの特定のチャネルがソース DB インスタンスから遅れる時間。詳細については、「the section called “マルチソースレプリケーションチャネルのモニタリング”」を参照してください。	MySQL	秒
ReplicationSlotDiskUsage	レプリケーションスロットファイルで使用されているディスク容量。	PostgreSQL	バイト
SwapUsage	DB インスタンスで使用するスワップ領域の量。	MariaDB MySQL Oracle PostgreSQL	バイト

メトリクス	説明	Applies to	単位
TempDbAvailableDataSpace	tempdb の使用可能なデータスペースの量と tempdb が配置されているボリューム。 このメトリクスを使用して、tempdb データスペースの可用性をモニタリングし、それに応じて容量を計画します。値が小さい場合は、ストレージを増加するか、tempdb を頻繁に使用するクエリを最適化する必要が生じることがあります。	SQL Server	バイト
TempDbAvailableLogSpace	tempdb の使用可能なログスペースの量と tempdb が配置されているボリューム。 このメトリクスを使用して、tempdb ログスペースの可用性をモニタリングし、トランザクションログが満杯状態になるのを防止します。大規模なトランザクションや高い同時実行性によって大量のログアクティビティが生成されるワークロードに不可欠です。	SQL Server	バイト
TempDbDataFileUsage	tempdb で使用されているデータファイルの割合。このメトリクスは、ファイルが増加する可能性を考慮しません。 このメトリクスを使用して、tempdb データファイルの使用率をモニタリングし、潜在的なパフォーマンスのボトルネックを特定します。値が大きい場合は、大きな一時オブジェクトの作成や tempdb サイズの増加をもたらすクエリの最適化が必要になることがあります。	SQL Server	割合 (%)

メトリクス	説明	Applies to	単位
TempDbLogFileUsage	tempdb で使用されている口グファイルの割合。このメトリクスは、ファイルが増加する可能性を考慮しません。 このメトリクスを使用して、tempdb ログファイルの使用率をモニタリングし、パフォーマンスの問題を防止します。値が大きい場合は、長時間稼働のトランザクションや過剰な口グ記録アクティビティにより、データベース全体のパフォーマンスが影響を受けている可能性があります。	SQL Server	割合 (%)
TransactionLogsDiskUsage	トランザクション口グで使用されているディスク容量。	PostgreSQL	バイト
TransactionLogsGeneration	1 秒あたりに生成されるトランザクション口グのサイズ。	PostgreSQL	1 秒あたりのバイト数
WriteIOPS	1 秒あたりのディスク書き込み I/O オペレーションの平均回数。	すべて	1 秒あたりの力ウント数

メトリクス	説明	Applies to	単位
`WriteIOPSLocalStorage`	ローカルストレージでの 1 秒あたりのディスク書き込み I/O 操作の平均回数。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		1 秒あたりのカウント数
`WriteIOPSLogVolume`	ログボリュームに対する 1 秒あたりのディスク書き込み I/O オペレーションの平均回数。	専用ログボリュームが有効になっている DB インスタンス	1 秒あたりのカウント数
`WriteLatency`	1 回のディスク I/O 操作にかかる平均時間。	すべて	秒

メトリクス	説明	Applies to	単位
WriteLatencyLocalStorage	ローカルストレージのディスク I/O 操作ごとにかかる平均時間。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		秒
WriteLatencyLogVolume	ログボリュームに対して 1 回のディスク I/O オペレーションにかかる平均時間。	専用ログボリュームが有効になっている DB インスタンス	秒
WriteThroughput	1 秒あたりのディスクへの平均書き込みバイト数。	すべて	1 秒あたりのバイト数
WriteThroughputLogVolume	ログボリュームに対して 1 秒あたりのディスクへの平均書き込みバイト数。	専用ログボリュームが有効になっている DB インスタンス	1 秒あたりのバイト数

メトリクス	説明	Applies to	単位
`WriteThroughputLocalStorage`	ローカルストレージのためにディスクに書き込まれる1秒あたりの平均バイト数。 このメトリクスは、NVMe SSD インスタンスストアボリュームを持つ DB インスタンスクラスにのみ適用されます。NVMe SSD インスタンスストアボリュームを使用する Amazon EC2 インスタンスについては、インスタンスストアボリュームを参照してください。同等の RDS DB インスタンスクラスには同じインスタンスストアボリュームがあります。例えば、db.m6gd および db.r6gd DB インスタンスクラスには NVMe SSD インスタンスストアボリュームがあります。		1 秒あたりのバイト数

Amazon RDS の Amazon CloudWatch 使用状況メトリクス

Amazon CloudWatch の `AWS/Usage` 名前空間には、Amazon RDS サービスクォータのアカウントレベルの使用状況メトリクスが含まれています。CloudWatch では、すべての AWS リージョン の使用状況メトリクスを自動的に収集します。

詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch 使用状況メトリクスの使用」を参照してください。クォータの詳細については、Service Quotas ユーザーガイドの Amazon RDS のクォータと制約 および クォータの引き上げのリクエストを参照してください。

メトリクス	説明	単位*
`AllocatedStorage`	すべての DB インスタンスの合計ストレージ この合計では、一時的な移行インスタンスは除外されます。	Gigabytes

メトリクス	説明	単位*
AuthorizationsPerDBSecurityGroup	AWS アカウント の DB セキュリティグループあたりの進入ルールの数。使用される値は、アカウントの DB セキュリティグループ内の進入ルールの最大数です。アカウント内の他の DB セキュリティグループでは、進入ルールの数が少ない場合があります。	カウント
CustomEndpointsPerDBCluster	AWS アカウントの DB クラスターあたりのカスタムエンドポイントの数。使用される値は、アカウントの DB クラスター内のカスタムエンドポイントの最大数です。アカウントの他の DB クラスターでは、カスタムエンドポイントの数が少ない場合があります。	カウント
CustomEngineVersions	AWS アカウントの Amazon RDS Custom のカスタムエンジンバージョン (CEV) の数。	カウント
DBClusterParameterGroups	AWS アカウント の DB クラスターパラメータグループの数。カウントでは、デフォルトのパラメータグループは除外されます。	カウント
DBClusterRoles	AWS アカウントの DB クラスターあたりの関連付けられた AWS Identity and Access Management (IAM) ロールの数。使用される値は、アカウント内の DB クラスターに関連付けられた IAM ロールの最大数です。アカウントの他の DB クラスターでは、関連付けられた IAM ロールの数がより少ない場合があります。	カウント
DBClusters	AWS アカウント の Amazon Aurora DB クラスターの数。	カウント
DBInstanceRoles	AWS アカウントの DB インスタンスあたりの関連付けられた AWS Identity and Access Management (IAM) ロールの数。使用される値は、アカウント内の DB インスタンスに関連付けられた IAM ロールの最大数です。アカウント内の他の DB インスタンスは、関連付けられた IAM ロールの数がより少ない場合があります。	カウント
DBInstances	AWS アカウント の DB インスタンスの数。	カウント

メトリクス	説明	単位*
DBParameterGroups	AWS アカウント の DB パラメータグループの数。カウントでは、デフォルトの DB パラメータグループは除外されます。	カウント
DBSecurityGroups	AWS アカウント 内のセキュリティグループの数。カウントでは、デフォルトのセキュリティグループおよびデフォルトの VPC セキュリティグループは除外されます。	カウント
DBSubnetGroups	AWS アカウント の DB サブネットグループの数。カウントでは、デフォルトのサブネットグループは除外されます。	カウント
EventSubscriptions	AWS アカウントのイベント通知サブスクリプションの数。	カウント
Integrations	AWS アカウントの Amazon Redshift とのゼロ ETL 統合の数。	カウント
ManualClusterSnapshots	AWS アカウント にある、手動で作成された DB クラスタースナップショットの数。このカウントでは、無効なスナップショットは除外されます。	カウント
ManualSnapshots	AWS アカウント にある、手動で作成された DB スナップショットの数。このカウントでは、無効なスナップショットは除外されます。	カウント
OptionGroups	AWS アカウント 内のオプショングループの数。カウントでは、デフォルトのオプショングループは除外されます。	カウント
Proxies	AWS アカウントの RDS プロキシの数。	カウント
ReadReplicasPerMaster	アカウントの DB インスタンスあたりのリードレプリカの数。使用される値は、アカウント内の DB インスタンスのリードレプリカの最大数です。アカウントの他の DB インスタンスでは、リードレプリカの数がより少ない場合があります。	カウント

メトリクス	説明	単位*
ReservedDBInstances	AWS アカウント の予約済み DB インスタンスの数。カウントでは、使用停止または拒否されたインスタンスは除外されます。	カウント
SubnetsPerDBSubnetGroup	AWS アカウントの DB サブネットグループあたりのサブネットの数。アカウント内の DB サブネットグループのサブネットの最大数。アカウントの他の DB サブネットグループでは、サブネットの数がより少ない場合があります。	カウント

> **Note**
>
> Amazon RDS は、使用状況メトリクスのユニットを CloudWatch に発行しません。ユニットはドキュメントにのみ表示されます。

Amazon RDS の Amazon CloudWatch ディメンション

次の表に示す任意のディメンションを使用して、Amazon RDS メトリックスデータをフィルタリングができます。

ディメンション	以下で要求されたデータをフィルタリングします。
DBInstanceIdentifier	特定の DB インスタンス。
DatabaseClass	データベースクラスのすべてのインスタンス。例えば、db.r5.large というデータベースクラスに属するすべてのインスタンスのメトリクスを集計できます。
EngineName	特定されたエンジン名のみ。例えば、メトリクスを組み合わせて、postgres というエンジン名を有する全インスタンスを抽出することができます。
SourceRegion	指定されたリージョンのみ。例えば、us-east-1 リージョンのすべての DB インスタンスのメトリクスを集計できます。

ディメンション	以下で要求されたデータをフィルタリングします。
`DbInstanceIdentifier, VolumeName`	1 つのインスタンスのボリュームあたりのメトリクス。 RDS は、複数のストレージボリュームのメトリクスをキャプチャします。

> **Note**
>
> 追加のストレージボリュームを使用している場合は、`DBInstanceIdentifier` ディメンションの下にストレージメトリクスの集計が表示されます。ボリュームごとのストレージメトリクスを表示するには、`DbInstanceIdentifier, VolumeName` ディメンションを使用します。

Amazon RDS Performance Insights の Amazon CloudWatch メトリクス

Performance Insights はメトリクスを自動的に Amazon CloudWatch に発行します。Performance Insights から同じデータに対してクエリを実行できますが、CloudWatch にメトリクスを含めると、CloudWatch アラームを追加しやすくなります。また、既存の CloudWatch ダッシュボードにメトリクスを追加しやすくなります。

メトリクス	説明
DBLoad	データベースのアクティブセッション数。通常、アクティブセッションの平均数に関するデータを使用します。Performance Insights で、このデータは `db.load.avg` としてクエリされます。
DBLoadCPU	待機イベントタイプが CPU であるアクティブセッションの数。Performance Insights で、このデータは、待機イベントタイプ `db.load.avg` でフィルタ処理された `CPU` としてクエリされます。

メトリクス	説明
DBLoadNonCPU	待機イベントタイプが CPU でないアクティブセッションの数。
DBLoadRelativeToNumVCPUs	データベースの仮想 CPU 数に対する DB 負荷の比率。

> **Note**
>
> これらのメトリクスは、DB インスタンスにロードがある場合にのみ CloudWatch に公開されます。

これらのメトリクスは、CloudWatch コンソール、AWS CLI、または CloudWatch API を使用して調査できます。特別な Metric Math 関数を使用して、他の Performance Insights カウンターメトリクスを調べることもできます。詳細については、「CloudWatch での他の Performance Insights カウンターメトリクスのクエリ」を参照してください。

例えば、DBLoad メトリクスの統計情報は、get-metric-statistics コマンドを実行して取得できます。

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

次のコマンドでは、以下のような出力が生成されます。

```
{
  "Datapoints": [
  {
  "Timestamp": "2021-07-19T21:30:00Z",
  "Unit": "None",
  "Average": 2.1
```

```
},
{
"Timestamp": "2021-07-19T21:34:00Z",
"Unit": "None",
"Average": 1.7
},
{
"Timestamp": "2021-07-19T21:35:00Z",
"Unit": "None",
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


CloudWatch の詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch とは」を参照してください。

CloudWatch での他の Performance Insights カウンターメトリクスのクエリ

> **Note**
>
> Database Insights のアドバンストモードを有効にすると、Amazon RDS は Performance Insights カウンターメトリクスを Amazon CloudWatch に発行します。Database Insights では、`DB_PERF_INSIGHTS` Metric Math 関数を使用する必要はありません。CloudWatch Database Insights ダッシュボードを使用して、Performance Insights カウンターメトリクスの検索、クエリ、アラームの設定を行うことができます。

CloudWatch から RDS Performance Insights メトリクスのクエリ、アラーム、グラフを実行できます。CloudWatch の `DB_PERF_INSIGHTS` Metric Math 関数を使用して、DB インスタンスに関する情報にアクセスできます。この関数を使用すると、CloudWatch に直接レポートされない Performance Insights メトリクスを使用して、新しい時系列を作成できます。

新しい Metric Math 関数を使用するには、CloudWatch コンソールの [メトリクスの選択] 画面の [数式を追加] ドロップダウンメニューをクリックします。これを使用して、Performance Insights メトリクス、または CloudWatch と Performance Insights メトリクス (1 分未満のメトリクスの高解像度アラームなど) の組み合わせに関するアラームとグラフを作成できます。`get-metric-data` リクエストに Metric Math 式を含めることで、プログラムでこの関数を使用することもできます。詳細については、「Metric Math 構文と関数」および「AWS データベースから Performance Insights カウンターメトリクスのアラームを作成する」を参照してください。

Performance Insights カウンターメトリクス

カウンターメトリクスは、Performance Insights ダッシュボードのオペレーティングシステムとデータベースのパフォーマンスメトリクスのことです。カウンターメトリクスを DB ロードと関連付けることで、パフォーマンスの問題を特定して分析できます。メトリクス値を取得するには、統計関数をメトリクスに追加する必要があります。例えば、`os.memory.active` メトリクスでサポートされている関数は、`.avg`、`.min`、`.max`、`.sum`、および `.sample_count` です。

カウンターメトリクスは 1 分に 1 回収集されます。OS メトリクスの収集は、拡張モニタリングがオンかオフかによって異なります。拡張モニタリングがオフになっている場合、OS メトリックは 1 分に 1 回収集されます。拡張モニタリングがオンになっている場合、選択した期間の OS メトリックが収集されます。拡張モニタリングのオンまたはオフの詳細については、拡張モニタリングのオンとオフを切り替える を参照してください。

トピック

- Performance Insights オペレーティングシステムのカウンター
- Amazon RDS for MariaDB および MySQL の Performance Insights カウンター
- Amazon RDS for Microsoft SQL Server の Performance Insights カウンター
- Amazon RDS for Oracle の Performance Insights カウンター
- Amazon RDS for PostgreSQL の Performance Insights カウンター

Performance Insights オペレーティングシステムのカウンター

次のオペレーティングシステムカウンターは、os のプレフィックスが付き、RDS for SQL Server 以外のすべての RDS エンジン では、Performance Insights で使用できます。

DB インスタンスで使用可能なカウンターメトリクスのリストについて、`ListAvailableResourceMetrics` API を使用できます。詳細については、「Amazon RDS Performance Insights API リファレンスガイド」の「ListAvailableResourceMetics」を参照してください。

Counter	タイプ	単位	メトリクス	説明
アクティブ	メモリ	Kilobytes	os.memory.active	割り当てられたメモリの量 (キロバイト単位)。
バッファ	メモリ	Kilobytes	os.memory.buffers	ストレージデバイスへの書き込み前に I/O バッファリングリクエストに使用されたメモリの量 (キロバイト単位)。
キャッシュ済み	メモリ	Kilobytes	os.memory.cached	ファイルシステムベースの I/O のキャッシュに使用されたメモリの量 (キロバイト単位)。

Counter	タイプ	単位	メトリクス	説明
DB キャッシュ	メモリ	バイト	os.memory.db.cache	tmpfs (shmem) を含めて、データベースプロセスがページキャッシュに使用したメモリの量 (バイト単位)。
DB レジデントセットサイズ	メモリ	バイト	os.memory.db.residentSetSize	tmpfs (shmem) を含めずに、データベースプロセスが匿名キャッシュとスワップキャッシュに使用したメモリの量 (バイト単位)。
DB スワップ	メモリ	バイト	os.memory.db.swap	データベースプロセスがスワップに使用したメモリの量 (バイト単位)。
ダーティ	メモリ	Kilobytes	os.memory.dirty	変更されたがストレージ内のその関連データブロックに書き込まれなかった RAM 内のメモリページの量 (キロバイト単位)。

Counter	タイプ	単位	メトリクス	説明
空き	メモリ	Kilobytes	os.memory.free	未割り当てのメモリの量 (キロバイト単位)。
huge ページ (空き)	メモリ	ページ	os.memory .hugePagesFree	空き huge ページの数。huge ページは Linux カーネルの機能です。
huge ページ (予約)	メモリ	ページ	os.memory .hugePagesRsvd	コミットされた huge ページの数。
huge ページサイズ	メモリ	Kilobytes	os.memory .hugePagesSize	各 huge ページユニットのサイズ (キロバイト単位)。
huge ページ (余剰)	メモリ	ページ	os.memory .hugePagesSurp	使用可能な huge ページの余剰数/合計数。
huge ページ (合計)	メモリ	ページ	os.memory .hugePagesTotal	huge ページの合計数。
非アクティブ	メモリ	Kilobytes	os.memory .inactive	最も使用されていないメモリページの量 (キロバイト単位)。

Counter	タイプ	単位	メトリクス	説明
マップ済み	メモリ	Kilobytes	os.memory.mapped	プロセスアドレス空間内でメモリマップされているファイルシステムの内容の合計量 (キロバイト単位)。
メモリ不足キルカウント	メモリ	Kills	os.memory.outOfMemoryKillCount	前回の収集間隔で発生した OOM キルの数。
ページテーブル	メモリ	Kilobytes	os.memory.pageTables	ページテーブルが使用中のメモリの量 (キロバイト単位)。
スラブ	メモリ	Kilobytes	os.memory.slab	再利用可能なカーネルデータ構造体の量 (キロバイト単位)。
Total	メモリ	Kilobytes	os.memory.total	メモリの合計量 (キロバイト単位)。
書き戻し	メモリ	Kilobytes	os.memory.writeback	バックアップストレージにまだ書き込み中の RAM 内のダーティページの量 (キロバイト単位)。

Counter	タイプ	単位	メトリクス	説明
ゲスト	CPU 使用率	割合 (%)	os.cpuUtilization.guest	ゲストプログラムが使用中の CPU の使用率。
アイドル状態	CPU 使用率	割合 (%)	os.cpuUtilization.idle	アイドル状態の CPU の使用率。
irq	CPU 使用率	割合 (%)	os.cpuUtilization.irq	ソフトウェア割り込みが使用中の CPU の使用率。
Nice	CPU 使用率	割合 (%)	os.cpuUtilization.nice	最も低い優先順位で実行されているプログラムが使用中の CPU の使用率。
Steal	CPU 使用率	割合 (%)	os.cpuUtilization.steal	他の仮想マシンが使用中の CPU の使用率。
システム	CPU 使用率	割合 (%)	os.cpuUtilization.system	カーネルが使用中の CPU の使用率。
Total	CPU 使用率	割合 (%)	os.cpuUtilization.total	使用中の CPU の合計使用率。この値は nice 値を含みます。
ユーザー	CPU 使用率	割合 (%)	os.cpuUtilization.user	ユーザープログラムが使用中の CPU の使用率。

Counter	タイプ	単位	メトリクス	説明
Wait	CPU 使用率	割合 (%)	os.cpuUtilization.wait	I/O アクセスを待機中の CPU の未使用率。
読み取り IO PS	ディスク IO	1 秒あたりのリクエスト	os.diskIO.<devicename>.readIOsPS	読み取りオペレーションの 1 秒あたりの数。
書き込み IO PS	ディスク IO	1 秒あたりのリクエスト	os.diskIO.<devicename>.writeIOsPS	書き込みオペレーションの 1 秒あたりの数。
平均キュー長さ	ディスク IO	リクエスト	os.diskIO.<devicename>.avgQueueLen	I/O デバイスのキューで待機中のリクエストの数。
平均リクエストサイズ	ディスク IO	リクエスト	os.diskIO.<devicename>.avgReqSz	I/O デバイスのキューで待機中のリクエストの数。
待機中	ディスク IO	ミリ秒	os.diskIO.<devicename>.await	リクエストへの応答に必要なミリ秒数 (キュー時間とサービス時間を含む)。
読み取り IO PS	ディスク IO	リクエスト	os.diskIO.<devicename>.readIOsPS	読み取りオペレーションの 1 秒あたりの数。

Counter	タイプ	単位	メトリクス	説明
読み取り KB	ディスク IO	Kilobytes	os.diskIO.<devicename>.readKb	読み取りの合計キロバイト数。
読み取り KB PS	ディスク IO	KB/秒	os.diskIO.<devicename>.readKbPS	読み取りの 1 秒あたりのキロバイト数。
Rrqm PS	ディスク IO	1 秒あたりのリクエスト	os.diskIO.<devicename>.rrqmPS	キューに入れられてマージされた読み取りリクエストの 1 秒あたりの数。
TPS	ディスク IO	1 秒あたりのトランザクション	os.diskIO.<devicename>.tps	I/O トランザクションの 1 秒あたりの数。
使用率	ディスク IO	割合 (%)	os.diskIO.<devicename>.util	リクエスト発行中の CPU 時間の消費率。
書き込み KB	ディスク IO	Kilobytes	os.diskIO.<devicename>.writeKb	書き込みの合計キロバイト数。
書き込み KB PS	ディスク IO	KB/秒	os.diskIO.<devicename>.writeKbPS	書き込みの 1 秒あたりのキロバイト数。

Counter	タイプ	単位	メトリクス	説明
Wrqm PS	ディスク IO	1 秒あたりのリクエスト	os.diskIO.<devicename>.wrqmPS	キューに入れられてマージされた書き込みリクエストの 1 秒あたりの数。
ブロック	タスク	タスク	os.tasks.blocked	ブロックされているタスクの数。
実行中	タスク	タスク	os.tasks.running	実行中のタスクの数。
Sleeping	タスク	タスク	os.tasks.sleeping	スリープ中のタスクの数。
停止	タスク	タスク	os.tasks.stopped	停止中のタスクの数。
Total	タスク	タスク	os.tasks.total	タスクの合計数。
ゾンビ	タスク	タスク	os.tasks.zombie	アクティブな親タスクの非アクティブな子タスクの数。
1	負荷平均分	プロセス	os.loadAverageMinute.one	過去 1 分間に CPU 時間をリクエストしたプロセスの数。
15	負荷平均分	プロセス	os.loadAverageMinute.fifteen	過去 15 分間に CPU 時間をリクエストしたプロセスの数。

Counter	タイプ	単位	メトリクス	説明
Five	負荷平均分	プロセス	os.loadAverageMinute.five	過去 5 分間に CPU 時間をリクエストしたプロセスの数。
キャッシュ済み	スワップ	Kilobytes	os.swap.cached	キャッシュメモリとして使用されたスワップメモリの量 (キロバイト単位)。
空き	スワップ	Kilobytes	os.swap.free	空きスワップメモリの量 (キロバイト単位)。
In (中にある)	スワップ	Kilobytes	os.swap.in	ディスクからスワップされたメモリの量 (キロバイト単位)。
Out	スワップ	Kilobytes	os.swap.out	ディスクにスワップされたメモリの量 (キロバイト単位)。
Total	スワップ	Kilobytes	os.swap.total	使用可能なスワップメモリの合計量 (キロバイト単位)。

Counter	タイプ	単位	メトリクス	説明
最大ファイル数	ファイルシステム	ファイル	os.fileSys.maxFiles	すべてのストレージボリュームにわたってファイルシステム用に作成できるファイルの最大数。
使用済みファイル	ファイルシステム	ファイル	os.fileSys.usedFiles	すべてのストレージボリュームにわたるファイルシステム内のファイルの数。
使用済みファイルパーセント	ファイルシステム	ファイル	os.fileSys.usedFilePercent	すべてのストレージボリュームで使用可能なファイルの割合。
使用率	ファイルシステム	割合 (%)	os.fileSys.usedPercent	すべてのストレージボリュームにわたってファイルシステムが使用中のディスク領域の割合。

Counter	タイプ	単位	メトリクス	説明
使用済み	ファイルシステム	Kilobytes	os.fileSys.used	すべてのストレージボリュームにわたってファイルシステム内のファイルが使用中のディスク領域の量 (キロバイト単位)。
Total	ファイルシステム	Kilobytes	os.fileSys.total	すべてのストレージボリュームにわたるファイルシステムに使用できるディスク領域の合計 (キロバイト単位)。
最大ファイル数	ファイルシステム	ファイル	os.fileSys.<volumeName>.maxFiles	ストレージボリューム用に作成できるファイルの最大数。
使用済みファイル	ファイルシステム	ファイル	os.fileSys.<volumeName>.usedFiles	ストレージボリューム内のファイルの数。
使用済みファイルパーセント	ファイルシステム	ファイル	os.fileSys.<volumeName>.usedFilePercent	すべてのストレージボリュームで使用中の使用可能なファイルの割合。

Counter	タイプ	単位	メトリクス	説明
使用率	ファイルシステム	割合 (%)	os.fileSys.<volumeName>.usedPercent	ストレージボリュームが使用中のディスク領域の割合。
使用済み	ファイルシステム	Kilobytes	os.fileSys.<volumeName>.used	ストレージボリューム内のファイルが使用中のディスク領域の量 (キロバイト単位)。
Total	ファイルシステム	Kilobytes	os.fileSys.<volumeName>.total	ストレージボリューム内で利用可能なディスク容量の合計 (キロバイト単位)。
受信	Network	1 秒あたりのバイト数	os.network.rx	1 秒あたりの受信バイト数。
送信	Network	1 秒あたりのバイト数	os.network.tx	1 秒あたりのアップロードバイト数。
ACU 使用率	General	割合 (%)	os.general.acuUtilization	設定された最大容量のうち、現在の容量の割合。
最大構成 ACU	General	ACU	os.general.maxConfiguredAcu	ユーザーが設定した Aurora 容量ユニット (ACU 数) での最大容量。

Counter	タイプ	単位	メトリクス	説明
最小構成 ACU	General	ACU	os.general.minConfiguredAcu	ユーザーが設定した最小容量 (ACU 数)。
Num VCPU	General	vCPU	os.general.numVCPUs	DB インスタンスの仮想 CPU (vCPU) の数。
サーバーレスデータベース容量	General	ACU	os.general.serverlessDatabaseCapacity	ACU 内の DB インスタンスの現在の容量。

Amazon RDS for MariaDB および MySQL の Performance Insights カウンター

以下のデータベースカウンターは、Amazon RDS for MariaDB および MySQL の Performance Insights で利用できます。

トピック

- RDS for MariaDB および RDS for MySQL のネイティブカウンター
- Amazon RDS for MariaDB および MySQL の非ネイティブカウンター

RDS for MariaDB および RDS for MySQL のネイティブカウンター

ネイティブメトリクスは、Amazon RDS ではなく、データベースエンジンによって定義されます。これらのネイティブメトリクスの定義については、MySQL ドキュメントの「Server Status Variables」(8.0 の場合) および「「Server Status Variables」(8.4 の場合) を参照してください。

Counter	タイプ	単位	メトリクス
Com_analyze	SQL	1 秒あたりのクエリ数	db.SQL.Com_analyze
Com_optimize	SQL	1 秒あたりのクエリ数	db.SQL.Com_optimize

Counter	タイプ	単位	メトリクス
Com_select	SQL	1 秒あたりのクエリ数	db.SQL.Com_select
接続	SQL	MySQL サーバーへの 1 分あたりの接続試行回数 (成功の是非)	db.Users.Connections
Innodb_rows_deleted	SQL	1 秒あたりの行数	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	1 秒あたりの行数	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	1 秒あたりの行数	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	1 秒あたりの行数	db.SQL.Innodb_rows_updated
Select_full_join	SQL	1 秒あたりのクエリ数	db.SQL.Select_full_join
Select_full_range_join	SQL	1 秒あたりのクエリ数	db.SQL.Select_full_range_join
Select_range	SQL	1 秒あたりのクエリ数	db.SQL.Select_range
Select_range_check	SQL	1 秒あたりのクエリ数	db.SQL.Select_range_check
Select_scan	SQL	1 秒あたりのクエリ数	db.SQL.Select_scan

Counter	タイプ	単位	メトリクス
Slow_queries	SQL	1 秒あたりのクエリ数	db.SQL.Slow_queries
Sort_merge_passes	SQL	1 秒あたりのクエリ数	db.SQL.Sort_merge_passes
Sort_range	SQL	1 秒あたりのクエリ数	db.SQL.Sort_range
Sort_rows	SQL	1 秒あたりのクエリ数	db.SQL.Sort_rows
Sort_scan	SQL	1 秒あたりのクエリ数	db.SQL.Sort_scan
Questions	SQL	1 秒あたりのクエリ数	db.SQL.Questions
Innodb_row_lock_time	ロック	ミリ秒 (平均)	db.Locks.Innodb_row_lock_time
Table_locks_immediate	ロック	1 秒あたりのリクエスト	db.Locks.Table_locks_immediate
Table_locks_waited	ロック	1 秒あたりのリクエスト	db.Locks.Table_locks_waited
Aborted_clients	ユーザー	接続	db.Users.Aborted_clients
Aborted_connects	ユーザー	接続	db.Users.Aborted_connects
max_connections	Users	Connections	db.User.max_connections
Threads_created	ユーザー	接続	db.Users.Threads_created
Threads_running	ユーザー	接続	db.Users.Threads_running

Counter	タイプ	単位	メトリクス
Innodb_data_writes	I/O	1 秒あたりのオペレーション数	db.IO.Innodb_data_writes
Innodb_dblwr_writes	I/O	1 秒あたりのオペレーション数	db.IO.Innodb_dblwr_writes
Innodb_log_write_requests	I/O	1 秒あたりのオペレーション数	db.IO.Innodb_log_write_requests
Innodb_log_writes	I/O	1 秒あたりのオペレーション数	db.IO.Innodb_log_writes
Innodb_pages_written	I/O	1 秒あたりのページ数	db.IO.Innodb_pages_written
Created_tmp_disk_tables	Temp	1 秒あたりのテーブル数	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	1 秒あたりのテーブル数	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Cache	ページ	db.Cache.Innodb_buffer_pool_pages_data
Innodb_buffer_pool_pages_total	Cache	ページ	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	1 秒あたりのページ数	db.Cache.Innodb_buffer_pool_read_requests

Counter	タイプ	単位	メトリクス
Innodb_buffer_pool_reads	Cache	1 秒あたりのページ数	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	テーブル	db.Cache.Opened_tables
Opened_table_definitions	Cache	テーブル	db.Cache.Opened_table_definitions
Qcache_hits	Cache	クエリ	db.Cache.Qcache_hits

Amazon RDS for MariaDB および MySQL の非ネイティブカウンター

非ネイティブカウンターメトリクスは、Amazon RDS で定義されているカウンターです。非ネイティブメトリクスは、特定のクエリで取得するメトリクスである場合があります。非ネイティブメトリクスは派生メトリクスである場合もあります。この場合は、複数のネイティブカウンターが比率、ヒット率、またはレイテンシーの計算で使用されます。

Counter	タイプ	単位	メトリクス	説明	定義
innodb_buffer_pool_hits	Cache	reads	db.Cache.innoDB_buffer_pool_hits	InnoDB がバッファプールから満たすことができる読み取りの数。	`innodb_buffer_pool_read_requests - innodb_buffer_pool_reads`
innodb_buffer_pool_hit_rate	Cache	割合 (%)	db.Cache.innoDB_buffer_pool_hit_rate	InnoDB がバッファプールから満たすことができる読み取りの割合 (%)。	`100 * innodb_buffer_pool_read_requests / (innodb_buffer_pool_read_requests +`

Counter	タイプ	単位	メトリクス	説明	定義
					innodb_buffer_pool_reads)

Counter	タイプ	単位	メトリクス	説明	定義
innodb_buffer_pool_usage	Cache	割合 (%)	db.Cache.innoDB_buffer_pool_usage	データ (ページ) を含む InnoDB バッファプールの割合 (%)。 Note 圧縮テーブルを使用すると、この値は変動します。詳細に	`Innodb_buffer_pool_pages_data / Innodb_buffer_pool_pages_total * 100.0`

Counter	タイプ	単位	メトリクス	説明	定義
				ついては、My ドキュメントの「Serv Status Variab」(8.0 の場合) および「Serv Status Variab」(8.4 の場合) で、Iffer_ _page ta および	

Counter	タイプ	単位	メトリクス	説明	定義
				び Innodffer_ _pagetal に関する情報を参照してください。	
query_cache_hit_rate	Cache	割合 (%)	db.Cache.query_cache_hit_rate	MySQL 結果セットキャッシュ (クエリキャッシュ) ヒット率。	Qcache_hits / (QCache_hits + Com_select) * 100

Counter	タイプ	単位	メトリクス	説明	定義
innodb_datafile_writes_to_disk	I/O	writes	db.IO.innoDB_datafile_writes_to_disk	ディスクに対する InnoDB データファイル書き込みの数 (ダブル書き込みおよび REDO ログ書き込みオペレーションを除く)。	`Innodb_data_writes - Innodb_log_writes - Innodb_dblwr_writes`
innodb_rows_changed	SQL	Rows	db.SQL.innodb_rows_changed	InnoDB の行オペレーションの合計数。	`db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated`
active_transactions	トランザクション	トランザクション	db.Transactions.active_transactions	アクティブトランザクションの合計数。	`SELECT COUNT(1) AS active_transactions FROM INFORMATION_SCHEMA.INNODB_TRX`

Counter	タイプ	単位	メトリクス	説明	定義
trx_rseg_history_len	取引	なし	db.Transactions.trx_rseg_history_len	マルチバージョン同時実行制御を実装するために InnoDB トランザクションシステムによって管理される、コミットされたトランザクションの UNDO ログページのリスト。undo ログレコードの詳細については、MySQL ドキュメントの「InnoDB Multi-Versioning」(8.0 の場合) および「InnoDB Multi-Ver	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Counter	タイプ	単位	メトリクス	説明	定義
				sioning」(8.4 の場合) を参照してください。	
innodb_deadlocks	ロック	ロック	db.Locks.innodb_deadlocks	デッドロックの合計数。	`SELECT COUNT AS innodb_deadlocks FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_deadlocks'`
innodb_lock_timeouts	ロック	ロック	db.Locks.innodb_lock_timeouts	タイムアウトしたロックの総数。	`SELECT COUNT AS innodb_lock_timeouts FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_timeouts'`

Counter	タイプ	単位	メトリクス	説明	定義
innodb_row_lock_waits	ロック	ロック	db.Locks.innodb_row_lock_waits	行ロックを待機した合計数。	`SELECT COUNT AS innodb_row_lock_waits FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_row_lock_waits'`

Amazon RDS for Microsoft SQL Server の Performance Insights カウンター

以下のデータベースカウンターは、RDS for Microsoft SQL Server の Performance Insights で利用できます。

RDS for Microsoft SQL Server のネイティブカウンター

ネイティブメトリクスは、Amazon RDS ではなく、データベースエンジンによって定義されます。これらのネイティブメトリクスの定義は、Microsoft SQL Server ドキュメントの「SQL Server オブジェクトを使用する」にあります。

Counter	タイプ	単位	メトリクス
転送されたレコード	アクセス方法	1 秒あたりのレコード数	db.Access Methods.Forwarded Records
ページ分割	アクセス方法	1 秒あたりの分割数	db.Access Methods.Page Splits
バッファキャッシュヒット率	バッファマネージャー	Ratio	db.Buffer Manager.Buffer cache hit ratio
ページの平均寿命	バッファマネージャー	寿命 (秒)	db.Buffer Manager.Page life expectancy

Counter	タイプ	単位	メトリクス
ページ検索	バッファマネージャー	1 秒あたりの検索数	db.Buffer Manager.Page lookups
ページの読み取り	バッファマネージャー	1 秒あたりの読み取り数	db.Buffer Manager.Page reads
ページの書き込み	バッファマネージャー	1 秒あたりの書き込み数	db.Buffer Manager.Page writes
アクティブなトランザクション	データベース	トランザクション	db.Databases.Active Transactions (_Total)
フラッシュされたログバイト	データベース	1 秒あたりのフラッシュされたバイト数	db.Databases.Log Bytes Flushed (_Total)
ログフラッシュ待機	データベース	1 秒あたりの待機数	db.Databases.Log Flush Waits (_Total)
ログフラッシュ	データベース	1 秒あたりのフラッシュ	db.Databases.Log Flushes (_Total)
書き込みトランザクション	データベース	1 秒あたりのトランザクション	db.Databases.Write Transactions (_Total)
ブロックされたプロセス	一般的な統計	ブロックされたプロセス	db.General Statistics.Processes blocked
ユーザー接続	一般的な統計	接続	db.General Statistics.User Connections
ラッチ待機	ラッチ	1 秒あたりの待機数	db.Latches.Latch Waits
デッドロックの数。	ロック	1 秒あたりのデッドロック数	db.Locks.Number of Deadlocks (_Total)

Counter	タイプ	単位	メトリクス
保留中のメモリ許可	メモリマネージャー	メモリ許可	db.Memory Manager.Memory Grants Pending
バッチリクエスト	SQL 統計	1 秒あたりのリクエスト	db.SQL Statistic s.Batch Requests
SQL コンピレーション	SQL 統計	1 秒あたりのコンパイル数	db.SQL Statistics.SQL Compilations
SQL 再コンパイル	SQL 統計	1 秒あたりの再コンパイル数	db.SQL Statistics.SQL Re-Compilations

Amazon RDS for Oracle の Performance Insights カウンター

以下のデータベースカウンターは、RDS for Oracle の Performance Insights で利用できます。

RDS for Oracle のネイティブカウンター

ネイティブメトリクスは、Amazon RDS ではなく、データベースエンジンによって定義されます。これらのネイティブメトリクスの定義については、Oracle ドキュメントの「統計の説明」を参照してください。

> **Note**
>
> `CPU used by this session` カウンターメトリクスでは、値を使いやすくするために、単位はネイティブのセンチ秒からアクティブセッションに変換されました。例えば、DB ロードグラフの CPU 送信は、CPU の需要を表します。カウンターメトリクス `CPU used by this session` は、Oracle セッションで使用される CPU の容量を表します。CPU 送信と `CPU used by this session` カウンターメトリクスを比較することができます。CPU の需要が使用する CPU より高い場合、セッションは、CPU 時間待ちます。

Counter	タイプ	単位	メトリクス
このセッションで使用される CPU	ユーザー	アクティブなセッション	このセッションで使用される db.User.CPU
クライアントとの間の SQL*Net ラウンドトリップ	ユーザー	1 秒あたりのラウンドトリップ	クライアントとの間の db.User.SQL ラウンドトリップ
SQL*Net 経由でクライアントから受信したバイト数	ユーザー	1 秒あたりのバイト数	SQL*Net 経由でクライアントから受信した db.User. バイト数
ユーザーコミット	ユーザー	1 秒あたりのコミット数	db.User.user コミット
ログオン累積数	ユーザー	1 秒あたりのログオン数	db.User.logons 累積合計
ユーザーの呼び出し	ユーザー	1 秒あたりの呼び出し数	db.User.user 呼び出し
SQL*Net からクライアントに送信されるデータ (バイト)	ユーザー	1 秒あたりのバイト数	SQL*Net からクライアントに送信される db.User. バイト数
ユーザーのロールバック	ユーザー	1 秒あたりのロールバック数	db.User.user ロールバック
再実行サイズ	再実行	1 秒あたりのバイト数	db.Redo.redo サイズ
分析数 (合計)	SQL	1 秒あたりの分析数	db.SQL.parse count (合計)
分析数 (ハード)	SQL	1 秒あたりの分析数	db.SQL.parse count (ハード)

Counter	タイプ	単位	メトリクス
取得したテーブルスキャン行	SQL	1 秒あたりの行数	取得した db.SQL.テーブルスキャン行
ソート (メモリ)	SQL	1 秒あたりのソート	db.SQL.sorts (メモリ)
ソート (ディスク)	SQL	1 秒あたりのソート	db.SQL.sorts (ディスク)
ソート (行)	SQL	1 秒あたりのソート	db.SQL.sorts (行)
物理的な読み取りバイト	Cache	1 秒あたりのバイト数	db.Cache.physical 読み取りバイト数
DB ブロック取得	Cache	1 秒あたりのブロック数	db.Cache.db ブロック取得数
DBWR チェックポイント	Cache	1 分あたりのチェックポイント数	db.Cache.DBWR チェックポイント
物理的な読み取り	Cache	1 秒あたりの読み取り数	db.Cache.physical 読み取り数
キャッシュからの一貫した取得数	Cache	1 秒あたりの取得数	db.Cache.consistent キャッシュからの取得数
キャッシュからの DB ブロックの取得数	Cache	1 秒あたりの取得数	db.Cache.db キャッシュからのブロック取得数
整合性のある取得数	Cache	1 秒あたりの取得数	db.Cache.consistent 取得数

Amazon RDS for PostgreSQL の Performance Insights カウンター

以下のデータベースカウンターは、Amazon RDS for PostgreSQL の Performance Insights で利用できます。

トピック

- Amazon RDS for PostgreSQL のネイティブカウンター
- Amazon RDS for PostgreSQL の非ネイティブカウンター

Amazon RDS for PostgreSQL のネイティブカウンター

ネイティブメトリクスは、Amazon RDS ではなく、データベースエンジンによって定義されます。これらのネイティブメトリクスの定義については、PostgreSQL の「統計情報の表示」を参照してください。

Counter	タイプ	単位	メトリクス
blks_hit	Cache	1 秒あたりのブロック数	db.Cache.blks_hit
buffers_alloc	Cache	1 秒あたりのブロック数	db.Cache.buffers_alloc
buffers_checkpoint	Checkpoint	1 秒あたりのブロック数	db.Checkpoint.buffers_checkpoint
checkpoint_sync_time	Checkpoint	チェックポイントあたりのミリ秒数	db.Checkpoint.checkpoint_sync_time
checkpoint_write_time	Checkpoint	チェックポイントあたりのミリ秒数	db.Checkpoint.checkpoint_write_time
checkpoints_req	Checkpoint	1 分あたりのチェックポイント数	db.Checkpoint.checkpoints_req
checkpoints_timed	Checkpoint	1 分あたりのチェックポイント数	db.Checkpoint.checkpoints_timed
maxwritten_clean	Checkpoint	1 分あたりの Bgwriter の完全停止数	db.Checkpoint.maxwritten_clean

Counter	タイプ	単位	メトリクス
deadlocks	Concurrency	1 分あたりのデッドロック数	db.Concurrency.deadlocks
blk_read_time	I/O	Milliseconds	db.IO.blk_read_time
blks_read	I/O	1 秒あたりのブロック数	db.IO.blks_read
buffers_backend	I/O	1 秒あたりのブロック数	db.IO.buffers_backend
buffers_backend_fsync	I/O	1 秒あたりのブロック数	db.IO.buffers_backend_fsync
buffers_clean	I/O	1 秒あたりのブロック数	db.IO.buffers_clean
tup_deleted	SQL	1 秒あたりのタプル数	db.SQL.tup_deleted
tup_fetched	SQL	1 秒あたりのタプル数	db.SQL.tup_fetched
tup_inserted	SQL	1 秒あたりのタプル数	db.SQL.tup_inserted
tup_returned	SQL	1 秒あたりのタプル数	db.SQL.tup_returned
tup_updated	SQL	1 秒あたりのタプル数	db.SQL.tup_updated
temp_bytes	Temp	1 秒あたりのバイト数	db.Temp.temp_bytes
temp_files	Temp	1 分あたりのファイル数	db.Temp.temp_files

Counter	タイプ	単位	メトリクス
xact_commit	トランザクション	1 秒あたりのコミット数	db.Transactions.xact_commit
xact_rollback	トランザクション	1 秒あたりのロールバック数	db.Transactions.xact_rollback
numbackends	ユーザー	接続	db.User.numbackends
total_auth_attempts	ユーザー	1 分あたりの接続数	db.User.total_auth_attempts
archived_count	ログ先行書き込み (WAL)	1 分あたりのファイル数	db.WAL.archived_count

Amazon RDS for PostgreSQL の非ネイティブカウンター

非ネイティブカウンターメトリクスは、Amazon RDS で定義されているカウンターです。非ネイティブメトリクスは、特定のクエリで取得するメトリクスである場合があります。非ネイティブメトリクスは派生メトリクスである場合もあります。この場合は、複数のネイティブカウンターが比率、ヒット率、またはレイテンシーの計算で使用されます。

Counter	タイプ	単位	メトリクス	説明	定義
checkpoint_sync_latency	Checkt	db.Checkpoint.checkpoint_sync_latency		チェックポイント処理でファイルをディスクに同期する部分に費やした合計時間。	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Checkt	db.Checkpoint.checkpoint_	チェックポイント処理でファイルをディスクに書き込む部分	`checkpoint_write_time /`	

Counter	タイプ	単位	メトリクス	説明	定義
		ite_late cy	に費やした合計時間。	(checkpoints_timed + checkpoints_req)	
read_latency	I/O	db.IO.r d_later	このインスタンスのバックエンドでデータファイルブロックの読み取りに費やした時間。	blk_read_time / blks_read	
idle_in_transaction_aborted_count	State	セッション	db.state.idle_in_transaction_aborted_count	idle in transaction (aborted) 状態のセッションの数。	該当しない
idle_in_transaction_count	State	セッション	db.state.idle_in_transaction_count	idle in transaction 状態のセッションの数。	該当しない
idle_in_transaction_max_time	State	秒	db.state.idle_in_transaction_max_time	idle in transaction 状態で実行されている最も長いトランザクションの時間を秒単位で表します。	該当しない
active_transactions	トランザクション	トランザクション	db.Transactions.active_transactions	アクティブなトランザクションの数。	該当しない

Counter	タイプ	単位	メトリクス	説明	定義
blocked_transactions	トランザクション	トランザクション	db.Transactions.blocked_transactions	ブロックされたトランザクションの数。	該当しない
oldest_active_logical_replication_slot_xid_age	取引	db.Transactions.dest_active_logl_replic tion_sl _xid_a	アクティブな論理レプリケーションスロット内の最も古いトランザクションの経過時間。 詳細については、「論理レプリケーションスロット」を参照してください。	–	
oldest_inactive_logical_replication_slot_xid_age	取引	db.Transactions.dest_inactive_local_rep cation_ ot_xid_ e	非アクティブな論理レプリケーションスロット内の最も古いトランザクションの経過時間。 詳細については、「論理レプリケーションスロット」を参照してください。	–	

Counter	タイプ	単位	メトリクス	説明	定義
oldest_prepared_transaction_xid_age	取引	db.Transactions.dest_prepared_transaction_id_age	最も古い準備済みトランザクションの経過時間。 詳細については、「準備済みトランザクション」を参照してください。	–	
oldest_running_transaction_xid_age	取引	db.Transactions.dest_running_transaction_id_age	最も古い実行中トランザクションの経過時間。 最も古い実行中のアクティブなトランザクションについては「アクティブなステートメント」を、最も古い実行中のアイドル状態のトランザクションについては「トランザクションでのアイドル状態」を参照してください。	–	

Counter	タイプ	単位	メトリクス	説明	定義
oldest_hot_standby_feedback_xid_age	取引	db.Transactions.oldest_hot_standby_feedback_xid_age	hot_standby_feedback が有効になっているリードレプリカで実行されている最も古いトランザクションの経過時間。 詳細については、「リードレプリカ」を参照してください。	-	
max_used_xact_ids	トランザクション	トランザクション	db.Transactions.max_used_xact_ids	バキューム処理されていないトランザクションの数。	該当しない
max_connections	Users	Conns	db.User.max_connections	max_connections パラメータで設定された DB インスタンスに許可される接続の最大数。	該当しない
total_auth_attempts	Users	1 分あたりの接続数	db.User.total_auth_attempts	このインスタンスへの接続試行の数	該当しない

Counter	タイプ	単位	メトリクス	説明	定義
archive_failed_count	WAL	1 分あたりのファイル数	db.WAL.archive_failed_count	WAL ファイルのアーカイブに失敗した 1 分あたりのファイル数。	該当しない

Performance Insights の SQL 統計

SQL 統計は、Performance Insights によって収集される SQL クエリに関するパフォーマンス関連のメトリックです。Performance Insights は、クエリが実行中の 1 秒ごとおよび SQL 呼び出しごとに統計を収集します。SQL 統計は、選択した時間範囲の平均です。

SQL ダイジェストは、特定のパターンを持つすべてのクエリの複合体ですが、必ずしも同じリテラル値を持つ必要はありません。ダイジェストは、リテラル値を疑問符に置き換えます。例えば、`SELECT * FROM emp WHERE lname= ?` と指定します。このダイジェストは、次の子クエリで構成されます。

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

すべてのエンジンは、ダイジェストクエリの SQL 統計をサポートしています。

この機能のリージョン、DB エンジン、およびインスタンスクラスのサポート情報については、「Amazon RDS DB エンジン、リージョン、およびインスタンスクラスでサポートされている Performance Insights 機能」を参照してください。

トピック

- MariaDB および MySQL のSQL統計
- Amazon RDS for Oracle の SQL 統計
- Amazon RDS for SQL Server の SQL 統計
- RDS PostgreSQL での SQL 統計

MariaDB および MySQL のSQL統計

MariaDB、および MySQLは、ダイジェストレベルでのみSQL 統計を収集します。ステートメントレベルでは、統計は表示されません。

トピック

- MariaDB および MySQL の Digest 統計
- MariaDB および MySQL の秒単位の統計データ
- MariaDB および MySQL の呼び出しごとの統計データ
- MariaDB および MySQL のプライマリ統計情報

MariaDB および MySQL の Digest 統計

Performance Insightsは、`events_statements_summary_by_digest` テーブルから SQL ダイジェスト統計を収集します。`events_statements_summary_by_digest`テーブルは、データベースによって管理されます。

ダイジェストテーブルには削除ポリシーはありません。テーブルがいっぱいになると、AWS マネジメントコンソール に次のメッセージが表示されます。

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

このような状況では、MariaDB および MySQL 、はSQL クエリを追跡しません。この問題に対処するため、Performance Insights は、次の条件の両方が満たされた場合に、ダイジェストテーブルを自動的に切り捨てます。

- テーブルがいっぱいの場合、
- Performance Insights は、Performance Schema を自動的に管理します。

  自動管理の場合、`performance_schema` パラメータを `0` に設定する必要があります。[Source (ソース)] を `user` に設定しないでください。Performance Insights がパフォーマンススキーマを自動的に管理していない場合は、Amazon RDS for MariaDB または MySQL における Performance Insights のPerformance Schema の概要 を参照してください。

AWS CLI で、describe-db-pameters コマンドを実行し、パラメータ値のソースをチェックします。

MariaDB および MySQL の秒単位の統計データ

次の SQL 統計は、MariaDB および MySQL DB インスタンスで使用できます。

メトリクス	Unit
db.sql_tokenized.stats.count_star_per_sec	1 秒あたりの呼び出し数
db.sql_tokenized.stats.sum_timer_wait_per_sec	1 秒あたりの平均レイテンシー (ミリ秒)
db.sql_tokenized.stats.sum_select_full_join_per_sec	1 秒ごとに完全結合を選択
db.sql_tokenized.stats.sum_select_range_check_per_sec	1 秒ごとに範囲チェックを選択
db.sql_tokenized.stats.sum_select_scan_per_sec	1 秒ごとにスキャンを選択
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	1 秒ごとにマージパスを並べ替え
db.sql_tokenized.stats.sum_sort_scan_per_sec	1 秒あたりの並べ替えスキャン数
db.sql_tokenized.stats.sum_sort_range_per_sec	1 秒ごとの並べ替え範囲
db.sql_tokenized.stats.sum_sort_rows_per_sec	1 秒あたりの行の並べ替え
db.sql_tokenized.stats.sum_rows_affected_per_sec	1 秒あたりの影響を受ける行数
db.sql_tokenized.stats.sum_rows_examined_per_sec	1 秒あたりの検査される行数
db.sql_tokenized.stats.sum_rows_sent_per_sec	1 秒あたりに送信される行数
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	1 秒ごとに作成されるテンポラリディスクテーブル

メトリクス	Unit
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	1 秒ごとに作成されるテンポラリテーブル
db.sql_tokenized.stats.sum_lock_time_per_sec	1 秒あたりのロック時間 (ミリ秒)

MariaDB および MySQL の呼び出しごとの統計データ

以下のメトリクスは、SQL ステートメントの呼び出しごとの統計を提供します。

メトリクス	単位
db.sql_tokenized.stats.sum_timer_wait_per_call	呼び出しごとの平均レイテンシー (ミリ秒)
db.sql_tokenized.stats.sum_select_full_join_per_call	コールごとに完全結合を選択
db.sql_tokenized.stats.sum_select_range_check_per_call	コールごとに範囲チェックを選択
db.sql_tokenized.stats.sum_select_scan_per_call	コールごとにスキャンを選択
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	コールごとにマージパスを並べ替え
db.sql_tokenized.stats.sum_sort_scan_per_call	コールごとに並べ替えスキャン
db.sql_tokenized.stats.sum_sort_range_per_call	コールごとの並べ替え範囲
db.sql_tokenized.stats.sum_sort_rows_per_call	呼び出しごとの行の並べ替え
db.sql_tokenized.stats.sum_rows_affected_per_call	コールごとに影響を受ける行数
db.sql_tokenized.stats.sum_rows_examined_per_call	コールごとに検査される行数

メトリクス	単位
db.sql_tokenized.stats.sum_rows_sent_per_call	コールごとに送信される行数
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	コールごとに作成されたテンポラリディスクテーブル数
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	コールごとに作成されたテンポラリテーブル数
db.sql_tokenized.stats.sum_lock_time_per_call	呼び出しごとのロック時間 (ミリ秒)

MariaDB および MySQL のプライマリ統計情報

次の SQL 統計は、MariaDB および MySQL DB インスタンスで使用できます。

メトリクス	単位
db.sql_tokenized.stats.count_star	通話
db.sql_tokenized.stats.sum_timer_wait	待機時間 (ミリ秒単位)
db.sql_tokenized.stats.sum_select_full_join	完全結合を選択する
db.sql_tokenized.stats.sum_select_range_check	範囲チェックを選択する
db.sql_tokenized.stats.sum_select_scan	スキャンの選択
db.sql_tokenized.stats.sum_sort_merge_passes	マージパスの並べ替え
db.sql_tokenized.stats.sum_sort_scan	スキャンの並べ替え
db.sql_tokenized.stats.sum_sort_range	並べ替え範囲
db.sql_tokenized.stats.sum_sort_rows	行のソート
db.sql_tokenized.stats.sum_rows_affected	影響を受けた行
db.sql_tokenized.stats.sum_rows_examined	検査された行

メトリクス	単位
db.sql_tokenized.stats.sum_rows_sent	送信された行
db.sql_tokenized.stats.sum_created_tmp_disk_tables	作成されたテンポラリディスクテーブル数
db.sql_tokenized.stats.sum_created_tmp_tables	作成されたテンポラリテーブル
db.sql_tokenized.stats.sum_lock_time	ロック時間 (ミリ秒単位)

Amazon RDS for Oracle の SQL 統計

Amazon RDS for Oracle は、ステートメントレベルとダイジェストレベルの両方で SQL 統計を収集します。ステートメントレベルでは、ID 列は`V$SQL.SQL_ID`の値を表します。ダイジェストレベルでは、ID 列には`V$SQL.FORCE_MATCHING_SIGNATURE`の値が表示されます。

ダイジェストレベルで ID が`0`の場合、Oracle データベースはこのステートメントが再利用に適していないと判断しました。この場合、子の SQL ステートメントは異なるダイジェストに属している可能性があります。ただし、ステートメントは初期に収集された SQL ステートメントに関する `digest_text` の中にグループ化されています。

トピック

- Oracle の秒単位の統計
- Oracleのコールごとの統計情報
- Oracle のプライマリ統計

Oracle の秒単位の統計

次のメトリクスは、Oracle SQL クエリの秒単位の統計を提供します。

メトリクス	単位
db.sql.stats.executions_per_sec	1 秒あたりの実行回数
db.sql.stats.elapsed_time_per_sec	平均アクティブ実行 (AAE)
db.sql.stats.rows_processed_per_sec	1 秒あたりに処理される行

メトリクス	単位
db.sql.stats.buffer_gets_per_sec	1 秒あたりのバッファ取得数
db.sql.stats.physical_read_requests_per_sec	1 秒あたりの物理的な読み取り数
db.sql.stats.physical_write_requests_per_sec	1 秒あたりの物理的な書き込み数
db.sql.stats.total_sharable_mem_per_sec	1 秒あたりの共有可能なメモリの合計数 (バイト単位)
db.sql.stats.cpu_time_per_sec	1 秒あたりの CPU 時間 (ミリ秒)

次のメトリクスは、Oracle SQL ダイジェストクエリの秒単位の統計を提供します。

メトリクス	単位
db.sql_tokenized.stats.executions_per_sec	1 秒あたりの実行回数
db.sql_tokenized.stats.elapsed_time_per_sec	平均アクティブ実行 (AAE)
db.sql_tokenized.stats.rows_processed_per_sec	1 秒あたりに処理される行
db.sql_tokenized.stats.buffer_gets_per_sec	1 秒あたりのバッファ取得数
db.sql_tokenized.stats.physical_read_requests_per_sec	1 秒あたりの物理的な読み取り数
db.sql_tokenized.stats.physical_write_requests_per_sec	1 秒あたりの物理的な書き込み数
db.sql_tokenized.stats.total_sharable_mem_per_sec	1 秒あたりの共有可能なメモリの合計数 (バイト単位)
db.sql_tokenized.stats.cpu_time_per_sec	1 秒あたりの CPU 時間 (ミリ秒)

Oracleのコールごとの統計情報

以下のメトリクスは、Oracle SQL ステートメントの呼び出しごとの統計を提供します。

メトリクス	単位
db.sql.stats.elapsed_time_per_exec	実行ごとの経過時間(ミリ秒)
db.sql.stats.rows_processed_per_exec	実行ごとに処理される行
db.sql.stats.buffer_gets_per_exec	実行ごとのバッファ取得数
db.sql.stats.physical_read_requests_per_exec	実行ごとの物理的読み取り数
db.sql.stats.physical_write_requests_per_exec	実行ごとの物理的書き込み数
db.sql.stats.total_sharable_mem_per_exec	実行ごとの共有可能なメモリの合計(バイト単位)
db.sql.stats.cpu_time_per_exec	実行ごとの CPU 時間(ミリ秒)

以下のメトリクスは、Oracle SQL ダイジェストクエリの呼び出しごとの統計を提供します。

メトリクス	単位
db.sql_tokenized.stats.elapsed_time_per_exec	実行ごとの経過時間(ミリ秒)
db.sql_tokenized.stats.rows_processed_per_exec	実行ごとに処理される行
db.sql_tokenized.stats.buffer_gets_per_exec	実行ごとのバッファ取得数
db.sql_tokenized.stats.physical_read_requests_per_exec	実行ごとの物理的読み取り数
db.sql_tokenized.stats.physical_write_requests_per_exec	実行ごとの物理的書き込み数
db.sql_tokenized.stats.total_sharable_mem_per_exec	実行ごとの共有可能なメモリの合計(バイト単位)

メトリクス	単位
db.sql_tokenized.stats.cpu_time_per_exec	実行ごとの CPU 時間(ミリ秒)

Oracle のプライマリ統計

次のメトリクスは、Oracle SQL クエリのプライマリ統計を提供します。

メトリクス	単位
db.sql.stats.executions	実行の回数
db.sql.stats.elapsed_time	経過時間 (ミリ秒単位)
db.sql.stats.rows_processed	処理された行
db.sql.stats.buffer_gets	バッファ取得数
db.sql.stats.physical_read_requests	物理的な読み取り
db.sql.stats.physical_write_requests	物理的な書き込み数
db.sql.stats.total_sharable_mem	共有可能なメモリの合計 (バイト単位)
db.sql.stats.cpu_time	CPU 時間 (ミリ秒単位)

次のメトリクスは、Oracle SQL ダイジェストクエリのプライマリ統計を提供します。

メトリクス	単位
db.sql_tokenized.stats.executions	実行の回数
db.sql_tokenized.stats.elapsed_time	経過時間 (ミリ秒単位)
db.sql_tokenized.stats.rows_processed	処理された行
db.sql_tokenized.stats.buffer_gets	バッファ取得数
db.sql_tokenized.stats.physical_read_requests	物理的な読み取り

メトリクス	単位
db.sql_tokenized.stats.physical_write_requests	物理的な書き込み数
db.sql_tokenized.stats.total_sharable_mem	共有可能なメモリの合計 (バイト単位)
db.sql_tokenized.stats.cpu_time	CPU 時間 (ミリ秒単位)

Amazon RDS for SQL Server の SQL 統計

Amazon RDS for SQL Server は、ステートメントレベルとダイジェストレベルの両方で SQL 統計を収集します。ステートメントレベルの場合、ID 列は `sql_handle` の値を表します。ダイジェストレベルの場合、ID 列は `query_hash` の値を示します。

SQL Server はいくつかのステートメントで `query_hash` として NULL 値を返します。例えば、ALTER INDEX、CHECKPOINT、UPDATE STATISTICS、COMMIT TRANSACTION、FETCH NEXT FROM Cursor、およびいくつかの INSERT ステートメント、SELECT @<variable>、条件ステートメント、実行可能なストアドプロシージャが該当します。この場合、`sql_handle` 値は、そのステートメントのダイジェストレベルで ID として表示されます。

トピック

- SQL Server の秒単位の統計
- SQL Server の呼び出し単位の統計
- SQL Server のプライマリ統計

SQL Server の秒単位の統計

以下のメトリクスは、SQL Server の SQL クエリの秒単位の統計を提供します。

メトリクス	単位
db.sql.stats.execution_count_per_sec	1 秒あたりの実行回数
db.sql.stats.total_elapsed_time_per_sec	1 秒あたりの合計経過時間
db.sql.stats.total_rows_per_sec	1 秒あたりの処理された行の合計数
db.sql.stats.total_logical_reads_per_sec	1 秒あたりの論理読み取りの合計数

メトリクス	単位
db.sql.stats.total_logical_writes_per_sec	1 秒あたりの論理書き込みの合計数
db.sql.stats.total_physical_reads_per_sec	1 秒あたりの物理読み取りの合計数
db.sql.stats.total_worker_time_per_sec	合計 CPU 時間 (ミリ秒単位)

以下のメトリクスは、SQL Server の SQL ダイジェストクエリの秒単位の統計を示します。

メトリクス	単位
db.sql_tokenized.stats.execution_count_per_sec	1 秒あたりの実行数
db.sql_tokenized.stats.total_elapsed_time_per_sec	1 秒あたりの合計経過時間
db.sql_tokenized.stats.total_rows_per_sec	1 秒あたりの処理された行の合計数
db.sql_tokenized.stats.total_logical_reads_per_sec	1 秒あたりの論理読み取りの合計数
db.sql_tokenized.stats.total_logical_writes_per_sec	1 秒あたりの論理書き込みの合計数
db.sql_tokenized.stats.total_physical_reads_per_sec	1 秒あたりの物理読み取りの合計数
db.sql_tokenized.stats.total_worker_time_per_sec	合計 CPU 時間 (ミリ秒単位)

SQL Server の呼び出し単位の統計

以下のメトリクスは、SQL Server の SQL ステートメントの呼び出しあたりの統計を示します。

メトリクス	単位
db.sql.stats.total_elapsed_time_per_call	実行あたりの合計経過時間 (ミリ秒)
db.sql.stats.total_rows_per_call	実行あたりの処理された行の合計数
db.sql.stats.total_logical_reads_per_call	実行あたりの論理読み取りの合計数
db.sql.stats.total_logical_writes_per_call	実行あたりの論理書き込みの合計数
db.sql.stats.total_physical_reads_per_call	実行あたりの物理読み取りの合計数
db.sql.stats.total_worker_time_per_call	実行あたりの合計 CPU 時間 (ミリ秒単位)

以下のメトリクスは、SQL Server の SQLダイジェストクエリの呼び出しあたりの統計を示します。

メトリクス	単位
db.sql_tokenized.stats.total_elapsed_time_per_call	実行あたりの合計経過時間
db.sql_tokenized.stats.total_rows_per_call	実行あたりの処理された行の合計数
db.sql_tokenized.stats.total_logical_reads_per_call	実行あたりの論理読み取りの合計数
db.sql_tokenized.stats.total_logical_writes_per_call	実行あたりの論理書き込みの合計数
db.sql_tokenized.stats.total_physical_reads_per_call	実行あたりの物理読み取りの合計数
db.sql_tokenized.stats.total_worker_time_per_call	実行あたりの合計 CPU 時間 (ミリ秒単位)

SQL Server のプライマリ統計

以下のメトリクスは、SQL Server の SQL クエリのプライマリ統計を示します。

メトリクス	単位
db.sql.stats.execution_count	実行の回数
db.sql.stats.total_elapsed_time	合計経過時間 (ミリ秒単位)
db.sql.stats.total_rows	処理された行の合計数
db.sql.stats.total_logical_reads	論理読み取りの合計数
db.sql.stats.total_logical_writes	論理書き込みの合計数
db.sql.stats.total_physical_reads	物理読み取りの合計数
db.sql.stats.total_worker_time	合計 CPU 時間 (ミリ秒単位)

以下のメトリクスは、SQL Server の SQLダイジェストクエリのプライマリ統計を示します。

メトリクス	単位
db.sql_tokenized.stats.execution_count	実行の回数
db.sql_tokenized.stats.total_elapsed_time	合計経過時間 (ミリ秒単位)
db.sql_tokenized.stats.total_rows	処理された行の合計数
db.sql_tokenized.stats.total_logical_reads	論理読み取りの合計数
db.sql_tokenized.stats.total_logical_writes	論理書き込みの合計数
db.sql_tokenized.stats.total_physical_reads	物理読み取りの合計数
db.sql_tokenized.stats.total_worker_time	合計 CPU 時間 (ミリ秒単位)

RDS PostgreSQL での SQL 統計

Performance Insights は、SQL 呼び出しごとおよびクエリが実行中の 1 秒ごとに SQL 統計を収集します。RDS for PostgreSQL は SQL 統計をダイジェストレベルでのみ収集します。ステートメントレベルでは、統計は表示されません。

RDS for PostgreSQL のダイジェストレベルの統計の詳細については、以下を参照してください。

トピック

- RDS PostgreSQL でのダイジェスト統計
- RDS PostgreSQL での秒単位のダイジェスト統計
- RDS PostgreSQL のコールごとのダイジェスト統計
- RDS PostgreSQL でのプライマリ統計

RDS PostgreSQL でのダイジェスト統計

SQL ダイジェストの統計を表示するには、RDS PostgreSQL が `pg_stat_statements` ライブラリをロードする必要があります。PostgreSQL 11 以降と互換性のある PostgreSQL DB インスタンスでは、このライブラリはデフォルトでロードされます。PostgreSQL 10 以前と互換性のある PostgreSQL DB インスタンスでは、このライブラリをマニュアルで有効にします。手動で有効にするには、DB インスタンスに関連付けられた DB パラメータグループの `shared_preload_libraries` に `pg_stat_statements` を追加します。次に DB インスタンスを再起動します。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

> **Note**
>
> Performance Insights は、切り捨てられない `pg_stat_activity` 内のクエリの統計のみを収集できます。デフォルトでは、PostgreSQLデータベースは4,096バイトより長い問い合わせを切り捨てます。問い合わせサイズを増やすには、DB インスタンスに関連付けられた DB パラメータグループの `track_activity_query_size` パラメータを変更します。このパラメータを変更した場合は、DB インスタンスの再起動が必要です。

RDS PostgreSQL での秒単位のダイジェスト統計

PostgreSQL DB インスタンスでは、次の SQL ダイジェストの統計を使用できます。

メトリクス	単位
db.sql_tokenized.stats.calls_per_sec	1 秒あたりの呼び出し数
db.sql_tokenized.stats.rows_per_sec	1 秒あたりの行数
db.sql_tokenized.stats.total_time_per_sec	1 秒あたりの平均アクティブ実行 (AAE)

メトリクス	単位
db.sql_tokenized.stats.shared_blks_hit_per_sec	1 秒あたりのブロックヒット数
db.sql_tokenized.stats.shared_blks_read_per_sec	1 秒あたりのブロック読み取り数
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	1 秒あたりのダーティになったブロック数
db.sql_tokenized.stats.shared_blks_written_per_sec	1 秒あたりのブロック書き込み数
db.sql_tokenized.stats.local_blks_hit_per_sec	1 秒あたりのローカルブロックヒット数
db.sql_tokenized.stats.local_blks_read_per_sec	1 秒あたりのローカルブロック読み取り数
db.sql_tokenized.stats.local_blks_dirtied_per_sec	1 秒あたりのダーティになったローカルブロック数
db.sql_tokenized.stats.local_blks_written_per_sec	1 秒あたりのローカルブロック書き込み数
db.sql_tokenized.stats.temp_blks_written_per_sec	1 秒あたりの一時的な書き込み数
db.sql_tokenized.stats.temp_blks_read_per_sec	1 秒あたりの一時的な読み取り数
db.sql_tokenized.stats.blk_read_time_per_sec	1 秒あたりの平均的な同時読み取り数
db.sql_tokenized.stats.blk_write_time_per_sec	1 秒あたりの平均的な同時書き込み数

RDS PostgreSQL のコールごとのダイジェスト統計

以下のメトリクスは、SQL ステートメントの呼び出しごとの統計を提供します。

メトリクス	単位
db.sql_tokenized.stats.rows_per_call	呼び出しごとの行数

メトリクス	単位
db.sql_tokenized.stats.avg_latency_per_call	呼び出しごとの平均レイテンシー (ミリ秒)
db.sql_tokenized.stats.shared_blks_hit_per_call	呼び出しごとのブロックヒット数
db.sql_tokenized.stats.shared_blks_read_per_call	呼び出しごとのブロック読み取り数
db.sql_tokenized.stats.shared_blks_written_per_call	呼び出しごとのブロック書き込み数
db.sql_tokenized.stats.shared_blks_dirtied_per_call	呼び出しあたりのダーティになったブロック数
db.sql_tokenized.stats.local_blks_hit_per_call	呼び出しあたりのローカルブロックヒット数
db.sql_tokenized.stats.local_blks_read_per_call	呼び出しあたりのローカルブロック読み取り数
db.sql_tokenized.stats.local_blks_dirtied_per_call	呼び出しあたりのダーティになったローカルブロック数
db.sql_tokenized.stats.local_blks_written_per_call	呼び出しあたりのローカルブロック書き込み数
db.sql_tokenized.stats.temp_blks_written_per_call	呼び出しあたりのテンポラリブロック書き込み数
db.sql_tokenized.stats.temp_blks_read_per_call	呼び出しあたりのテンポラリブロック読み取り数
db.sql_tokenized.stats.blk_read_time_per_call	呼び出しごとの読み取り時間 (ミリ秒)
db.sql_tokenized.stats.blk_write_time_per_call	呼び出しごとの書き込み時間 (ミリ秒)

RDS PostgreSQL でのプライマリ統計

PostgreSQL DB インスタンスでは、次の SQL の統計を使用できます。

メトリクス	単位
db.sql_tokenized.stats.calls	通話
db.sql_tokenized.stats.rows	Rows
db.sql_tokenized.stats.total_time	合計時間 (ミリ秒単位)
db.sql_tokenized.stats.shared_blks_hit	ブロックヒット
db.sql_tokenized.stats.shared_blks_read	ブロック読み取り数
db.sql_tokenized.stats.shared_blks_dirtied	ダーティになったブロック数
db.sql_tokenized.stats.shared_blks_written	ブロック書き込み数
db.sql_tokenized.stats.local_blks_hit	ローカルブロックヒット数
db.sql_tokenized.stats.local_blks_read	ロジカルブロックヒット数
db.sql_tokenized.stats.local_blks_dirtied	ダーティになったローカルブロック
db.sql_tokenized.stats.local_blks_written	ローカルブロック書き込み数
db.sql_tokenized.stats.temp_blks_written	一時的な書き込み数
db.sql_tokenized.stats.temp_blks_read	一時的な読み取り数
db.sql_tokenized.stats.blk_read_time	平均的な同時読み取り数 (ミリ秒単位)
db.sql_tokenized.stats.blk_write_time	平均的な同時書き込み数 (ミリ秒単位)

これらのメトリクスの詳細については、PostgreSQL ドキュメントの「pg_stat_statements」を参照してください。

拡張モニタリングの OS メトリクス

Amazon RDS には、DB インスタンスが実行されているオペレーティングシステム (OS) のリアルタイムのメトリクスが用意されています。RDS は、拡張モニタリングのメトリクスを Amazon

CloudWatch Logs アカウントに配信します。以下の表では、Amazon CloudWatch Logs で使用できる OS メトリクスを示しています。

トピック

- Db2、MariaDB、MySQL、Oracle、および PostgreSQL の OS メトリクス
- Microsoft SQL Server の OS メトリクス

Db2、MariaDB、MySQL、Oracle、および PostgreSQL の OS メトリクス

RDS は、rdsdbdata のディスクメトリクスと、OS に存在する追加のストレージボリュームを収集します。CloudWatch Logs で OS メトリクスを表示するには、「CloudWatch Logs を使用した OS メトリクスの表示」を参照してください。

Group	メトリクス	コンソール名	説明
General	engine	該当しません	DB インスタンスのデータベースエンジン。
	instanceID	該当しません	DB インスタンス識別子。
	instanceResourceID	該当しません	AWS リージョンに固有の DB インスタンスの不変の識別子。ログストリーム識別子としても使用されます。
	numVCPUs	該当しません	DB インスタンスの仮想 CPU の数。
	timestamp	該当しません	メトリクスが作成された時間。
	uptime	該当しません	DB インスタンスがアクティブになってからの経過時間。
	version	該当しません	OS メトリクスのストリーム JSON 形式のバージョン。

Group	メトリクス	コンソール名	説明
cpuUtilization	guest	CPU ゲスト	ゲストプログラムが使用中の CPU の使用率。
	idle	CPU アイドル	アイドル状態の CPU の使用率。
	irq	CPU IRQ	ソフトウェア割り込みが使用中の CPU の使用率。
	nice	CPU Nice	最も低い優先順位で実行されているプログラムが使用中の CPU の使用率。
	steal	CPU Steal	他の仮想マシンが使用中の CPU の使用率。
	system	CPU システム	カーネルが使用中の CPU の使用率。
	total	CPU 合計	使用中の CPU の合計使用率。この値は nice 値を含みます。
	user	CPU ユーザー	ユーザープログラムが使用中の CPU の使用率。
	wait	CPU 待機	I/O アクセスを待機中の CPU の未使用率。
diskIO	avgQueueLen	平均キューサイズ	I/O デバイスのキューで待機中のリクエストの数。
	avgReqSz	平均リクエストサイズ	平均リクエストサイズ (キロバイト単位)。
	await	ディスク I/O 待機	リクエストへの応答に必要なミリ秒数 (キュー時間とサービス時間を含む)。
	device	該当しません	使用中のディスクデバイスの識別子。

Group	メトリクス	コンソール名	説明
	readIOsPS	読み取り I/O/秒	読み取りオペレーションの 1 秒あたりの数。
	readKb	読み取り合計	読み取りの合計キロバイト数。
	readKbPS	読み取り KB/秒	読み取りの 1 秒あたりのキロバイト数。
	readLatency	読み取りレイテンシー	読み取り I/O リクエスト送信からその完了までの経過時間 (ミリ秒単位) 。 このメトリクスは Amazon Aurora にのみ使用できます。
	readThroughput	読み取りスループット	DB クラスターへのリクエストによって使用されるネットワークスループットの量 (バイト/秒単位)。 このメトリクスは Amazon Aurora にのみ使用できます。
	rrqmPS	Rrqms	キューに入れられてマージされた読み取りリクエストの 1 秒あたりの数。
	tps	TPS	I/O トランザクションの 1 秒あたりの数。
	util	ディスク I/O 使用率	リクエスト発行中の CPU 時間の消費率。
	writeIOsPS	書き込み IO/秒	書き込みオペレーションの 1 秒あたりの数。
	writeKb	書き込み合計	書き込みの合計キロバイト数。
	writeKbPS	書き込み KB/秒	書き込みの 1 秒あたりのキロバイト数。

Group	メトリクス	コンソール名	説明
	writeLatency	書き込みレイテンシー	書き込み I/O リクエスト送信から完了までの平均経過時間 (ミリ秒単位) 。 このメトリクスは Amazon Aurora にのみ使用できます。
	writeThroughput	書き込みスループット	DB クラスターからのレスポンスによって使用されるネットワークスループットの量 (バイト/秒単位)。 このメトリクスは Amazon Aurora にのみ使用できます。
	wrqmPS	Wrqms	キューに入れられてマージされた書き込みリクエストの 1 秒あたりの数。
physicalDeviceIO	avgQueueLen	Physical Devices Avg Queue Size (物理デバイスの平均キューサイズ	I/O デバイスのキューで待機中のリクエストの数。
	avgReqSz	Physical Devices Ave Request Size (物理デバイスの平均リクエストサイズ	平均リクエストサイズ (キロバイト単位)。

Group	メトリクス	コンソール名	説明
	await	Physical Devices Disk I/O Await (物理デバイスディスク I/O 待機	リクエストへの応答に必要なミリ秒数 (キュー時間とサービス時間を含む)。
	device	該当しません	使用中のディスクデバイスの識別子。
	mountPoint	該当しない	ファイルシステムへのパス。
	readIOsPS	Physical Devices Read IO/s (物理デバイスの読み取り I/O/秒	読み取りオペレーションの 1 秒あたりの数。
	readKb	Physical Devices Read Total (物理デバイスの読み取り合計	読み取りの合計キロバイト数。
	readKbPS	Physical Devices Read Kb/s (物理デバイスの読み取り KB/秒	読み取りの 1 秒あたりのキロバイト数。

Group	メトリクス	コンソール名	説明
	rrqmPS	Physical Devices Rrqms (物理デバイス Rrqms	キューに入れられてマージされた読み取りリクエストの 1 秒あたりの数。
	tps	Physical Devices TPS (物理デバイス TPS	I/O トランザクションの 1 秒あたりの数。
	util	Physical Devices Disk I/O Util (物理デバイスディスク I/O ユーティリティ	リクエスト発行中の CPU 時間の消費率。
	writeIOsPS	Physical Devices Write IO/s (物理デバイスの書き込み IO/秒	書き込みオペレーションの 1 秒あたりの数。
	writeKb	Physical Devices Write Total (物理デバイスの書き込み合計	書き込みの合計キロバイト数。

Group	メトリクス	コンソール名	説明
	writeKbPS	Physical Devices Write Kb/s (物理デバイス書き込み KB/秒	書き込みの 1 秒あたりのキロバイト数。
	wrqmPS	Physical Devices Wrqms (物理デバイス Wrqms	キューに入れられてマージされた書き込みリクエストの 1 秒あたりの数。
fileSys	maxFiles	最大 i ノード	ファイルシステム用に作成できるファイルの最大数。
	mountPoint	該当しません	ファイルシステムへのパス。これが /rdsdbdata* の場合、すべてのストレージボリュームの集計を表します。
	name	該当しない	ファイルシステムの名前。
	total	合計ファイルシステム	ファイルシステムに使用できるディスク領域の合計量 (キロバイト単位)。
	used	使用済みファイルシステム	ファイルシステム内のファイルが使用中のディスク領域の量 (キロバイト単位)。
	usedFilePercent	使用済み i ノード	使用中のファイルの割合。
	usedFiles	使用済み (%)	ファイルシステム内のファイルの数。

Group	メトリクス	コンソール名	説明
	usedPercent	使用済みファイルシステム	ファイルシステムが使用中のディスク領域の割合。
loadAverageMinute	fifteen	平均負荷 (15 分	過去 15 分間に CPU 時間をリクエストしたプロセスの数。
	five	平均負荷 (5 分	過去 5 分間に CPU 時間をリクエストしたプロセスの数。
	one	平均負荷 (1 分	過去 1 分間に CPU 時間をリクエストしたプロセスの数。
memory	active	アクティブなメモリ	割り当てられたメモリの量 (キロバイト単位)。
	buffers	バッファ済みメモリ	ストレージデバイスへの書き込み前に I/O バッファリングリクエストに使用されたメモリの量 (キロバイト単位)。
	cached	キャッシュ済みメモリ	ファイルシステムベースの I/O のキャッシュに使用されたメモリの量。
	dirty	ダーティメモリ	変更されたがストレージ内のその関連データブロックに書き込まれなかった RAM 内のメモリページの量 (キロバイト単位)。
	free	空きメモリ	未割り当てのメモリの量 (キロバイト単位)。
	hugePagesFree	huge ページ (空き	空き huge ページの数。huge ページは Linux カーネルの機能です。
	hugePagesRsvd	huge ページ (予約	コミットされた huge ページの数。

Group	メトリクス	コンソール名	説明
	hugePagesSize	huge ページサイズ	各 huge ページユニットのサイズ (キロバイト単位)。
	hugePagesSurp	huge ページ (余剰	使用可能な huge ページの余剰数/合計数。
	hugePagesTotal	huge ページ (合計	huge ページの合計数。
	inactive	非アクティブメモリ	最も使用されていないメモリページの量 (キロバイト単位)。
	mapped	マップ済みメモリ	プロセスアドレス空間内でメモリマップされているファイルシステムの内容の合計量 (キロバイト単位)。
	pageTables	ページテーブル	ページテーブルが使用中のメモリの量 (キロバイト単位)。
	slab	スラブメモリ	再利用可能なカーネルデータ構造体の量 (キロバイト単位)。
	total	合計メモリ	メモリの合計量 (キロバイト単位)。
	writeback	書き戻しメモリ	バックアップストレージにまだ書き込み中の RAM 内のダーティページの量 (キロバイト単位)。
network	interface	該当しません	DB インスタンスに使用中のネットワークインターフェイスの識別子。
	rx	RX	1 秒あたりの受信バイト数。
	tx	TX	1 秒あたりのアップロードバイト数。
processList	cpuUsedPc	CPU %	プロセスが使用中の CPU の使用率。

Group	メトリクス	コンソール名	説明
	id	該当しません	プロセスの識別子。
	memoryUsedPc	MEM%	プロセスが使用中のメモリの使用率。
	name	該当しません	プロセスの名前。
	parentID	該当しません	プロセスの親プロセスの識別子。
	rss	RES	プロセスに割り当てられた RAM の量 (キロバイト単位)。
	tgid	該当しません	スレッドグループ識別子 (スレッドが属するプロセス ID を表す番号)。この識別子は同じプロセスのグループスレッドに使用されます。
	vss	VIRT	プロセスに割り当てられた仮想メモリの量 (キロバイト単位)。
swap	total	スワップ	使用可能なスワップメモリの量 (キロバイト単位)。
	in	スワップイン	ディスクからスワップされたメモリの量 (キロバイト単位)。
	out	スワップアウト	ディスクにスワップされたメモリの量 (キロバイト単位)。
	free	空きスワップ	空きスワップメモリの量 (キロバイト単位)。
	cached	コミット済みスワップ	キャッシュメモリとして使用されたスワップメモリの量 (キロバイト単位)。

Group	メトリクス	コンソール名	説明
tasks	blocked	ブロック済みタスク	ブロックされているタスクの数。
	running	実行中のタスク	実行中のタスクの数。
	sleeping	スリープ中のタスク	スリープ中のタスクの数。
	stopped	停止済みタスク	停止中のタスクの数。
	total	タスク合計	タスクの合計数。
	zombie	Zombie タスク	アクティブな親タスクの非アクティブな子タスクの数。

Microsoft SQL Server の OS メトリクス

RDS は、rdsdbdata のディスクメトリクスと、OS に存在する追加のストレージボリュームを収集します。CloudWatch Logs で OS メトリクスを表示するには、「CloudWatch Logs を使用した OS メトリクスの表示」を参照してください。

Group	メトリクス	コンソール名	説明
General	engine	該当しません	DB インスタンスのデータベースエンジン。
	instanceID	該当しません	DB インスタンス識別子。
	instanceResourceID	該当しません	AWS リージョンに固有の DB インスタンスの不変の識別子。ログストリーム識別子としても使用されます。
	numVCPUs	該当しません	DB インスタンスの仮想 CPU の数。

Group	メトリクス	コンソール名	説明
	timestamp	該当しません	メトリクスが作成された時間。
	uptime	該当しません	DB インスタンスがアクティブになってからの経過時間。
	version	該当しません	OS メトリクスのストリーム JSON 形式のバージョン。
cpuUtilization	idle	CPU アイドル	アイドル状態の CPU の使用率。
	kern	CPU カーネル	カーネルが使用中の CPU の使用率。
	user	CPU ユーザー	ユーザープログラムが使用中の CPU の使用率。
disks	name	該当しません	ディスクの識別子。
	totalKb	合計ディスク容量	ディスクの合計容量 (キロバイト単位)。
	usedKb	使用済みディスク領域	ディスクで使用されている容量 (キロバイト単位)。
	usedPc	使用済みディスク領域 (%)	ディスクで使用されている容量の割合。
	availKb	空きディスク領域	ディスクの空き領域 (キロバイト単位)。
	availPc	空きディスク領域 (%)	ディスクで使用可能な容量の割合。
	rdCountPS	読み取り/秒	読み取りオペレーションの 1 秒あたりの数。
	rdBytesPS	読み取り KB/秒	読み取りの 1 秒あたりのバイト数。

Group	メトリクス	コンソール名	説明
	wrCountPS	書き込み IO/秒	書き込みオペレーションの 1 秒あたりの数。
	wrBytesPS	書き込み KB/秒	1 秒あたりに書き込まれるバイト数。
memory	commitTotKb	コミット合計	使用中のページファイルバックド仮想アドレス容量は、結果として現在のコミットチャージとなります。この値は、メインメモリ (RAM) とディスク (ページファイル) の合計になります。
	commitLimitKb	最大コミット	commitTotKb メトリックの最大有効値。この値は、現在のページファイルサイズに、ページングできない領域に割り当てられた RAM を除くページング可能コンテンツに使用可能な物理メモリを加えた合計値になります。
	commitPeakKb	コミットピーク	オペレーティングシステムが最後に起動された後の commitTotKb メトリックの最大値。
	kernTotKb	合計カーネルメモリ	ページングされたカーネルプールとページングされないカーネルプールのメモリの合計 (キロバイト単位)。
	kernPagedKb	ページングカーネルメモリ	ページングカーネルプールのメモリの量 (キロバイト単位)。
	kernNonpagedKb	非ページングカーネルメモリ	非ページングカーネルプールのメモリの量 (キロバイト単位)。
	pageSize	ページサイズ	ページのサイズ (バイト単位)。
	physTotKb	合計メモリ	物理メモリの量 (キロバイト単位)。

Group	メトリクス	コンソール名	説明
	physAvailKb	利用可能なメモリ	使用可能な物理メモリの量 (キロバイト単位)。
	sqlServerTotKb	SQL サーバー合計メモリ	Microsoft SQL Server にコミットされたメモリの量 (キロバイト単位)。
	sysCacheKb	システムキャッシュ	システムキャッシュメモリの量 (キロバイト単位)。
network	interface	該当しません	DB インスタンスに使用中のネットワークインターフェイスの識別子。
	rdBytesPS	ネットワーク読み取り KB/秒	1 秒あたりの受信バイト数。
	wrBytesPS	ネットワーク書き込み KB/秒	1 秒あたりの送信バイト数。
processList	cpuUsedPc	使用済み (%)	プロセスが使用中の CPU の使用率。
	memUsedPc	MEM%	プロセスが使用中のメモリの合計使用率。
	name	該当しません	プロセスの名前。
	pid	該当しません	プロセスの識別子。この値は、Amazon RDS が所有するプロセスでは表示されません。
	ppid	該当しません	このプロセスの親プロセスの識別子。この値は子プロセスについてのみ表示されます。
	tid	該当しません	スレッド識別子。この値はスレッドについてのみ表示されます。所有プロセスは、pid 値を使用することにより識別できます。

Group	メトリクス	コンソール名	説明
	workingSetKb	該当しません	プライベート作業セットのメモリ量に、プロセスで使用中で他のプロセスと共有できるのメモリ量を加えたメモリ量。
	workingSetPrivKb	該当しません	プロセスで使用中で他のプロセスと共有できないメモリ量 (キロバイト単位)。
	workingSetShareableKb	該当しません	プロセスで使用中で他のプロセスと共有できるメモリ量 (キロバイト単位)。
	virtKb	該当しません	プロセスが使用している仮想アドレス容量 (キロバイト単位)。仮想アドレススペースの使用は、必ずしもディスクまたはメインメモリページのいずれかが対応して使用されていることを意味しません。
system	handles	ハンドル	システムが使用しているハンドル数。
	processes	プロセス	システムで実行されているプロセス数。
	threads	スレッド	システムで実行されているスレッド数。

Amazon RDS DB インスタンスでの、イベント、ログ、およびストリーミングのモニタリング

Amazon RDS データベースおよびその他の AWS ソリューションをモニタリングする場合、目標は次の条件を維持することです。

- 信頼性
- 利用可能な状況
- パフォーマンス
- セキュリティ

「Amazon RDS インスタンスでのメトリクスのモニタリング」では、メトリクスを使用してインスタンスをモニタリングする方法について説明します。完全なソリューションでは、データベースイベント、ログファイル、およびアクティビティストリーミングもモニタリングする必要があります。AWS には、次のモニタリングツールが用意されています。

- Amazon EventBridge は、アプリケーションをさまざまなソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、お客様独自のアプリケーション、Software-as-a-Service (SaaS) アプリケーション、および AWS サービスから受け取るリアルタイムデータをストリームとして配信します。EventBridge では、データは AWS Lambda などのターゲットにルーティングされます。これにより、サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、「Amazon EventBridge ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs を使用すると、Amazon RDS インスタンス、AWS CloudTrail、およびその他のソースからのログファイルをモニタリングして保存し、それらにアクセスすることができます。Amazon CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることもできます。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- AWS CloudTrail は、AWS アカウントによって行われた、またはそのアカウントに代わって実行された API コールと関連イベントをキャプチャします。次に、CloudTrailは指定した Amazon S3 バケットにログファイルを渡します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。詳細については、AWS CloudTrailユーザーガイドを参照してください。

- データベースアクティビティストリームは、Amazon RDS の機能であり、DB インスタンス内のアクティビティのストリームをほぼリアルタイムで提供します。Amazon RDS は、アクティビティを Amazon Kinesis データストリーミングにプッシュします。Kinesis ストリーミングが自動的に作成されます。Kinesis から、Amazon Data Firehose や AWS Lambda などの AWS サービスを設定して、ストリーミングを消費し、データを保存できます。

トピック

- Amazon RDS コンソールでのログ、イベント、およびストリーミングの表示
- Amazon RDS イベントのモニタリング
- Amazon RDS ログファイルのモニタリング
- AWS CloudTrail での Amazon RDS API コールのモニタリング
- データベースアクティビティストリームを使用した Amazon RDS のモニタリング
- Amazon GuardDuty RDS Protection による脅威のモニタリング

Amazon RDS コンソールでのログ、イベント、およびストリーミングの表示

Amazon RDS が AWS のサービス と統合し、RDS コンソールでログ、イベント、およびデータベースアクティビティストリーミングに関する情報を表示できるようになりました。

RDS DB インスタンスの [ログとイベント] タブで、次の情報が表示されます。

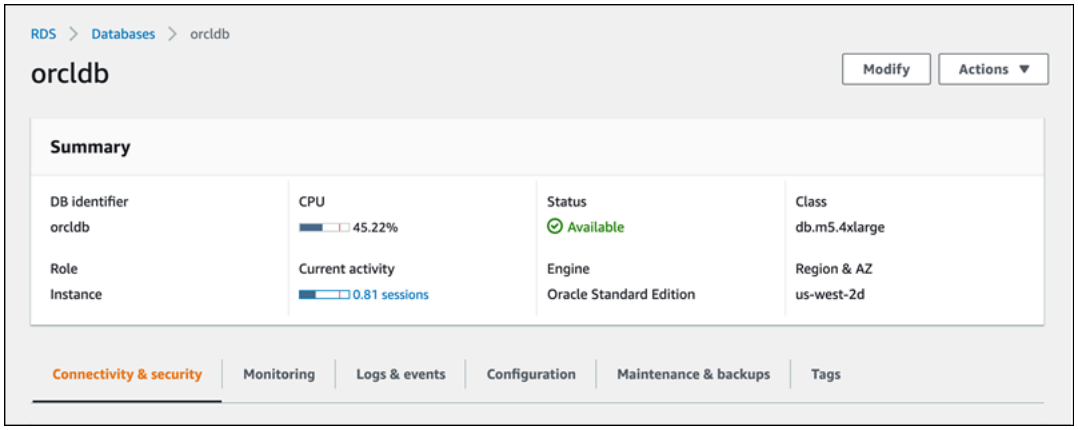
- [Amazon CloudWatch alarms] (Amazon CloudWatch アラーム) – DB インスタンス用に設定したメトリクスアラームがすべて表示されます。アラームを設定していない場合は、RDS コンソールでアラームを作成できます。詳細については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。
- [Recent events] (最近のイベント) – RDS DB インスタンスのイベント (環境の変更) の概要が表示されます。詳細については、「Amazon RDS イベントの表示」を参照してください。
- [Logs] (ログ) – DB インスタンスによって生成されたデータベースログファイルが表示されます。詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

[Configuration] (設定) タブには、データベースアクティビティストリーミングに関する情報が表示されます。

RDS コンソールで、 DB インスタンスのログ、イベント、およびストリームを表示するには

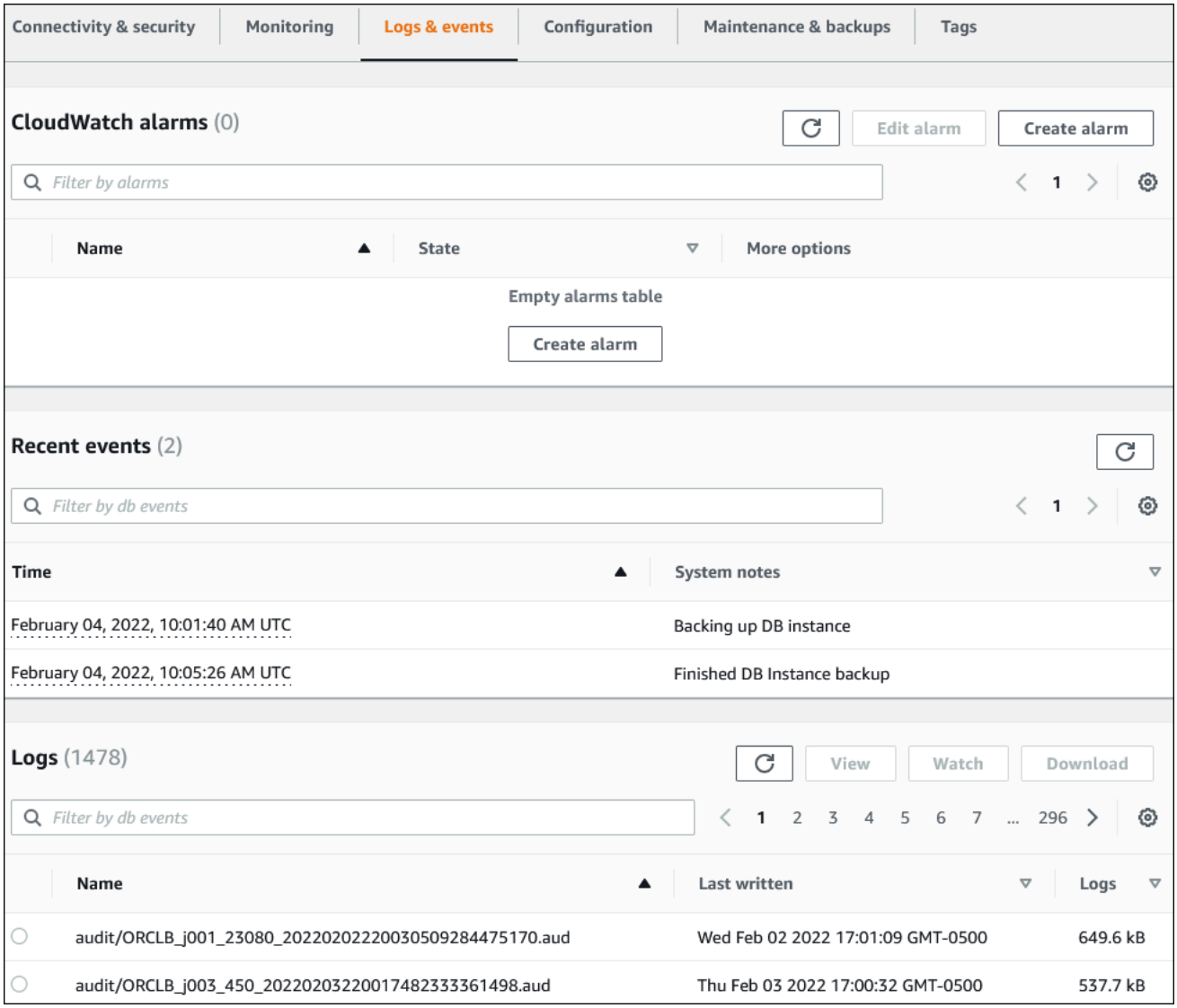
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. モニタリングする DB インスタンスの名前を選択します。

   [Database] (データベース) ページが表示されます。次の例は、`orclb` という名前の Oracle データベースを示しています。



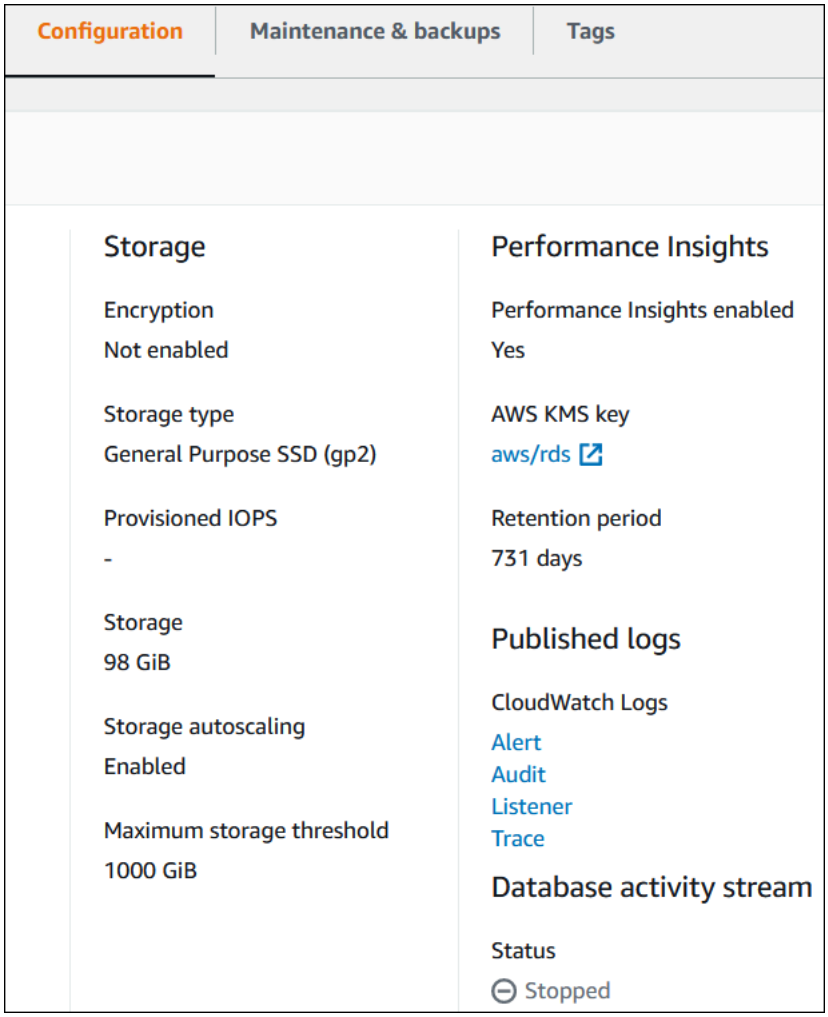
4. [ログとイベント] を選択します。

   [Logs & events] (ログとイベント) セクションが表示されます。



5. [設定] を選択します。

   次の例は、DB インスタンスのデータベースアクティビティストリームのステータスを示しています。


Configuration
Maintenance & backups
Tags
Storage
Encryption
Not enabled
Storage type
General Purpose SSD (gp2)
Provisioned IOPS
-
Storage
98 GiB
Storage autoscaling
Enabled
Maximum storage threshold
1000 GiB
Performance Insights
Performance Insights enabled
Yes
AWS KMS key
aws/rds
Retention period
731 days
Published logs
CloudWatch Logs
Alert
Audit
Listener
Trace
Database activity stream
Status
Stopped

Amazon RDS イベントのモニタリング

イベントは環境の変更を示します。これは、AWS 環境、SaaS パートナーサービスやアプリケーション、カスタムアプリケーションまたはサービスのいずれかです。RDS イベントの説明については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

トピック

- Amazon RDS のイベントの概要
- Amazon RDS イベントの表示
- Amazon RDS イベント通知の操作
- Amazon RDS イベントでトリガーするルールの作成
- Amazon RDS イベントカテゴリとイベントメッセージ

Amazon RDS のイベントの概要

RDS イベントは、Amazon RDS 環境における変更を示します。例えば、、DB インスタンスの状態が保留から実行に変わると Amazon RDS はイベントを生成します。Amazon RDS は、EventBridge に対してほぼリアルタイムでイベントを配信します。

> **Note**
>
> Amazon RDS は、ベストエフォートベースでイベントを発行します。通知イベントの順序または存在に依存するプログラムは、順序が正しくないか、または欠落している可能性があるため、作成しないことをお勧めします。

Amazon RDS は、次のリソースに関連するイベントを記録します。

- DB インスタンス

  DB インスタンスイベントのリストについては、「DB インスタンスイベント」を参照してください。
- DB パラメータグループ

  DB パラメータグループイベントのリストについては、「DB パラメータグループイベント」を参照してください。
- DB セキュリティグループ

DB セキュリティグループイベントのリストについては、「DB セキュリティグループイベント」を参照してください。

- DB スナップショット

  DB スナップショットイベントのリストについては、「DB スナップショットイベント」を参照してください。

- RDS Proxy イベント

  RDS Proxy イベントのリストについては、「RDS Proxy イベント」を参照してください。

- ブルー/グリーンデプロイイベント

  ブルー/グリーンデプロイイベントのリストについては、「ブルー/グリーンデプロイイベント」を参照してください。

この情報には以下が含まれます。

- リクエストの日付と時刻
- イベントのソース名とソースタイプ
- イベントに関連付けられたメッセージ。
- イベント通知には、メッセージが送信されたときのタグが含まれ、イベントが発生した時点のタグが反映されない場合があります

Amazon RDS イベントの表示

Amazon RDS リソースの次のイベント情報を取得できます:

- リソース名
- リソースタイプ
- イベントの時刻
- イベントのメッセージの概要

イベントには、AWS マネジメントコンソール の以下の部分でアクセスできます。

- 過去 24 時間のイベントを示す [イベント] タブ。
- [データベース] タブの [ログとイベント] セクションの [最近のイベント] テーブル。過去 2 週間のイベントを表示できます。

また、describe-events AWS CLI コマンドまたは DescribeEvents RDS API オペレーションを使用してイベントを取得することもできます。AWS CLI または RDS API を使用してイベントを表示する場合は、最大で過去 14 日間のイベントを取得できます。

> **Note**
>
> イベントを長期間保存する必要がある場合は、Amazon RDS イベントを EventBridge に送信できます。詳細については、Amazon RDS イベントでトリガーするルールの作成を参照してください。

Amazon RDS イベントの説明については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

AWS CloudTrail を使用してリクエストパラメータを含むイベントの詳細情報にアクセスするには、「CloudTrail のイベント」を参照してください。

コンソール

過去 24 時間のすべての Amazon RDS イベントを表示するには

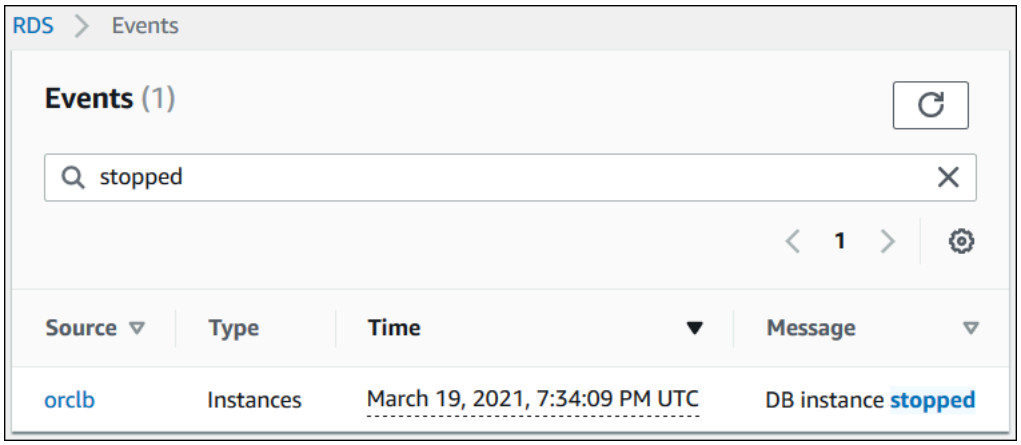
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインの [Events] を選択します。

   使用できるイベントがリストに表示されます。

3. (オプション) 検索語を入力して、結果をフィルタリングします。

   次の例は、文字 **stopped** によってフィルタリングされたイベントのリストを示しています。



AWS CLI

過去 1 時間に生成されたすべてのイベントを表示するには、describe-events をパラメータを指定せずに呼び出します。

```
aws rds describe-events
```

次の出力例は、DB インスタンスが停止したことを示しています。

```
{
    "Events": [
        {
            "EventCategories": [
                "notification"
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:testinst",
            "Date": "2022-04-22T21:31:00.681Z",
            "Message": "DB instance stopped",
            "SourceIdentifier": "testinst"
        }
    ]
}
```

過去 10080 分間 (7 日間) のすべての Amazon RDS イベントを表示するには、describe-events AWS CLI コマンドを呼び出し、`--duration` パラメータを `10080` に設定します。

```
aws rds describe-events --duration 10080
```

次の例は、DB インスタンス *testtest-instance* について、指定された時間範囲内のイベントを示しています。

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

次の出力例は、バックアップのステータスを示しています。

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

過去 14 日間のすべての Amazon RDS インスタンスのイベントを表示するには、RDS API の DescribeEvents オペレーションを呼び出して、`Duration` パラメータを `20160` に設定します。

Amazon RDS イベント通知の操作

Amazon RDS では、Amazon RDS のイベントが発生したときに、Amazon Simple Notification Service (Amazon SNS) を使用して通知を送信します。これらの通知については、AWS リージョンの Amazon SNS でサポートされているすべての通知の形式が使用可能です (E メール、テキストメッセージ、HTTP エンドポイントの呼び出しなど)。

トピック

- Amazon RDS イベント通知の概要
- Amazon SNS トピックに通知を発行するアクセス許可を付与する
- Amazon RDS イベント通知にサブスクライブする
- Amazon RDS イベント通知タグと属性
- Amazon RDS DB イベント通知サブスクリプションのリスト化
- Amazon RDS イベント通知サブスクリプションを変更する
- Amazon RDS イベント通知サブスクリプションへのソース識別子の追加
- Amazon RDS イベント通知サブスクリプションからのソース識別子の削除
- Amazon RDS イベント通知カテゴリのリスト化
- Amazon RDS イベント通知サブスクリプションの削除

Amazon RDS イベント通知の概要

Amazon RDS は、サブスクライブ可能なカテゴリにイベントをグループ分けします。これにより、そのカテゴリのイベントが発生すると、通知を受け取ることができます。

トピック

- イベントサブスクリプションの対象となる RDS リソース
- Amazon RDS イベント通知にサブスクライブする基本的な手順は次のとおりです。
- RDS イベント通知のデリバリー
- Amazon RDS イベント通知の請求
- Amazon EventBridgei を使用した Amazon RDS イベントの例

イベントサブスクリプションの対象となる RDS リソース

次のリソースのイベントカテゴリにサブスクライブできます。

- DB インスタンス
- DB スナップショット
- DB パラメータグループ
- DB セキュリティグループ
- RDS Proxy
- カスタムエンジンバージョン

例えば、特定の DB インスタンスのバックアップカテゴリにサブスクライブした場合、DB インスタンスに影響するバックアップ関連のイベントが発生するたびに通知が送信されます。DB インスタンスの設定変更カテゴリにサブスクライブした場合は、DB インスタンスが変更されると、通知を受け取ります。また、イベント通知サブスクリプションが変更されても、通知を受け取ります。

複数の異なるサブスクリプションを作成することもできます。例えば、すべての DB インスタンスの全イベント通知を受信するサブスクリプションや、DB インスタンスのサブセットに関する重要なイベントのみを含むサブスクリプションを作成することもできます。2 番目のサブスクリプションでは、フィルターで 1 つ以上の DB インスタンスを指定します。

Amazon RDS イベント通知にサブスクライブする基本的な手順は次のとおりです。

Amazon RDS イベント通知にサブスクライブする手順は次のとおりです。

1. Amazon RDS コンソール、AWS CLI、または API を使用して、Amazon RDS イベント通知サブスクリプションを作成します。

   Amazon RDS では、Amazon SNS トピックの ARN を使用して各サブスクリプションを識別します。Amazon RDS コンソールでは、サブスクリプションの作成時に ARN が作成されます。Amazon SNS コンソール、AWS CLI、または Amazon SNS API を使用して ARN を作成します。
2. サブスクリプションの作成で指定したアドレス宛てに、Amazon RDS から承認の E メールまたは SMS メッセージが送信されます。
3. サブスクリプションを確認するには、受信した通知に記載されているリンクを選択します。
4. Amazon RDS コンソールでは、サブスクリプションのステータスで [My Event Subscriptions] (イベントのサブスクリプション) が更新されます。
5. Amazon RDS は、サブスクリプションを作成したときに指定したアドレスへの通知の送信を開始します。

Amazon SNS を使用する際の Identity and Access Management に関する詳細については、Amazon Simple Notification Service デベロッパーガイドの「Identity and access management in Amazon SNS」を参照してください。

DB インスタンスからのイベント通知の処理には、AWS Lambda を使用することができます。詳細については、AWS Lambda デベロッパーガイドの「Amazon RDS で AWS Lambda を使用する」を参照してください。

RDS イベント通知のデリバリー

Amazon RDS は、サブスクリプションを作成するときに指定したアドレスに通知を送信します。通知には、メッセージに関する構造化メタデータを提供する、メッセージ属性を含めることができます。メッセージ属性の詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

イベント通知が配信されるまでに最大 5 分かかります。

> **Important**
>
> Amazon RDS はイベントストリーミングのイベントの順番を保証しません。イベントの順番は変わる場合があります。

Amazon SNS が受信登録している HTTP または HTTPS エンドポイントに通知を送信した場合、エンドポイントに送信される POST メッセージには、JSON ドキュメントを含むメッセージ本文が含まれます。詳細については、Amazon Simple Notification Service デベロッパーガイドの「Amazon SNS メッセージと JSON 形式」を参照してください。

テキストメッセージで通知を送信するように SNS を設定できます。詳細については、Amazon Simple Notification Service デベロッパーガイドの「Mobile text messaging (SMS)」を参照してください。

サブスクリプションを削除せずに通知を無効にするには、Amazon RDS コンソールで [Enabled] (有効) として [No] (いいえ) を選択します。または、AWS CLI や Amazon RDS API を使用して `Enabled` パラメータを`false` に設定します。

Amazon RDS イベント通知の請求

Amazon RDS イベント通知の請求は、Amazon SNS を通じて行われます。使用したイベント通知に対して、Amazon SNS 料金が適用されます。Amazon SNS の請求の詳細については、「Amazon Simple Notification Service 料金表」を参照してください。

Amazon EventBridgei を使用した Amazon RDS イベントの例

次の例は、JSON 形式のさまざまなタイプの Amazon RDS イベントを示しています。JSON 形式でイベントをキャプチャして表示する方法を示すチュートリアルについては、「チュートリアル: Amazon EventBridge を使用して DB インスタンスの状態変化を ログに記録する」を参照してください。

トピック

- DB インスタンスイベントの例
- DB パラメータグループイベントの例
- DB スナップショットイベントの例

DB インスタンスイベントの例

以下は、JSON 形式の DB インスタンスイベントの例です。このイベントは、RDS が `my-db-instance` という名前のインスタンスのためにマルチ AZ フェイルオーバーを実行したことを示しています。イベント ID は RDS-EVENT-0049 です。

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Instance Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
  ],
  "detail": {
    "EventCategories": [
      "failover"
    ],
    "SourceType": "DB_INSTANCE",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "A Multi-AZ failover has completed.",
    "SourceIdentifier": "my-db-instance",
    "EventID": "RDS-EVENT-0049"
  }
```

```
}
```

DB パラメータグループイベントの例

以下は、JSON 形式の DB パラメータグループイベントの例です。イベントは、パラメータグループ time_zone でパラメータ my-db-param-group が更新されたことを示します。イベント ID は RDS-EVENT-0037 です。

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

DB スナップショットイベントの例

以下は、JSON 形式の DB スナップショットイベントの例です。イベントは、my-db-snapshot という名前のスナップショットの削除を示します。イベント ID は RDS-EVENT-0041 です。

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Snapshot Event",
  "source": "aws.rds",
```

```
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:snapshot:rds:my-db-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "deletion"
    ],
    "SourceType": "SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:snapshot:rds:my-db-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Deleted manual snapshot",
    "SourceIdentifier": "my-db-snapshot",
    "EventID": "RDS-EVENT-0041"
  }
}
```

Amazon SNS トピックに通知を発行するアクセス許可を付与する

Amazon Simple Notification Service (Amazon SNS) トピックに通知を発行するための Amazon RDS アクセス許可を付与するには、AWS Identity and Access Management (IAM) ポリシーを送信先トピックに添付します。アクセス許可の詳細については、Amazon Simple Notification Service デベロッパーガイドの「Amazon Simple Notification Service のケース例」を参照してください。

デフォルトで、Amazon SNS トピックには、同じアカウント内のすべての Amazon RDS リソースが通知を発行するのを許可するポリシーがあります。カスタムポリシーをアタッチして、クロスアカウント通知を許可したり、特定のリソースへのアクセスを制限したりできます。

発行先の Amazon SNS トピックにアタッチする IAM ポリシーの例を次に示します。トピックは、指定したプレフィックスと一致する名前を持つ DB インスタンスに制限されます。このポリシーを使用するには、次の値を指定します。

- `Resource` – Amazon SNS トピックの Amazon リソースネーム (ARN)
- `SourceARN` – RDS リソース ARN
- `SourceAccount` – 自分の AWS アカウント ID。

リソースタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon RDS で定義されるリソース」を参照してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
```

```
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Amazon RDS イベント通知にサブスクライブする

サブスクリプションを作成する最も簡単な方法は、RDS コンソールを使用する方法です。CLI または API を使用してイベント通知サブスクリプションを作成する場合、Amazon Simple Notification Service トピックを作成し、Amazon SNS コンソールまたは Amazon SNS API を使用してそのトピックにサブスクライブする必要があります。トピックの Amazon Resource Name (ARN) は、CLI コマンドや API オペレーションを送信するときに使用されるため維持する必要があります。SNS トピックの作成とサブスクライブについては、Amazon Simple Notification Service デベロッパーガイドの「Amazon SNS のスタート方法」を参照してください。

通知を受け取る対象となるソースのタイプ、およびイベントをトリガーする Amazon RDS ソースを指定できます。

ソースタイプ

ソースのタイプ。例えば、[Source type] (ソースタイプ) が [Instances] (インスタンス) の可能性があります。ソースタイプを選択する必要があります。

インクルードする####

イベントを生成している Amazon RDS リソース。例えば、[Select specific instances] (特定のインスタンスを選択) を選択して、[myDBInstance1] を選択します。

次の表は、インクルードする####を指定した場合と、指定しなかった場合の結果について説明しています。

インクルードするリソース	説明	例
指定	RDS は、指定したリソースのみに関するすべてのイベントを通知します。	選択した [Source type] (ソースタイプ) が [Instances] (インスタンス) で、リソースが [myDBInstance1] の場合は、RDS は MyDBInstance1 のみに関するすべてのイベントを通知します。
指定されていません	RDS は、すべての Amazon RDS リソースについて、指定したソースタイプのイベントを通知します。	[Source type] (ソースタイプ) が [Instances] (インスタンス) の場合は、RDS はそのアカウントの

インクルードするリソース	説明	例
		すべてのインスタンス関連イベントを通知します。

デフォルトでは、Amazon SNS トピックのサブスクライバーは、トピックに対して発行されたすべてのメッセージを受信します。メッセージのサブセットのみを受信するには、サブスクライバーはトピックのサブスクリプションにフィルターポリシーを割り当てる必要があります。詳細については、「Amazon SNS 開発者ガイド」の「Amazon SNS メッセージのフィルター処理」を参照してください。

コンソール

RDS イベント通知にサブスクライブするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[イベントサブスクリプション] を選択します。
3. [イベントサブスクリプション] ページで、[イベントサブスクリプションの作成] を選択します。
4. サブスクリプションの詳細を次のように入力します。

   a. [Name] (名前) に、イベント通知サブスクリプションの名前を入力します。

   b. [Send notifications to] (通知の送信先) については、次のいずれかを実行します。

      - [New email topic] (新しい E メールトピック) を選択します。E メールトピックの名前と受信者のリストを入力します。イベントのサブスクリプションは、プライマリアカウントの連絡先と同じメールアドレスに設定することをお勧めします。推奨事項、サービスイベント、および個人の健康メッセージは、さまざまなチャネルを使用して送信されます。同じメールアドレスでサブスクリプションすると、すべてのメッセージが 1 か所にまとめられます。
      - [Amazon Resource Name (ARN)] (Amazon リソースネーム (ARN)) を選択します。次に、Amazon SNS トピックで既存の Amazon SNS ARN を選択します。

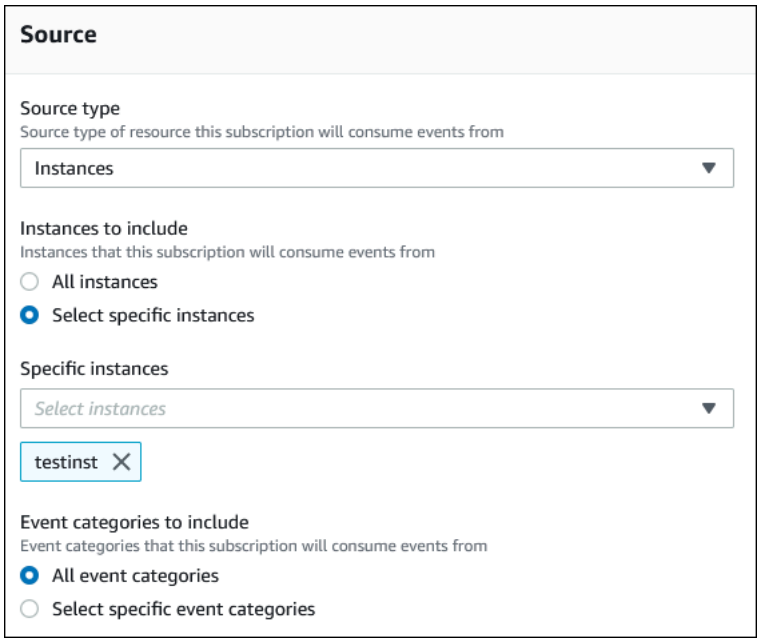
        サーバー側の暗号化 (SSE) で有効になっているトピックを使用する場合は、Amazon RDS に、AWS KMS key にアクセスするのに必要な許可を付与します。詳細について

は、Amazon Simple Notification Service デベロッパーガイドの「AWS サービスと暗号化されたトピックのイベントソース間の互換性を保つ」を参照してください。

c. [ソースタイプ] で、ソースタイプを選択します。例えば、[Instances] (インスタンス) または [Parameter groups] (パラメータグループ) を選択します。

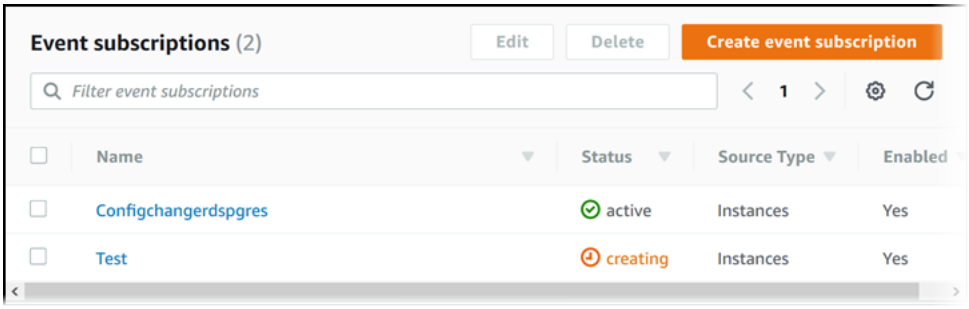
d. イベントの通知を受け取るイベントのカテゴリとリソースを選択します。

次の例では、`testinst` という名前の DB インスタンスについて、イベント通知を設定します。



e. [作成] を選択します。

Amazon RDS コンソールでは、サブスクリプションが作成されることが示されます。



AWS CLI

RDS イベント通知を受信するには、AWS CLI `create-event-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`
- `--sns-topic-arn`

Example

Linux、macOS、Unix の場合:

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Windows の場合:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Amazon RDS イベント通知を受信するには、Amazon RDS API 関数 `CreateEventSubscription` を呼び出します。以下の必須パラメータを含めます。

- `SubscriptionName`
- `SnsTopicArn`

Amazon RDS イベント通知タグと属性

Amazon RDS が Amazon Simple Notification Service (SNS) または Amazon EventBridge にイベント通知を送信するとき、通知にはメッセージ属性とイベントタグが含まれます。RDS はメッセージと共にメッセージ属性を個別に送信しますが、イベントタグはメッセージの本文に含まれています。メッセージ属性と Amazon RDS タグを使用して、リソースにメタデータを追加します。これらのタグは、DB インスタンスに関する独自の表記で変更できます。Amazon RDS リソースのタグ付けの詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

デフォルトでは、Amazon SNS と Amazon EventBridge は、送信されたすべてのメッセージを受信します。SNS と EventBridge では、メッセージをフィルタリングして、メールやテキストメッセージ、または HTTP エンドポイントの呼び出しなど、希望する通信モードに通知を送信できます。

> **Note**
>
> メールまたはテキストメッセージで送信される通知には、イベントタグはありません。

次の表では、トピックサブスクライバーに送信された RDS イベントのメッセージ属性を示します。

Amazon RDS イベント属性	説明
EventID	RDS イベントメッセージの識別子。例えば、RDS-EVENT-0006。
リソース	イベントを発行するリソースの ARN 識別子 (例: `arn:aws:rds:ap-southeast-2:123456789012:db:database-1`)。

RDS タグは、サービスイベントの影響を受けたリソースに関するデータを提供します。通知が SNS または EventBridge に送信されると、RDS はメッセージ本文にタグの現在の状態を追加します。

SNS メッセージ属性のフィルター処理の詳細については、Amazon Simple Notification Service デベロッパーガイドの「Amazon SNS メッセージフィルター処理」を参照してください。

EventBridge のイベントタグのフィルター処理については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge のイベントパターンで使用する比較演算子」を参照してください。

SNS のペイロードベースのタグのフィルタリングについては、「Introducing payload-based message filtering for Amazon SNS」を参照してください。

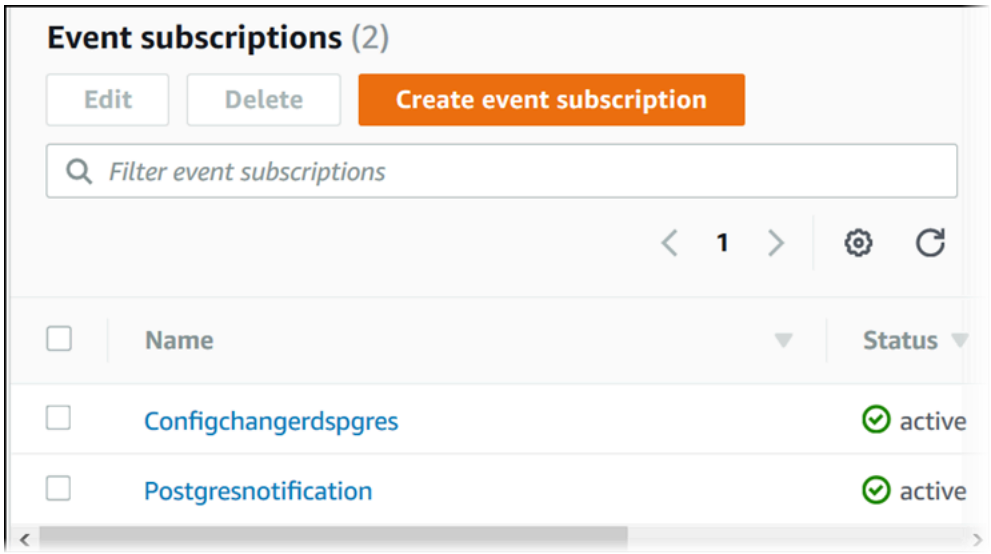
Amazon RDS DB イベント通知サブスクリプションのリスト化

現在の Amazon RDS イベント通知サブスクリプションのリストを表示できます。

コンソール

現在の Amazon RDS イベント通知サブスクリプションのリストを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[イベントサブスクリプション] を選択します。[イベントサブスクリプション] ペインにイベント通知サブスクリプションが一覧表示されます。



AWS CLI

現在の Amazon RDS イベント通知サブスクリプションを一覧表示するには、AWS CLI の `describe-event-subscriptions` コマンドを使用します。

Example

次の例は、すべてのイベントサブスクリプションを表しています。

```
aws rds describe-event-subscriptions
```

次の例は、`myfirsteventsubscription` を表しています。

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

Amazon RDS イベント通知に対する現在のサブスクリプションを一覧表示するには、Amazon RDS API の DescribeEventSubscriptions アクションを呼び出します。

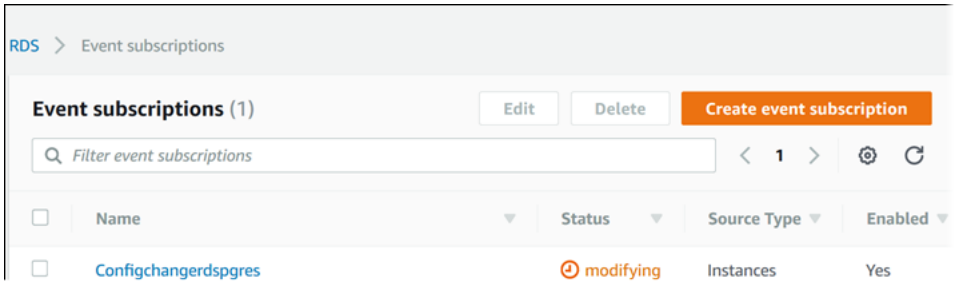
Amazon RDS イベント通知サブスクリプションを変更する

サブスクリプションを作成すると、サブスクリプション名、ソース識別子、カテゴリ、トピック ARN を変更できます。

コンソール

Amazon RDS イベント通知サブスクリプションを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[イベントサブスクリプション] を選択します。
3. [イベントサブスクリプション] ペインで、変更するサブスクリプションを選択し、[編集] をクリックします。
4. [ターゲット] セクションまたは [ソース] セクションのいずれかでサブスクリプションを変更します。
5. [編集] を選択します。Amazon RDS コンソールでは、サブスクリプションが変更されることが示されます。



AWS CLI

Amazon RDS イベント通知サブスクリプションを変更するには、AWS CLI の `modify-event-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`

Example

次のコードで `myeventsubscription` を有効にします。

Linux、macOS、Unix の場合:

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

Windows の場合:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Amazon RDS イベントを変更するには、Amazon RDS API オペレーション `ModifyEventSubscription` を呼び出します。以下の必須パラメータを含めます。

- `SubscriptionName`

Amazon RDS イベント通知サブスクリプションへのソース識別子の追加

既存のサブスクリプションにソース識別子 (イベントを生成する Amazon RDS ソース) を追加できます。

コンソール

Amazon RDS コンソールを使用し、サブスクリプションを変更するときに選択または選択解除することで、ソース識別子を簡単に追加または削除できます。詳細については、「Amazon RDS イベント通知サブスクリプションを変更する」を参照してください。

AWS CLI

Amazon RDS イベント通知サブスクリプションにソース識別子を追加するには、AWS CLI の `add-source-identifier-to-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`
- `--source-identifier`

Example

次の例は、ソース識別子 `mysqldb` を `myrdseventsubscription` サブスクリプションに追加します。

Linux、macOS、Unix の場合:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Windows の場合:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Amazon RDS イベント通知サブスクリプションにソース識別子を追加するには、Amazon RDS API `AddSourceIdentifierToSubscription` を呼び出します。以下の必須パラメータを含めます。

- SubscriptionName
- SourceIdentifier

Amazon RDS イベント通知サブスクリプションからのソース識別子の削除

そのソースのイベントの通知を今後は受け取らない場合、サブスクリプションからソース識別子 (イベントを生成する Amazon RDS ソース) を削除できます。

コンソール

Amazon RDS コンソールを使用し、サブスクリプションを変更するときに選択または選択解除することで、ソース識別子を簡単に追加または削除できます。詳細については、「Amazon RDS イベント通知サブスクリプションを変更する」を参照してください。

AWS CLI

Amazon RDS イベント通知サブスクリプションからソース識別子を削除するには、AWS CLI の `remove-source-identifier-from-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`
- `--source-identifier`

Example

次の例は、ソース識別子 `mysqldb` を `myrdseventsubscription` サブスクリプションから削除します。

Linux、macOS、Unix の場合:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Windows の場合:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Amazon RDS イベント通知サブスクリプションからソース識別子を削除するには、Amazon RDS API `RemoveSourceIdentifierFromSubscription` コマンドを使用します。以下の必須パラメータを含めます。

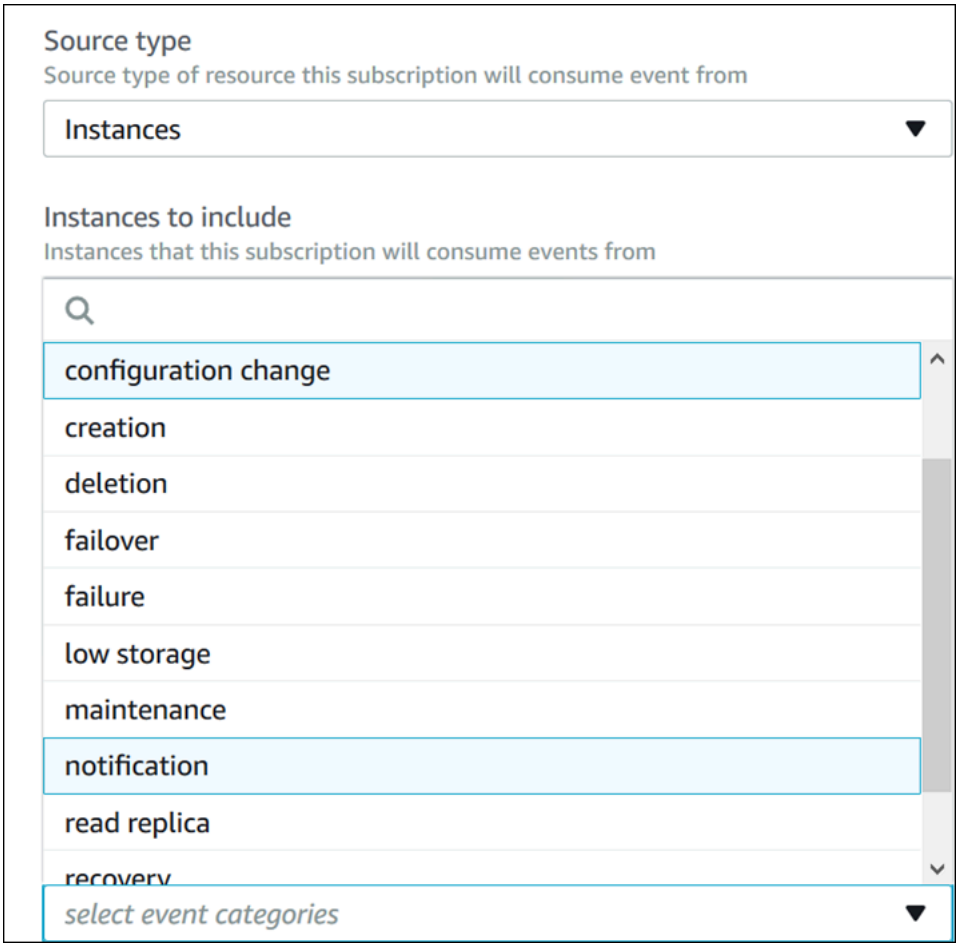
- `SubscriptionName`
- `SourceIdentifier`

Amazon RDS イベント通知カテゴリのリスト化

リソースタイプのすべてのイベントはカテゴリに分類されます。使用できるカテゴリのリストを表示するには、次の手順を実行します。

コンソール

イベント通知サブスクリプションを作成または変更するとき、Amazon RDS コンソールにイベントカテゴリが表示されます。詳細については、「Amazon RDS イベント通知サブスクリプションを変更する」を参照してください。



AWS CLI

Amazon RDS イベント通知カテゴリを一覧表示するには、AWS CLI の `describe-event-categories` コマンドを使用します。このコマンドには必須パラメータはありません。

Example

```
aws rds describe-event-categories
```

API

Amazon RDS イベント通知カテゴリを一覧表示するには、Amazon RDS API `DescribeEventCategories` コマンドを使用します。このコマンドには必須パラメータはありません。

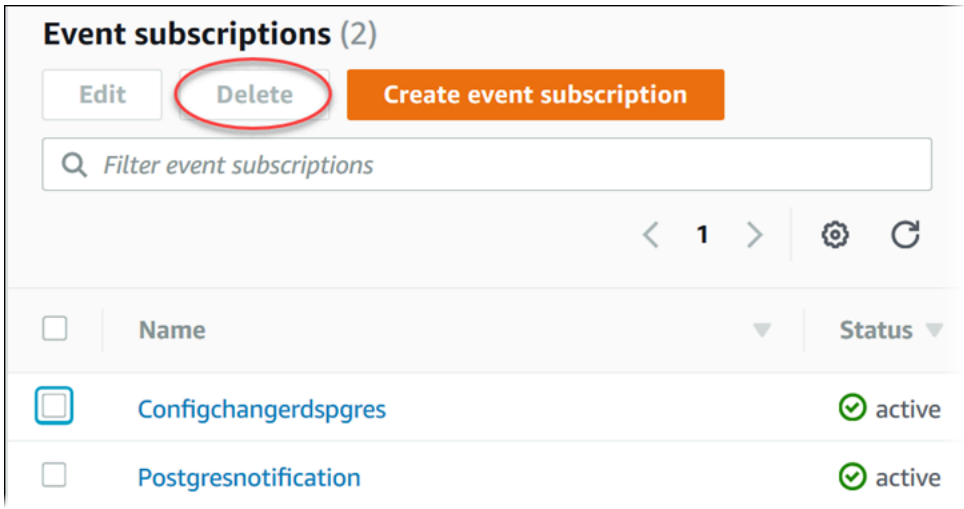
Amazon RDS イベント通知サブスクリプションの削除

不要になったサブスクリプションは削除できます。トピックへのすべてのサブスクライバは、サブスクリプションにより指定されたイベント通知を受け取らなくなります。

コンソール

Amazon RDS イベント通知サブスクリプションを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[DB イベントサブスクリプション] を選択します。
3. [My DB Event Subscriptions] (自分の DB イベントサブスクリプション) ペインで、削除するサブスクリプションを選択します。
4. [削除] を選択します。
5. サブスクリプションを削除中であることが Amazon RDS コンソールに表示されます。



AWS CLI

Amazon RDS イベント通知サブスクリプションを削除するには、AWS CLI の `delete-event-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`

Example

以下の例では、サブスクリプション myrdssubscription を削除します。

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Amazon RDS イベント通知サブスクリプションを削除するには、RDS API DeleteEventSubscription コマンドを使用します。以下の必須パラメータを含めます。

- SubscriptionName

Amazon RDS イベントでトリガーするルールの作成

Amazon EventBridge を使用すると、AWS のサービスを自動化して、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに対応できます。

トピック

- Amazon RDS イベントを Amazon EventBridge に送信するルールの作成
- チュートリアル: Amazon EventBridge を使用して DB インスタンスの状態変化を口グに記録する

Amazon RDS イベントを Amazon EventBridge に送信するルールの作成

簡単なルールを記述して、対象となる Amazon RDS イベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。イベントを JSON 形式で受け取る AWS Lambda 関数や Amazon SNS トピックなど、さまざまなターゲットの設定が可能です。例えば、DB インスタンスを作成または削除するたびに Amazon EventBridge にイベントを送信するように Amazon RDS を設定できます。詳細については、Amazon CloudWatch Events ユーザーガイド、および Amazon EventBridge ユーザーガイドを参照してください。

RDS イベントでトリガーするルールを作成するには:

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインの [イベント] で、[ルール] を選択します。
3. [ルールの作成] を選択します。
4. [イベントソース] で、以下の操作を実行します。
    a. [Event Pattern] を選択します。
    b. [サービス名] に [リレーショナルデータベースサービス (RDS)] を選択します。
    c. [Event Type] (イベントタイプ) として、イベントをトリガーする Amazon RDS リソースのタイプを選択します。例えば、内の DB インスタンスがイベントをトリガーする場合は、[RDS DB Instance Event] (RDS DB インスタンスイベント) を選択します。
5. [ターゲット] で [ターゲットの追加] を選択し、選択した種類のイベントが検出されたときに対応する AWS のサービスを選択します。
6. このセクションの他のフィールドに、このターゲットタイプに固有の情報を入力します (必要な場合)。

7. 多くのターゲットタイプで、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。この場合、EventBridge は、以下をイベントで実行するために必要な IAM ロールを作成できます。

   - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成するを選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。

8. オプションで、このルールに別のターゲットを追加するには、ステップ 5~7 を繰り返します。
9. [詳細の設定] を選択します。[Rule definition] で、ルールの名前と説明を入力します。

   ルール名はこのリージョン内で一意である必要があります。

10. [Create rule] を選択します。

詳細については、「Amazon CloudWatch ユーザーガイド」の「イベントでトリガーするEventBridge ルールの作成」を参照してください。

チュートリアル: Amazon EventBridge を使用して DB インスタンスの状態変化をログに記録する

このチュートリアルでは、Amazon RDS インスタンスの状態変化をログに記録する AWS Lambda 関数を作成します。次に、既存の RDS DB インスタンスの状態変化があったときに関数を実行するルールを作成します。このチュートリアルでは、一時的にシャットダウンできる小規模なテストインスタンスが実行されていることを前提としています。

> **Important**
>
> 実行中の本番 DB インスタンスに対して、このチュートリアルを実行しないでください。

トピック

- ステップ 1: AWS Lambda 関数を作成する
- ステップ 2: ルールを作成する
- ステップ 3: ルールをテストする

ステップ 1: AWS Lambda 関数を作成する

状態変更イベントのログを記録する Lambda 関数を作成します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[今すぐ始める] を選択します。それ以外の場合は、[関数の作成] を選択します。
3. [Author from scratch] を選択します。
4. [関数の作成] ページで、次の操作を実行します。
   a. Lambda 関数の名前と説明を入力します。例えば、関数名を **RDSInstanceStateChange** とします。
   b. [Runtime] (ランタイム) では、[Node.js 16x] を選択します。
   c. [Architecture] (アーキテクチャ) では、[x86_64] を選択します。
   d. [Execution role] (実行ロール) では、次のいずれかを実行します。
      - [基本的な Lambda アクセス権限で新しいロールを作成] を選択します。
      - [Existing role] (既存のロール) では、[Use an existing role] (既存のロールを使用する) を選択します。使用するロールを選択します。
   e. [Create function] (関数の作成) を選択します。
5. [RDSInstanceStateChange] ページで、次の操作を行います。
   a. [コードソース] で、[index.js] を選択します。
   b. で、[index.js] ウィンドウで、既存のコードを削除します。
   c. 次のコードを入力します。

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. [デプロイ] を選択します。

ステップ 2: ルールを作成する

Amazon RDS インスタンスを起動するたびに Lambda 関数を実行するルールを作成します。

EventBridge ルールを作成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [ルール] を選択します。
3. [ルールの作成] を選択します。
4. ルールの名前と説明を入力します。例えば、「**RDSInstanceStateChangeRule**」と入力します。
5. [Rule with an event pattern] (イベントパターンを持つルール) を選択してから、[Next] (次へ) を選択します。
6. [イベントソース] で、[AWS イベントまたは EventBridge パートナーイベント] を選択してください。
7. 下にスクロールして、[Event pattern] (イベントパターン) セクションを展開します。
8. イベントソース で AWS のサービス を選択してください。
9. [AWS サービス] では [リレーショナルデータベースサービス (RDS)] を選択します。
10. [イベントタイプ] で、[RDS DB インスタンスイベント] を選択します。
11. デフォルトのイベントパターンのままにします。次に、[次へ] を選択します。
12. [ターゲットタイプ] では、[AWS サービス] を選択します。
13. [Select a target] (ターゲットを選択) では、[Lambda function] (Lambda 関数) を選択します。
14. [Function] (関数) では、作成した Lambda 関数を選択します。次に、[次へ] を選択します。
15. [Configure Tags] (タグの設定) で、[Next] (次へ) を選択します。
16. ルール内の手順を確認します。次に、[Create rule] (ルールの作成) を選択します。

ステップ 3: ルールをテストする

ルールをテストするには、RDS DB インスタンスをシャットダウンします。インスタンスのシャットダウンの数分後に、Lambda 関数が呼び出されたことが確認できます。

DB インスタンスを停止してルールをテストするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. RDS DB インスタンスを停止します。

3. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
4. ナビゲーションペインで、[ルール] を選択し、作成したルールの名前を選択します。
5. [ルールの詳細] で、[モニタリング] を選択します。

   Amazon CloudWatch コンソールにリダイレクトされます。リダイレクトされない場合は、[CloudWatch でメトリクスを表示] をクリックします。
6. [すべてのメトリクス] で、作成したルールの名前を選択します。

   グラフには、ルールが呼び出されたことが示されます。
7. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
8. Lambda 関数 (/aws/lambda/***function-name***) のロググループの名前を選択します。
9. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーミング名を選択します。次のような受信イベントが表示されます。

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

JSON 形式の RDS イベントの例については、「Amazon RDS のイベントの概要」を参照してください。

10. (オプション) 終了したら、Amazon RDS コンソールを開き、停止したインスタンスをスタートできます。

Amazon RDS イベントカテゴリとイベントメッセージ

Amazon RDS では、多数のイベントがカテゴリ分けされて生成され、これらには Amazon RDS コンソール、AWS CLI、または API を使用してのサブスクライブが可能です。

トピック

- DB クラスターイベント
- DB クラススナップショットイベント
- DB インスタンスイベント
- DB パラメータグループイベント
- DB セキュリティグループイベント
- DB スナップショットイベント
- RDS Proxy イベント
- ブルー/グリーンデプロイイベント
- カスタムエンジンバージョンイベント

DB クラスターイベント

次の表は、DB クラスターがソースタイプである場合のイベントカテゴリとイベントのリストを示します。

マルチ AZ DB クラスターデプロイの詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。

カテゴリ	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0016	マスター認証情報をリセットします。	なし
作成	RDS-EVENT-0170	DB クラスターが作成されました。	なし
フェイルオーバー	RDS-EVENT-0069	クラスターフェイルオーバーが失敗しました。クラスターインスタンスの状態	なし

カテゴリ	RDS イベント ID	メッセージ	注意事項
		を確認して、もう一度試してください。	
フェイルオーバー	RDS-EVENT-0070	以前のプライマリを再度プロモートします: *name*。	なし
フェイルオーバー	RDS-EVENT-0071	インスタンスへのフェイルオーバーが完了しました: *name*。	なし
フェイルオーバー	RDS-EVENT-0072	DB インスタンスへの同じ AZ フェイルオーバーを開始しました: *name*。	なし
フェイルオーバー	RDS-EVENT-0073	DB インスタンスへのクロスAZ フェイルオーバーを開始しました: *name*。	なし
失敗	RDS-EVENT-0354	リソースに互換性がないため、DB クラスターを作成できません。*#####*。	*#####*には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0355	リソース制限が不十分なため、DB クラスターを作成できません。*#####*。	*#####*には、障害に関する詳細が含まれます。
メンテナンス	RDS-EVENT-0156	DB クラスターで DB エンジンのマイナーバージョンをアップグレードできます。	なし
メンテナンス	RDS-EVENT-0173	データベースクラスターエンジンのバージョンがアップグレードされました。	DB クラスターのパッチ適用が完了しました。

カテゴリ	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0174	データベースクラスターはアップグレードできない状態です。	なし
メンテナンス	RDS-EVENT-0176	データベースクラスターエンジンのメジャーバージョンがアップグレードされました。	なし
メンテナンス	RDS-EVENT-0177	データベースクラスターのアップグレードが進行中です。	なし
メンテナンス	RDS-EVENT-0286	データベースクラスターエンジン *version_number* バージョンのアップグレードが開始しました。クラスターはオンラインのままです。	なし
メンテナンス	RDS-EVENT-0287	オペレーティングシステムのアップグレード要件が検出されました。	なし
メンテナンス	RDS-EVENT-0288	クラスターオペレーティングシステムのアップグレードが開始されます。	なし
メンテナンス	RDS-EVENT-0289	クラスターオペレーティングシステムのアップグレードが完了しました。	なし

カテゴリ	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0290	データベースクラスターにパッチが適用されました: ソースバージョン *version_number* => *new_version_number* 。	なし
メンテナンス	RDS-EVENT-0410	データベースクラスターエンジンのバージョンアップグレードの事前チェックが開始されました。	なし
メンテナンス	RDS-EVENT-0412	データベースクラスターエンジンのバージョンアップグレードの事前チェックが失敗またはタイムアウトしました。	なし
メンテナンス	RDS-EVENT-0413	DB クラスターのアップグレード前のタスクが進行中です。	なし
メンテナンス	RDS-EVENT-0414	DB クラスターのアップグレード後のタスクが進行中です。	なし
メンテナンス	RDS-EVENT-0417	データベースクラスターエンジンのバージョンのアップグレードが開始しました。	なし
通知	RDS-EVENT-0172	クラスターの名前が *name* から *name* に変更されました。	なし

カテゴリ	RDS イベント ID	メッセージ	注意事項
リードレプリカ	RDS-EVENT-0411	データベースクラスターエンジンのバージョンアップグレードの事前チェックが終了しました。	なし

DB クラススナップショットイベント

次の表は、DB クラスターのスナップショットがソースタイプである場合のイベントカテゴリとイベントのリストを示しています。

Category	RDS イベント ID	メッセージ	注意事項
バックアップ	RDS-EVENT-0074	手動クラススナップショットの作成	なし
バックアップ	RDS-EVENT-0075	手動クラススナップショットが作成されました。	なし
通知	RDS-EVENT-0162	クラスターのスナップショットのエクスポートタスクに失敗しました。	なし
通知	RDS-EVENT-0163	クラスターのスナップショットのエクスポートタスクがキャンセルされました。	なし
通知	RDS-EVENT-0164	クラスターのスナップショットのエクスポートタスクが完了しました。	なし
バックアップ	RDS-EVENT-0168	自動クラススナップショットを作成しています。	なし

Category	RDS イベント ID	メッセージ	注意事項
バックアップ	RDS-EVENT-0169	自動クラスタースナップショットが作成されました。	なし
失敗	RDS-EVENT-0489	エクスポートタスク##: ######のため、テーブル##のテーブル抽出に失敗しました。	このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0491	エクスポートタスク##: テーブル##は低速のシングルスレッドモードでエクスポートされます。エクスポートの完了に遅延が発生することが予想されます。	このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0492	エクスポートタスク##: テーブル##は進行中で、エクスポートサイズは ## GB です。	このイベントは、完了までに 1 時間以上かかるテーブルに対してのみ送信されます。 このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0493	エクスポートタスク##は##ステージにあります。	なし
		エクスポートタスク *name*: エクスポートの進行状況は *number*% で、*number* テーブルが進行中、*number* テーブルが完了しています。完了したテーブルの合計サイズは ## GB です。	このイベントは、高速エクスポートではサポートされていません。

DB インスタンスイベント

次の表は、DB インスタンスがソースタイプである場合のイベントカテゴリとイベントを示しています。

Category	RDS イベント ID	メッセージ	注意事項
可用性	RDS-EVENT-0004	DB インスタンスがシャットダウンしました。	なし
高可用性	RDS-EVENT-0006	インスタンスが再起動しました。	なし
可用性	RDS-EVENT-0022	mysql の再起動中にエラーが発生しました: *message*。	MySQL の再起動中にエラーが発生しました。
可用性	RDS-EVENT-0221	DB インスタンスがストレージフルのしきい値に達し、データベースがシャットダウンされました。この問題に対処するために、割り当てられたストレージを増やすことができます。	なし
可用性	RDS-EVENT-0222	DB インスタンス *name* の空きストレージ容量が、割り当てられたストレージの*percentage* % と低くなっています [割り当てられたストレージ: *amount*、空きストレージ: *amount*]。空きストレージが *amount* より低い場合、データベースは破損を防ぐためにシャットダウンされます。この問題に対処するために、割り当てら	DB インスタンスが割り当てられたストレージの 90% 以上を消費する場合にのみ、RDS for MySQL に適用されます。Free Storage Space メトリクスを使用して、DB インスタンスのストレージスペースをモニタリングします。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
		れたストレージを増やすことができます。	
可用性	RDS-EVENT-0330	専用のトランザクションログボリュームの空きストレージ容量が DB インスタンス *name* に対して少なすぎます。ログボリュームの空きストレージは、割り当て済みストレージの *percentage* です。[割り当て済みストレージ: *amount*、空きストレージ: *amount*] 空きストレージが *amount* より少ない場合、データベースは破損を防ぐためにシャットダウンされます。この問題を解決するには、専用のトランザクションログボリュームを無効にすることができます。	詳細については、「専用ログボリューム (DLV)」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
可用性	RDS-EVENT-0331	専用のトランザクションログボリュームの空きストレージ容量が DB インスタンス *name* に対して少なすぎます。ログボリュームの空きストレージは、プロビジョニング済みストレージの *percentage* です。[プロビジョニング済みストレージ: *amount*、空きストレージ: *amount*] この問題を解決するには、専用のトランザクションログボリュームを無効にすることができます。	詳細については、「専用ログボリューム (DLV)」を参照してください。
可用性	RDS-EVENT-0396	Amazon RDS は、内部ユーザーパスワードのローテーション後に、このインスタンスの次のメンテナンスウィンドウでこのリードレプリカの再起動をスケジュールしました。	なし
可用性	RDS-EVENT-0419	Amazon RDS がデータベースインスタンス *name* の KMS 暗号化キーにアクセスできませんでした。このデータベースはアクセスできない状態になります。詳細については、Amazon RDS ドキュメントのトラブルシューティングセクションを参照してください。	なし

Category	RDS イベント ID	メッセージ	注意事項
バックアップ	RDS-EVENT-0001	DB インスタンスをバックアップする	なし
バックアップ	RDS-EVENT-0002	DB インスタンスのバックアップが完了しました。	なし
バックアップ	RDS-EVENT-0086	オプショングループ *name* をデータベースインスタンス *name* に関連付けることができませんでした。DB インスタンスクラスおよび設定でオプショングループ *name* がサポートされていることを確認してください。その場合は、オプショングループの設定をすべて確認してから再試行してください。	詳細については、「オプショングループを使用する」を参照してください。
設定変更	RDS-EVENT-0011	DBParameterGroup の##を使用するように更新されました。	なし
設定変更	RDS-EVENT-0012	変更をデータベースインスタンスクラスに適用しています。	なし
設定変更	RDS-EVENT-0014	インスタンスクラスへの変更適用が終了しました。	なし
設定変更	RDS-EVENT-0016	マスター認証情報をリセットします。	なし
設定変更	RDS-EVENT-0017	割り当てられたストレージへの変更適用が終了しました。	なし

Category	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0018	割り当てられたストレージに変更を適用します。	なし
設定変更	RDS-EVENT-0024	マルチ AZ DB インスタンスに変換するために、変更を適用します。	なし
設定変更	RDS-EVENT-0025	マルチ AZ DB インスタンスに変換するための変更適用が終了しました。	なし
設定変更	RDS-EVENT-0028	自動バックアップを無効化しました。	なし
設定変更	RDS-EVENT-0029	標準 (シングル AZ) DB インスタンスに変換するための変更適用が終了しました。	なし
設定変更	RDS-EVENT-0030	標準 (シングル AZ) DB インスタンスに変換するために、変更を適用します。	なし
設定変更	RDS-EVENT-0032	自動バックアップを有効化しました。	なし
設定変更	RDS-EVENT-0033	マスターユーザー名に一致するユーザーが *number* 人います。特定のホストに関連付けられていないユーザーのみがリセットされます。	なし
設定変更	RDS-EVENT-0067	パスワードをリセットできません。エラー情報: *message*。	なし

Category	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0078	モニタリング間隔が *number* に変更されました。	拡張モニタリングの設定が変更されました。
設定変更	RDS-EVENT-0092	DB パラメータグループの更新が完了しました。	なし
設定変更	RDS-EVENT-0217	オートスケーリングによってスタートされた変更を割り当てられたストレージに適用します。	なし
設定変更	RDS-EVENT-0218	割り当てられたストレージへのオートスケーリングによってスタートされた変更の適用が終了しました。	なし
設定変更	RDS-EVENT-0295	ストレージ設定のアップグレードが開始されました。	なし
設定変更	RDS-EVENT-0296	ストレージ設定のアップグレードが完了しました。	なし
設定変更	RDS-EVENT-0332	専用ログボリュームは無効になっています。	詳細については、「専用ログボリューム (DLV)」を参照してください。
設定変更	RDS-EVENT-0333	専用ログボリュームの無効化が開始されました。	詳細については、「専用ログボリューム (DLV)」を参照してください。
設定変更	RDS-EVENT-0334	専用ログボリュームの有効化が開始されました。	詳細については、「専用ログボリューム (DLV)」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0335	専用ログボリュームが有効になっています。	詳細については、「専用ログボリューム (DLV)」を参照してください。
設定変更	RDS-EVENT-0383	######### は memcached プラグインをサポートしていません。RDS は DB インスタンスのアップグレードを続行し、このプラグインを削除します。	MySQL 8.3.0 以降、memcached プラグインはサポートされていません。詳細については、「MySQL 8.3.0 での変更点 (2024-01-16、イノベーションリリース)」を参照してください。
作成	RDS-EVENT-0005	DB インスタンスが作成されました。	なし
削除	RDS-EVENT-0003	DB インスタンスが削除されました。	なし
フェイルオーバー	RDS-EVENT-0013	マルチ AZ インスタンスのフェイルオーバーが開始されました。	マルチ AZ フェイルオーバーがスタートされました。このフェイルオーバーにより、スタンバイ DB インスタンスの昇格が行われました。
フェイルオーバー	RDS-EVENT-0015	マルチ AZ フェイルオーバーからスタンバイへのマルチ AZ フェイルオーバーが完了しました。	マルチ AZ フェイルオーバーが完了しました。このフェイルオーバーにより、スタンバイ DB インスタンスの昇格が行われました。DNS によって新しいプライマリ DB インスタンスへの転送が行われるまで数分かかることがあります。

Category	RDS イベント ID	メッセージ	注意事項
フェイルオーバー	RDS-EVENT-0034	データベースインスタンスでフェイルオーバーを実行できません。これは、データベースインスタンスでフェイルオーバーが最近発生したためです。	Amazon RDS はリクエストされたフェイルオーバーを実行できません。これは、DB インスタンスでフェイルオーバーが最近発生したためです。
フェイルオーバー	RDS-EVENT-0049	マルチ AZ インスタンスのフェイルオーバーが完了しました。	なし
フェイルオーバー	RDS-EVENT-0050	マルチ AZ インスタンスのアクティベーションが開始されました。	マルチ AZ アクティベーションが、正常な DB インスタンスの復旧後に開始されました。このイベントは、Amazon RDS がプライマリ DB インスタンスを以前のプライマリ DB インスタンスと同じ AZ に昇格させる場合に発生します。
フェイルオーバー	RDS-EVENT-0051	マルチ AZ インスタンスのアクティベーションが完了しました。	マルチ AZ アクティベーションが完了しました。データベースがアクセス可能になりました。
フェイルオーバー	RDS-EVENT-0065	部分的なフェイルオーバーから回復しました。	なし
失敗	RDS-EVENT-0031	DB インスタンスが *name* 状態になりました。RDS は、ポイントインタイム復元を開始することを推奨します。	DB インスタンスは、互換性のない設定または基本的なストレージの問題により失敗しました。DB インスタンスのポイントインタイムの復元をスタートします。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0035	データベースインスタンスが *state* になりました。*message*。	DB インスタンスに無効なパラメータがあります。例えば、メモリ関連のパラメータがこのインスタンスクラスには高すぎる値に設定されていて、DB インスタンスをスタートできなかった場合、メモリのパラメータを変更して、DB インスタンスを再起動してください。
失敗	RDS-EVENT-0036	データベースインスタンスが *state* です。*message*。	DB インスタンスが互換性のないネットワーク上にあります。指定したサブネット ID の一部は無効であるか、存在しません。
失敗	RDS-EVENT-0058	Statspack のインストールに失敗しました。*message*。	Oracle Statspack ユーザーアカウント PERFSTAT の作成中にエラーが発生しました。STATSPACK オプションを追加する前に、アカウントを削除してください。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0079	Amazon RDS は拡張モニタリング用の認証情報を作成できなかったため、この機能は無効になっています。これは、アカウントに rds-monitoring-role が存在せず、正しく設定されていないことが原因と考えられます。詳細については、Amazon RDS ドキュメントのトラブルシューティングセクションを参照してください。	拡張モニタリングは、拡張モニタリング IAM ロールを使用してのみ有効にすることができます。IAM ロールの作成の詳細については、「Amazon RDS 拡張モニタリング用の IAM ロールを作成するには」を参照してください。
失敗	RDS-EVENT-0080	Amazon RDS がインスタンス:*name* に拡張モニタリングを設定できなかったため、この機能は無効になっています。これは、アカウントに rds-monitoring-role が存在せず、正しく設定されていないことが原因と考えられます。詳細については、Amazon RDS ドキュメントのトラブルシューティングセクションを参照してください。	拡張モニタリングは、設定変更中のエラーのため無効になりました。拡張モニタリング IAM ロールが正しく設定されていない可能性があります。拡張モニタリング IAM ロールの作成については、「Amazon RDS 拡張モニタリング用の IAM ロールを作成するには」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0081	Amazon RDS は *name* オプションの認証情報を作成できませんでした。これは、アカウントで *name* IAM ロールが正しく設定されていないことが原因です。詳細については、Amazon RDS ドキュメントのトラブルシューティングセクションを参照してください。	Amazon S3 バケットにアクセスして SQL Server のネイティブバックアップと復元を行うための IAM ロールが正しく設定されていません。詳細については、「ネイティブバックアップおよび復元のセットアップ」を参照してください。
失敗	RDS-EVENT-0165	RDS Custom DB インスタンスは、サポート範囲外にあります。	RDS Custom DB インスタンスを`unsupported-configuration` 状態にする設定上の問題は、お客様の責任で解決していただく必要があります。問題がAWSインフラストラクチャであれば、コンソールやAWS CLIを使用して修正できます。OSまたはデータベースの設定に問題がある場合は、ホストにログインして修正できます。 詳細については、「RDS Custom サポート範囲」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0188	DB インスタンスはアップグレードできない状態です。*message*。	データディクショナリに関連する非互換性があるため、Amazon RDS は MySQL DB インスタンスをアップグレードできませんでした。DB インスタンスは、バージョン 8.0 へのアップグレードが失敗したために MySQL バージョン 5.7 にロールバックされたか、バージョン 8.4 へのアップグレードが失敗したために MySQL バージョン 8.0 にロールバックされました。詳細については、「アップグレードに失敗した後のロールバック」を参照してください。
失敗	RDS-EVENT-0219	DB インスタンスは無効状態です。操作は必要ありません。オートスケーリングは後で再試行されます。	なし
失敗	RDS-EVENT-0220	DB インスタンスは、24 時間以内に許可されるストレージ変更の最大数に達しました。操作は必要ありません。DB インスタンスが別のストレージ変更の対象になると、自動スケーリングが再試行されます。	なし

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0223	ストレージのオートスケーリングは、次の理由でストレージをスケーリングできません: *reason*。	なし
失敗	RDS-EVENT-0224	ストレージのオートスケーリングにより、保留中のスケールストレージタスクの最大ストレージしきい値到達または超過がトリガーされました。最大ストレージしきい値を増やします。	なし
失敗	RDS-EVENT-0237	DB インスタンスのストレージタイプが、アベイラビリティーゾーンで現在使用できないタイプです。オートスケーリングは後で再試行されます。	なし
失敗	RDS-EVENT-0254	このカスタマーアカウントの基礎となるストレージクォータが制限を超えました。インスタンスでスケーリングを実行できるように、許容されるストレージクォータを増やしてください。	なし
失敗	RDS-EVENT-0278	DB インスタンスを作成できませんでした。#####	#####には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0279	RDS カスタムリードレプリカのプロモーションが失敗しました。#####	#####には、障害に関する詳細が含まれます。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0280	事前チェックが失敗したため、RDS Custom は DB インスタンスをアップグレードできませんでした。#####	#####には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0281	事前チェックが失敗したため、RDS Custom は DB インスタンスを修正できませんでした。#####	#####には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0282	Elastic IP 権限が正しくないため、RDS Custom は DB インスタンスを修正できませんでした。Elastic IP アドレスに AWSRDSCustom がタグ付けされていることを確認してください。	なし
失敗	RDS-EVENT-0283	アカウントの Elastic IP の上限に達したため、RDS Custom は DB インスタンスを修正できませんでした。未使用の Elastic IP をリリースするか、Elastic IP アドレス制限のクォータ増加をリクエストしてください。	なし
失敗	RDS-EVENT-0284	事前チェックが失敗したため、RDS Custom はインスタンスを高可用性に変換できませんでした。#####	#####には、障害に関する詳細が含まれます。

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0285	*#####*が原因で RDS Custom は DB インスタンスの最終スナップショットを作成できませんでした。	*#####*には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0421	RDS Custom が DB インスタンスをマルチ AZ 配置に変換できませんでした: *message*。インスタンスはシングル AZ デプロイのままになります。RDS Custom for Oracle のマルチ AZ 配置については、RDS ユーザーガイドを参照してください。	*#####*には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0306	ストレージ設定のアップグレードに失敗しました。アップグレードを再試行してください。	なし
失敗	RDS-EVENT-0315	互換性のないネットワークデータベース *name* を、「使用可能」ステータスへ移動できません:*#####*	データベースネットワーク設定が無効です。データベースを互換性のないネットワークから使用可能なネットワークに移動できませんでした。
失敗	RDS-EVENT-0328	ホストをドメインに結合できませんでした。インスタンス *instancename* のドメインメンバーシップステータスが Failed に設定されました。	なし

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0329	ホストをドメインに結合できませんでした。ドメイン結合プロセス中に、Microsoft Windows はエラーコード *message* を返しました。ネットワークとアクセス許可の設定を確認し、ドメイン参加を再試行する `modify-db-instance` リクエストを発行します。	セルフマネージド Active Directory を使用する場合は、「セルフマネージド Active Directory のトラブルシューティング」を参照してください。
失敗	RDS-EVENT-0353	リソース制限が不十分なため、DB インスタンスを作成できません。*#####*。	*#####*には、障害に関する詳細が含まれます。
失敗	RDS-EVENT-0356	RDS はドメインの Kerberos エンドポイントを設定できませんでした。これにより、DB インスタンスの Kerberos 認証が妨げられる可能性があります。DB インスタンスとドメインコントローラー間のネットワーク設定を確認してください。	なし

Category	RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0418	Amazon RDS がデータベースインスタンス *name* の KMS 暗号化キーにアクセスできません。キーが無効になっているか、Amazon RDS がキーにアクセスできないことが原因と考えられます。この状態が続くと、データベースはアクセスできない状態になります。詳細については、Amazon RDS ドキュメントのトラブルシューティングセクションを参照してください。	なし
失敗	RDS-EVENT-0420	Amazon RDS がデータベースインスタンス *name* の KMS 暗号化キーに正常にアクセスできるようになりました。	なし
ストレージの減少	RDS-EVENT-0007	割り当てられたストレージが使い果たされました。解決するには追加のストレージを割り当ててください。	DB インスタンスに割り当てられたストレージが消費されました。この問題を解決するには、DB インスタンスに追加のストレージを割り当てます。詳細については、「RDS に関数するよくある質問」を参照してください。[空きストレージ容量] メトリクスを使用して、DB インスタンスのストレージ容量をモニタリングできます。

Category	RDS イベント ID	メッセージ	注意事項
ストレージの減少	RDS-EVENT-0089	DB インスタンス: *name* の空きストレージ容量が、プロビジョニングされたストレージ [プロビジョニングされたストレージ: *size*、空きストレージ: *size*] が *percentage* と残りわずかです。この問題に対処するために、プロビジョニングされたストレージを増やす必要がある場合があります。	DB インスタンスは割り当てられたストレージの 90% 以上を使用しています。[Free Storage Space] メトリクスを使用して、DB インスタンスのストレージ容量をモニタリングできます。
ストレージの減少	RDS-EVENT-0227	Aurora クラスターのストレージは *amount* テラバイトしか残っておらず、非常に危険です。クラスターのストレージ負荷を軽減するための対策を講じてください。	Aurora ストレージサブシステムの容量が不足しています。
メンテナンス	RDS-EVENT-0026	DB インスタンスへオフラインパッチを適用しています。	DB インスタンスのオフラインメンテナンスが実行中です。現在、DB インスタンスは利用できません。
メンテナンス	RDS-EVENT-0027	DB インスタンスへのオフラインパッチ適用が終了しました。	DB インスタンスのオフラインメンテナンスが完了しました。現在、DB インスタンスは利用できます。
メンテナンス	RDS-EVENT-0047	データベースインスタンスにパッチが適用されました。	なし

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0155	DB インスタンスで DB エンジンのマイナーバージョンをアップグレードできます。	なし
メンテナンス	RDS-EVENT-0178	データベースインスタンスのアップグレードが進行中です。	なし
メンテナンス	RDS-EVENT-0264	DB エンジンバージョンアップグレードの事前チェックが開始されました。	なし
メンテナンス	RDS-EVENT-0265	DB エンジンバージョンアップグレードの事前チェックが終了しました。	なし
メンテナンス	RDS-EVENT-0266	DB インスタンスのダウンタイムが開始されました。	なし
メンテナンス	RDS-EVENT-0267	エンジンバージョンのアップグレードが開始されました。	なし
メンテナンス	RDS-EVENT-0268	エンジンバージョンのアップグレードが完了しました。	なし
メンテナンス	RDS-EVENT-0269	アップグレード後のタスクが進行中です。	なし
メンテナンス	RDS-EVENT-0270	DB エンジンバージョンアップグレードが失敗しました。エンジンバージョンアップグレードのロールバックが成功しました。	なし

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0398	プライマリ DB インスタンスで DB エンジンのバージョンアップグレードが終了するのを待機しています。	エンジンバージョンのメジャーアップグレード中にリードレプリカで出力されます。
メンテナンス	RDS-EVENT-0399	リードレプリカで DB エンジンのバージョンアップグレードが終了するのを待機しています。	エンジンバージョンのメジャーアップグレード中にソース DB エンジンで出力されます。
メンテナンス	RDS-EVENT-0422	保留中のメンテナンスアクションにより、RDS は DB インスタンス *name* のホストを置き換えます。	なし
メンテナンス、障害	RDS-EVENT-0195	#####	Oracle タイムゾーンファイルのアップデートに失敗しました。詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス、通知	RDS-EVENT-0191	タイムゾーンファイルの新しいバージョンにアップデートできます。	RDS for Oracle DB エンジンを更新する際に、タイムゾーンファイルのアップグレードを選択せず、データベースでインスタンスの使用可能な最新 DST タイムゾーンファイルを使用していない場合は、Amazon RDS がこのイベントを生成します。詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。
メンテナンス、通知	RDS-EVENT-0192	タイムゾーンファイルのアップデートが開始されました。	Oracle タイムゾーンファイルのアップグレードがスタートされました。詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス、通知	RDS-EVENT-0193	タイムゾーンファイルの新しいバージョンへのアップデートがありません。	Oracle DB インスタンスで最新のタイムゾーンファイルバージョンが使用されており、次のいずれかに当てはまります: • 最近、TIMEZONE_FILE_AUTOUPGRADE オプションを追加した。 • Oracle DB エンジンのアップグレード中。 詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。
メンテナンス、通知	RDS-EVENT-0194	タイムゾーンファイルのアップデートが終了しました。	Oracle タイムゾーンファイルのアップデートが完了しました。詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。
通知	RDS-EVENT-0044	#####	これはオペレーターが発行する通知です。詳細については、イベントメッセージを参照してください。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0048	このインスタンスには最初にアップグレードする必要のあるリードレプリカがあるため、データベースエンジンのアップグレードを遅らせます。	DB インスタンスへのパッチ適用が遅れました。
通知	RDS-EVENT-0054	#####	お使いの MySQL のストレージエンジンは InnoDB ではありません。InnoDB は、Amazon RDS 向けの MySQL のストレージエンジンとして推奨されています。MySQL ストレージエンジンの詳細については、「RDS for MySQL のサポートされているストレージエンジン」を参照してください。
通知	RDS-EVENT-0055	#####	DB インスタンス用のテーブルの数が、Amazon RDS の推奨ベストプラクティスを超えています。DB インスタンスのテーブルの数を減らします。推奨ベストプラクティスについては、「Amazon RDS の基本的な操作のガイドライン」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0056	*#####*	DB インスタンス用のデータベースの数が、Amazon RDS の推奨ベストプラクティスを超えています。DB インスタンスのデータベースの数を減らします。推奨ベストプラクティスについては、「Amazon RDS の基本的な操作のガイドライン」を参照してください。
通知	RDS-EVENT-0064	TDE 暗号化キーは正常にローテーションされました。	推奨ベストプラクティスについては、「Amazon RDS の基本的な操作のガイドライン」を参照してください。
通知	RDS-EVENT-0084	DB インスタンスをマルチ AZ に変換できません: *message*。	DB インスタンスをマルチ AZ に変換しようとしましたが、このインスタンス内のインメモリファイルグループはマルチ AZ でサポートされていません。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。
通知	RDS-EVENT-0087	DB インスタンスが停止しました。	なし
通知	RDS-EVENT-0088	DB インスタンスが開始されました。	なし

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0154	停止中の最大許容時間を超えたため、DB インスタンスが起動されています。	なし
通知	RDS-EVENT-0157	DB インスタンスクラスを変更できません。*message*。	ターゲットインスタンスクラスでは、ソース DB インスタンスに存在するデータベースの数はサポートされていないため、RDS で DB インスタンスクラスを変更することはできません。エラーメッセージは次のように表示されます: "The instance has N databases , but after conversion it would only support N"。詳細については、「Microsoft SQL Server DB インスタンスの制限」を参照してください。
通知	RDS-EVENT-0158	データベースインスタンスはアップグレードできない状態です: *message*。	なし
通知	RDS-EVENT-0167	#####	RDS Custom サポートの周辺構成が変更されました。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0189	RDS データベースインスタンスの gp2 バーストバランスクレジットが少なくなっています。この問題を解決するには、IOPS の使用量を減らすか、ストレージ設定を変更してパフォーマンスを向上させます。	RDS データベースインスタンスの gp2 バーストバランスクレジットが少なくなっています。この問題を解決するには、IOPS の使用量を減らすか、ストレージ設定を変更してパフォーマンスを向上させます。詳細については、「I/O クレジットおよびバーストパフォーマンス」の「Amazon Elastic Compute Cloud ユーザーガイド」を参照してください。
通知	RDS-EVENT-0225	割り当てられたストレージサイズ *amount* GB が最大ストレージしきい値 *amount* GB に近づいています。最大ストレージしきい値を増やします。	このイベントは、割り当てられたストレージが最大ストレージしきい値の 80% に達したときに呼び出されます。イベントを回避するには、最大ストレージしきい値を増やします。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0231	DB インスタンスのストレージの変更で内部エラーが発生しました。変更リクエストは保留中であり、後で再試行されます。	リードレプリケーションプロセスでエラーが発生しました。詳細については、イベントメッセージを参照してください。 さらに、DB エンジンのリードレプリカのトラブルシューティングのセクションを参照してください。 • MariaDB リードレプリカの問題のトラブルシューティング • SQL Server リードレプリカの問題のトラブルシューティング • MySQL リードレプリカに関する問題のトラブルシューティング • RDS for Oracle レプリカのトラブルシューティング

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0253	データベースは、二重書き込みバッファを使用しています。*message*。詳細については、*name* ドキュメントの「RDS Optimized Writes」を参照してください。	RDS 最適化書き込みは、インスタンスのストレージ構成と互換性がありません。詳細については、「RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上」および「Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上」を参照してください。 ブルー/グリーンデプロイを作成することで、ストレージ設定のアップグレードを実行して Optimized Writes を有効にできます。
通知	RDS-EVENT-0297	DB インスタンス#のストレージ設定は、16384 GiB の最大サイズをサポートします。16384 GiB を超えるストレージサイズをサポートするには、ストレージ設定のアップグレードを実行します。	DB インスタンスに割り当てられたストレージサイズを 16384 GiB を超えて増やすことはできません。この制約事項を克服するには、ストレージ設定をアップグレードしてください。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0298	DB インスタンス*#*のストレージ設定は、2048 GiB の最大テーブルサイズをサポートします。2048 GiB を超えるテーブルサイズをサポートするようにストレージ設定のアップグレードを実行します。	この制約事項がある RDS MySQL インスタンスと MariaDB インスタンスは、テーブルサイズが 2048 GiB を超えることはできません。この制約事項を克服するには、ストレージ設定をアップグレードしてください。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。
通知	RDS-EVENT-0327	Amazon RDS はシークレット *SECRET ARN* を見つけることができませんでした。*message*。	なし
通知	RDS-EVENT-0365	タイムゾーンファイルが更新されました。変更を有効にするために RDS インスタンスを再起動します。	なし
通知	RDS-EVENT-0385	クラスタートポロジーが更新されました。	DB インスタンスの DB クラスターに DNS の変更があります。これには、新しい DB インスタンスが追加または削除された場合、またはフェイルオーバーが発生した場合が含まれます。

Category	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0403	データベースワークロードにより、システムのメモリが著しく不足しています。この問題を軽減するために、RDS は innodb_buffer_pool_size の値を自動的に *amount* に設定します。	RDS for MySQL および RDS for MariaDB DB インスタンスにのみ適用されます。
通知	RDS-EVENT-0404	データベースワークロードにより、システムのメモリが著しく不足しています。この問題を軽減するために、RDS は shared_buffers の値を自動的に *amount* に設定します。	RDS for PostgreSQL DB インスタンスにのみ適用されます。
リードレプリカ	RDS-EVENT-0045	レプリケーションが停止されました。	このメッセージは、レプリケーション中にエラーが発生した場合に表示されます。エラーのタイプを特定するには、「MySQL リードレプリカに関する問題のトラブルシューティング」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
リードレプリカ	RDS-EVENT-0046	リードレプリカのレプリケーションが再開されました。	このメッセージは、初期にリードレプリカを作成したとき、またはレプリケーションが適切に機能していることを確認するモニタリングメッセージとして表示されます。このメッセージが RDS-EVENT-0045 通知の後に表示される場合は、エラーの後またはレプリケーションが停止した後で、レプリケーションが再開されました。
リードレプリカ	RDS-EVENT-0057	レプリケーションストリーミングは終了しました。	なし
リードレプリカ	RDS-EVENT-0062	リードレプリカのレプリケーションが手動で停止されました。	なし
リードレプリカ	RDS-EVENT-0063	RDS インスタンス以外のインスタンスからのレプリケーションがリセットされました。	なし
リードレプリカ	RDS-EVENT-0202	リードレプリカの作成に失敗しました。	なし
リードレプリカ	RDS-EVENT-0233	リードレプリカへのスイッチオーバーが開始されました。	なし

Category	RDS イベント ID	メッセージ	注意事項
リードレプリカ	RDS-EVENT-0357	レプリケーションチャネル#が開始されました。	レプリケーションチャネルについては、「the section called “マルチソースレプリケーションの設定”」を参照してください。
リードレプリカ	RDS-EVENT-0358	レプリケーションチャネル#が停止されました。	レプリケーションチャネルについては、「the section called “マルチソースレプリケーションの設定”」を参照してください。
リードレプリカ	RDS-EVENT-0359	レプリケーションチャネル#が手動で停止されました。	レプリケーションチャネルについては、「the section called “マルチソースレプリケーションの設定”」を参照してください。
リードレプリカ	RDS-EVENT-0360	レプリケーションチャネル#がリセットされました。	レプリケーションチャネルについては、「the section called “マルチソースレプリケーションの設定”」を参照してください。
リードレプリカ	RDS-EVENT-0415	アップグレードプロセスがリードレプリカでのレプリケーションを再開しました。	なし
リードレプリカ	RDS-EVENT-0416	アップグレードプロセスがリードレプリカのでレプリケーションを停止しました。	なし

Category	RDS イベント ID	メッセージ	注意事項
復旧	RDS-EVENT-0020	DB インスタンスの復旧がスタートされました。復旧時間は、復旧するデータの量に応じて変わります。	なし
復旧	RDS-EVENT-0021	DB インスタンスの復旧が完了しました。	なし
復旧	RDS-EVENT-0023	緊急スナップショットリクエスト: *message*。	手動バックアップがリクエストされましたが、現在、Amazon RDS は DB スナップショットの作成中です。Amazon RDS で DB スナップショットの作成が完了した後で、リクエストをもう一度送信してください。
復旧	RDS-EVENT-0052	マルチ AZ インスタンスの復旧が開始されました。	復旧時間は、復旧するデータの量に応じて変わります。
復旧	RDS-EVENT-0053	マルチ AZ インスタンスの復旧が完了しました。フェイルオーバーまたはアクティベーションが保留中です。	このメッセージは、Amazon RDS が、必要に応じてセカンダリインスタンスへのフェイルオーバーを開始するように DB インスタンスを準備したことを示します。

Category	RDS イベント ID	メッセージ	注意事項
復旧	RDS-EVENT-0066	ミラーリングの再確立中にインスタンスのパフォーマンスが低下します: *message*。	SQL Server DB インスタンスは、ミラーを再構築しています。ミラーが再構築されるまで、パフォーマンスが低下します。FULL ではない復旧モデルのデータベースが見つかりました。復旧モデルは FULL に戻され、ミラーリングによる復旧がスタートされました。(<dbname>: <recovery model found>[,…])”
復旧	RDS-EVENT-0166	#####	RDS Custom DB インスタンスは、サポート境界内にあります。
復旧	RDS-EVENT-0361	スタンバイ DB インスタンスの復旧が開始されました。	スタンバイ DB インスタンスは、復旧プロセス中に再構築されます。データベースのパフォーマンスは、復旧プロセス中に影響を受けます。
復旧	RDS-EVENT-0362	スタンバイ DB インスタンスの復旧が完了しました。	スタンバイ DB インスタンスは、復旧プロセス中に再構築されます。データベースのパフォーマンスは、復旧プロセス中に影響を受けます。
復元	RDS-EVENT-0019	DB インスタンス *name* から *name* に復元しました。	DB インスタンスが特定の時点のバックアップから復元されました。

Category	RDS イベント ID	メッセージ	注意事項
セキュリティ	RDS-EVENT-0068	hsm パーティションのパスワードを復号してインスタンスを更新します。	RDS は AWS CloudHSM パーティションパスワードを復号して、DB インスタンスを更新しています。詳細については、AWS CloudHSM ユーザーガイドの「AWS CloudHSM を使用した Oracle Database Transparent Data Encryption (TDE)」を参照してください。
セキュリティパッチ	RDS-EVENT-0230	DB インスタンスにシステムアップデートが使用可能です。アップデートの適用方法については、RDS ユーザーガイドの、「DB インスタンスのメンテナンス」を参照してください。	新しいオペレーティングシステムのアップデートが使用可能です。 DB インスタンスで、新しいマイナーバージョンのオペレーティングシステムを更新できます。更新の適用については、「RDS DB インスタンスのオペレーティングシステムの更新」を参照してください。
メンテナンス	RDS-EVENT-0425	指定したサブネットに使用可能な IP アドレスがないため、Amazon RDS は OS アップグレードを実行できません。使用可能な IP アドレスがあるサブネットを選択して、もう一度試してください。	なし

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0429	*zone* アベイラビリティーゾーンの *type* インスタンスタイプで使用可能な容量が不足しているため、Amazon RDS は OS アップグレードを実行できません	なし
メンテナンス	RDS-EVENT-0501	Amazon RDS DB インスタンスのサーバー証明書には、保留中のメンテナンスアクションによるローテーションが必要です。	DB インスタンスのサーバー証明書には、保留中のメンテナンスアクションによるローテーションが必要です。Amazon RDS は、このメンテナンス中にデータベースを再起動して、証明書のローテーションを完了します。このメンテナンスをスケジュールするには、[メンテナンスとバックアップ] タブに移動し、[今すぐ適用] または [次のメンテナンス期間の予定] を選択します。変更がスケジュールされていない場合、Amazon RDS はメンテナンスアクションに表示される自動適用日のメンテナンスウィンドウに自動的に適用します。

Category	RDS イベント ID	メッセージ	注意事項
メンテナンス	RDS-EVENT-0502	Amazon RDS は、次のメンテナンスウィンドウ中に DB インスタンスのサーバー証明書のローテーションをスケジュールしました。このメンテナンスでは、データベースを再起動する必要があります。	なし

DB パラメータグループイベント

次の表は、DB パラメータグループがソースタイプである場合のイベントカテゴリとイベントを示しています。

カテゴリ	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0037	パラメータ *name* を適用メソッド *method* で *value* に更新しました。	なし

DB セキュリティグループイベント

次の表は、DB セキュリティグループをソースタイプとするイベントカテゴリとイベントの一覧です。

> **Note**
>
> DB セキュリティグループは EC2-Classic 用リソースです。EC2-Classic は 2022 年 8 月 15 日に廃止されました。EC2-Classic から VPC に移行していない場合、できるだけ早く移行することをお勧めします。詳細については、「Amazon EC2 ユーザーガイド」の「EC2-Classic から VPC へ移行」およびブログ記事「EC2-Classic ネットワーキングがリタイア — 準備方法」を参照してください。

Category	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0038	セキュリティグループに変更を適用しました。	なし
失敗	RDS-EVENT-0039	*user* としての権限を取り消しています。	*user*.が所有するセキュリティグループが存在しません。セキュリティグループの認可が無効なため取り消されました。

DB スナップショットイベント

次の表は、DB スナップショットをソースタイプとするイベントカテゴリとイベントの一覧です。

カテゴリ	RDS イベント ID	メッセージ	注意事項
作成	RDS-EVENT-0040	手動スナップショットを作成しています。	なし
作成	RDS-EVENT-0042	手動スナップショットが作成されました。	なし
作成	RDS-EVENT-0090	自動スナップショットを作成しています。	なし
作成	RDS-EVENT-0091	自動スナップショットが作成されました。	なし
削除	RDS-EVENT-0041	ユーザースナップショットを削除しました。	なし
通知	RDS-EVENT-0059	スナップショット*name* からリージョン *name* へのコピーを開始しました。	これは、クロスリージョンでのスナップショットのコピーです。

カテゴリ	RDS イベント ID	メッセージ	注意事項
通知	RDS-EVENT-0060	スナップショット *name* からリージョン *name* へのコピーを *number* 分間で終了しました。	これは、クロスリージョンでのスナップショットのコピーです。
通知	RDS-EVENT-0061	リージョン *name*からスナップショット *name* のコピーリクエストをキャンセルしました。	これは、クロスリージョンでのスナップショットのコピーです。
通知	RDS-EVENT-0159	スナップショットのエクスポートタスクに失敗しました。	なし
通知	RDS-EVENT-0160	スナップショットのエクスポートタスクがキャンセルされました。	なし
通知	RDS-EVENT-0161	スナップショットのエクスポートタスクが完了しました。	なし
通知	RDS-EVENT-0196	リージョン *name* のスナップショット*name* のコピーを開始しました。	これはローカルでのスナップショットのコピーです。
通知	RDS-EVENT-0197	リージョン *name* のスナップショット*name* のコピーを終了しました。	これはローカルでのスナップショットのコピーです。
通知	RDS-EVENT-0190	リージョン *name* のスナップショット *name* のコピーリクエストをキャンセルしました。	これはローカルでのスナップショットのコピーです。

カテゴリ	RDS イベント ID	メッセージ	注意事項
復元	RDS-EVENT-0043	スナップショット *name* から復元しました。	DB インスタンスが DB スナップショットから復元されます。
失敗	RDS-EVENT-0484	エクスポートタスク##: ##が原因でテーブル##のテーブル抽出に失敗しました。	このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0486	エクスポートタスク##: テーブル##は低速のシングルスレッドモードでエクスポートされます。エクスポートの完了に遅延が発生することが予想されます。	このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0487	エクスポートタスク##: テーブル##は進行中で、エクスポートサイズは ## GB です。	このイベントは、完了までに 1 時間以上かかるテーブルに対してのみ送信されます。 このイベントは、高速エクスポートではサポートされていません。
通知	RDS-EVENT-0488	エクスポートタスク##は##ステージにあります。	なし
		エクスポートタスク*name*: エクスポートの進行状況は *number*% で、*number* テーブルが進行中、*number* テーブルが完了しています。完了したテーブルの合計サイズは ## GB です。	このイベントは、高速エクスポートではサポートされていません。

RDS Proxy イベント

ソースタイプが RDS Proxy である場合の、イベントのカテゴリとその一覧を次の表に示します。

Category	RDS イベント ID	メッセージ	注意事項
設定変更	RDS-EVENT-0204	RDS が DB プロキシ *name* を変更しました。	なし
設定変更	RDS-EVENT-0207	RDS において、DB プロキシ *name* のエンドポイントが修正されました。	なし
設定変更	RDS-EVENT-0213	RDS が DB インスタンスの追加を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループに自動的に追加しました。	なし
設定変更	RDS-EVENT-0214	RDS が DB インスタンス *name* の削除を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループ *name* から自動的に削除しました。	なし
設定変更	RDS-EVENT-0215	RDS が DB クラスター *name* の削除を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループ *name* から自動的に削除しました。	なし
作成	RDS-EVENT-0203	RDS は DB プロキシ *name* を作成しました。	なし

Category	RDS イベント ID	メッセージ	注意事項
作成	RDS-EVENT-0206	RDS は DB プロキシ *name* のエンドポイント*name*を作成しました。	なし
削除	RDS-EVENT-0205	RDS は DB プロキシ *name* を削除しました。	なし
削除	RDS-EVENT-0208	RDS は DB プロキシ*name*のエンドポイント *name* を削除しました。	なし
失敗	RDS-EVENT-0243	サブネット *name* に十分な IP アドレスがないため、RDS はプロキシの容量をプロビジョニングできませんでした: *name*。この問題を解決するには、RDS プロキシのドキュメントで推奨されているように、サブネットの未使用の IP アドレスが最小限であることを確認してください。	インスタンスクラスの推奨数を決定するには、「IP アドレス容量の計画」を参照してください。
失敗	RDS-EVENT-0275	RDS は DB プロキシ#への一部の接続をスロットリングしました。クライアントからプロキシへの同時接続リクエストの数が制限を超えました。	なし

ブルー/グリーンデプロイイベント

次の表は、ブルー/グリーンデプロイがソースタイプである場合のイベントカテゴリとイベントのリストを示します。

ブルー/グリーンデプロイの詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

Category	Amazon RDS イベント ID	メッセージ	注意事項
作成	RDS-EVENT-0244	ブルー/グリーンデプロイタスクが完了しました。グリーン環境のデータベースにさらに変更を加えたり、デプロイを切り替えたりできます。	なし
失敗	RDS-EVENT-0245	*reason* が原因でブルー/グリーンデプロイの作成に失敗しました。	なし
削除	RDS-EVENT-0246	ブルー/グリーンデプロイが削除されました。	なし
通知	RDS-EVENT-0247	###から####へのスイッチオーバーが開始されました。	なし
通知	RDS-EVENT-0248	ブルー/グリーンデプロイの切り替えが完了しました。	なし
失敗	RDS-EVENT-0249	ブルー/グリーンデプロイの切り替えがキャンセルされました。	なし
通知	RDS-EVENT-0250	プライマリ/リードレプリカの###から####へのスイッチオーバーが開始されました。	なし
通知	RDS-EVENT-0251	プライマリ/リードレプリカの###から####へのスイッチオーバーが完了しま	なし

Category	Amazon RDS イベント ID	メッセージ	注意事項
		した。###から#####、####から###に名前を変更しました。	
失敗	RDS-EVENT-0252	プライマリ/リードレプリカ の###から####へのスイッチオーバーは、##によりキャンセルされました。	なし
通知	RDS-EVENT-0307	の###から####へのスイッチオーバーのシーケンス同期が開始されました。シーケンス使用時にスイッチオーバーを行うと、ダウンタイムが長くなる可能性があります。	なし
通知	RDS-EVENT-0308	の###から####へのスイッチオーバーのためのシーケンス同期が完了しました。	なし
失敗	RDS-EVENT-0310	の###から####へのスイッチオーバーのシーケンス同期は、シーケンスが同期に失敗したためキャンセルされました。	なし
通知	RDS-EVENT-0405	ストレージボリュームを初期化中です。	なし
通知	RDS-EVENT-0406	ストレージボリュームが初期化されました。	なし
通知	RDS-EVENT-0409	#####	なし

カスタムエンジンバージョンイベント

次の表は、カスタムエンジンバージョンがソースタイプである場合のイベントカテゴリとイベントのリストを示しています。

Category	Amazon RDS イベント ID	メッセージ	注意事項
作成	RDS-EVENT-0316	カスタムエンジンバージョン#を作成する準備をしています。作成プロセスが完了までに最大 4 時間かかる場合があります。	なし
作成	RDS-EVENT-0317	カスタムエンジンバージョン#の作成。	なし
作成	RDS-EVENT-0318	カスタムエンジンバージョン#の評価。	なし
作成	RDS-EVENT-0319	カスタムエンジンバージョン#は正常に作成されました。	なし
作成	RDS-EVENT-0320	RDS は内部問題のため、カスタムエンジンバージョン#を作成できません。現在、問題に対処しており、必要に応じてご連絡いたします。さらにサポートが必要な場合は、AWS Premium Support/ にお問い合わせください。	なし
失敗	RDS-EVENT-0198	カスタムエンジンバージョン *name* の作成に失敗しました。*message*	#####には、見つからないファイルなど、障害に関する詳細が含まれます。

Category	Amazon RDS イベント ID	メッセージ	注意事項
失敗	RDS-EVENT-0277	カスタムエンジンバージョン *name* の削除中に障害が発生しました。*#####*	*#####*には、障害に関する詳細が含まれます。
復元中	RDS-EVENT-0352	ポイントインタイムリストアでサポートされる最大データベース数が変更されました。	*#####*には、イベントに関する詳細が含まれます。

Amazon RDS ログファイルのモニタリング

すべての RDS データベースエンジンは、監査やトラブルシューティング時にアクセスするログを生成します。ログの種類は、データベースエンジンによって異なります。

AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または Amazon RDS API を使用して、DB インスタンスのデータベースログにアクセスできます。トランザクションログを表示、監視、またはダウンロードすることはできません。

トピック

- データベースログファイルの表示とリスト化
- データベースログファイルのダウンロード
- データベースログファイルのモニタリング
- Amazon CloudWatch Logs へのデータベースログの発行
- REST を用いたログファイルの内容の読み取り
- Amazon RDS for Db2 データベースのログファイル
- MariaDB データベースのログファイル
- Amazon RDS for Microsoft SQL Server データベースのログファイル
- MySQL データベースのログファイル
- Amazon RDS for Oracle データベースのログファイル
- RDS for PostgreSQL データベースログファイル

データベースログファイルの表示とリスト化

AWS マネジメントコンソール を使用して、Amazon RDS DB エンジンのデータベースログファイルを表示できます。AWS CLI または Amazon RDS API を使用して、ダウンロードまたはモニタリングできるログファイルを一覧表示できます。

> **Note**
>
> 既存の RDS for Oracle DB インスタンスのログファイルのリストを表示できない場合は、インスタンスを再起動してリストを表示します。

コンソール

データベースログファイルを閲覧するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 表示するログファイルのある DB インスタンスの名前を選択します。
4. [ログとイベント] タブを選択します。
5. [ログ] セクションまで下にスクロールします。
6. (オプション) 検索語を入力して、結果をフィルタリングします。
7. 表示するログを選択してから、[View] (表示) を選択します。

AWS CLI

DB インスタンスで使用できるデータベースログファイルを一覧表示するには、AWS CLI の `describe-db-log-files` コマンドを使用します。

次の例では、DB インスタンス (`my-db-instance`) のログファイルのリストが返ります。

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

RDS API

DB インスタンスの使用可能なデータベースログファイルを一覧表示するには、Amazon RDS API の `DescribeDBLogFiles` アクションを使用します。

データベースログファイルのダウンロード

データベースログファイルをダウンロードするには、AWS マネジメントコンソール、AWS CLI、または API を使用します。

コンソール

データベースログファイルをダウンロードするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[データベース] を選択します。
3. 表示するログファイルのある DB インスタンスの名前を選択します。
4. [ログとイベント] タブを選択します。
5. [ログ] セクションまで下にスクロールします。
6. [ログ] セクションで、ダウンロードするログの横にあるボタンを選択し、[ダウンロード] を選択します。
7. 表示されたリンクのコンテキスト (右クリック) メニューを開き、[名前を付けて保存] を選択します。ログファイルを保存する場所を入力し、[保存] を選択します。

AWS CLI

データベースログファイルをダウンロードするには、AWS CLI の `download-db-log-file-portion` コマンドを使用します。デフォルトでは、このコマンドによってログファイルの最新部分のみがダウンロードされます。ただし、`--starting-token 0` パラメータを指定して、ファイル全体をダウンロードすることもできます。

以下の例では、ログファイル (log/ERROR.4) のすべての内容をダウンロードし、ローカルファイル (errorlog.txt) に格納する方法について説明します。

Example

Linux、macOS、Unix の場合:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
```

```
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Windows の場合:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

RDS API

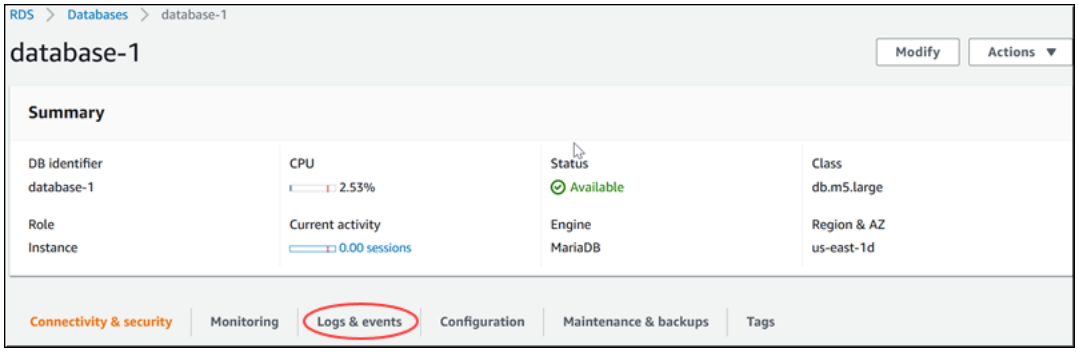
データベースログファイルをダウンロードするには、Amazon RDS API の DownloadDBLogFilePortion アクションを使用します。

データベースログファイルのモニタリング

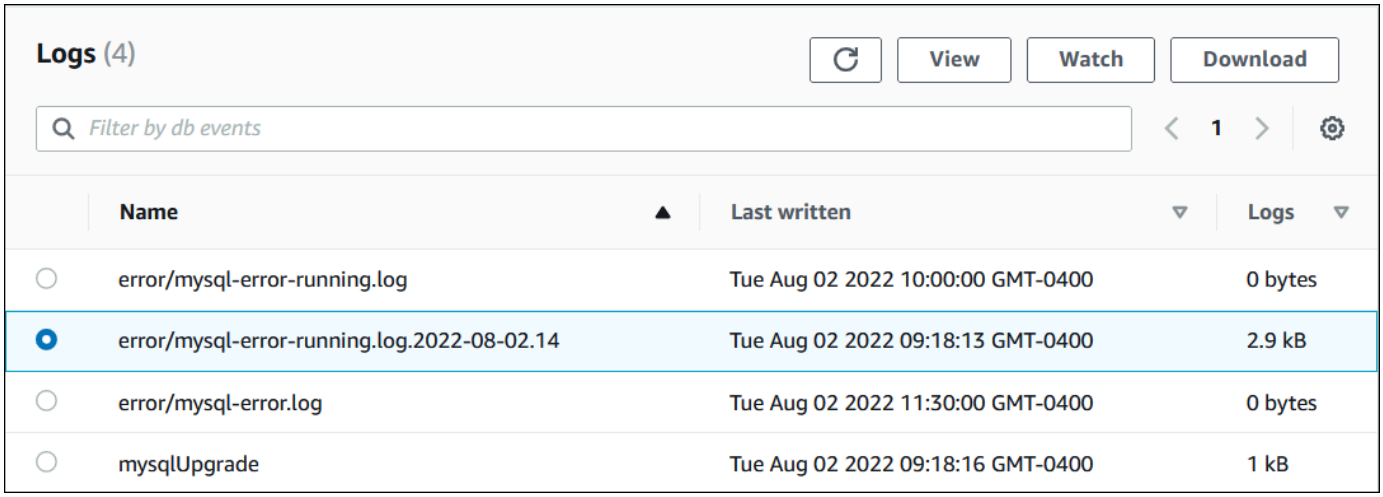
データベースログファイルを監視することは、UNIX または Linux システムでファイルをテーリングすることと同じです。AWS マネジメントコンソール を使用すると、ログファイルを監視できます。RDS は 5 秒ごとにログの末尾を更新します。

データベースログファイルをモニタリングするには

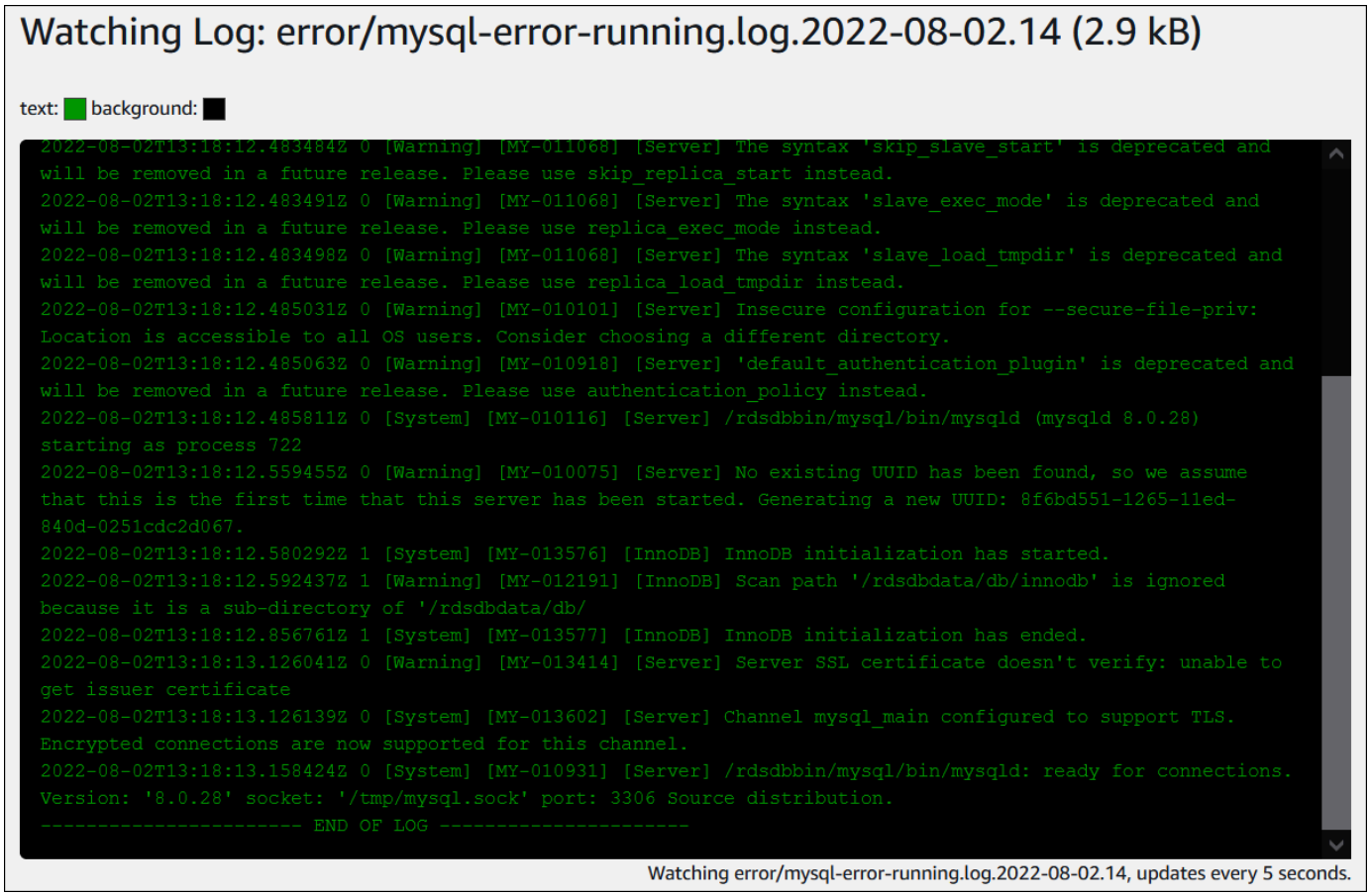
1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 表示するログファイルのある DB インスタンスの名前を選択します。
4. [ログとイベント] タブを選択します。



5. [ログ] セクションでログファイルを選択し、[モニタリング] を選択します。



RDS には、次の MySQL の例のようにログの末尾が表示されます。



Amazon CloudWatch Logs へのデータベースログの発行

オンプレミスデータベースでは、データベースログはファイルシステムに存在します。Amazon RDS では、DB インスタンスのファイルシステム上のデータベースログへのホストアクセスが許可

されません。このため、Amazon RDS では、Amazon CloudWatch Logs にデータベースログをエクスポートできます。CloudWatch Logs を使用すると、ログデータのリアルタイム分析を実行できます。高い耐久性を持つストレージにデータを保存し、CloudWatch Logs エージェントを使用したデータの管理を実行できます。

トピック

- RDS と CloudWatch Logs の統合の概要
- CloudWatch Logs に発行するログの決定
- CloudWatch Logs に発行するログの指定
- CloudWatch Logs でのログの検索とフィルタリング

RDS と CloudWatch Logs の統合の概要

CloudWatch Logs では、ログストリーミングは、同じ出典を共有する一連のログイベントです。CloudWatch Logs でのログの各ソースで各ログストリームが構成されます。ロググループは、保持、モニタリング、アクセス制御について同じ設定を共有するログストリームのグループです。

Amazon RDS は、DB インスタンスログレコードをロググループに継続的にストリームします。例えば、発行した各タイプのログについて、ロググループ `/aws/rds/instance/`*`instance_name`*`/`*`log_type`* があることを考えます。このロググループは、ログを生成するデータベースインスタンスと同じ AWS リージョンにあります。

AWS は、CloudWatch Logs に発行されたログデータを、保持期間を指定しない限り、無期限に保持します。詳細については、「CloudWatch Logs でのログデータ保管期間の変更」を参照してください。

CloudWatch Logs に発行するログの決定

各 RDS データベースエンジンは、独自のログセットをサポートします。データベースエンジンのオプションについては、以下のトピックを確認してください。

- the section called “Amazon CloudWatch Logs への Db2 ログの発行”
- the section called “MariaDB ログを Amazon CloudWatch Logs に発行する”
- the section called “Amazon CloudWatch Logs への MySQL ログの発行”
- the section called “Amazon CloudWatch Logs への Oracle ログの発行”
- the section called “Amazon CloudWatch Logs への PostgreSQL ログの発行”
- the section called “Amazon CloudWatch Logs への SQL Server ログの発行”

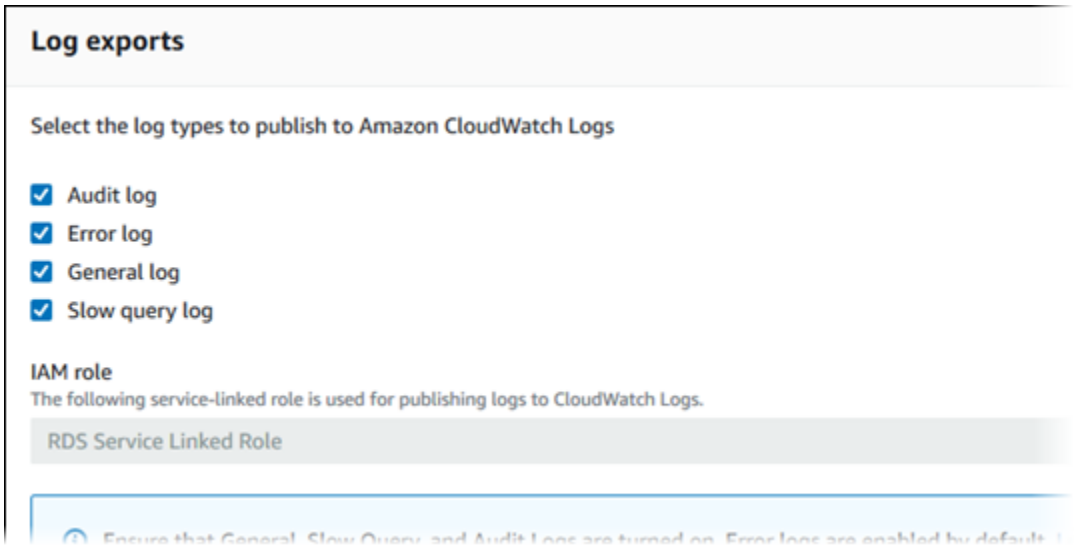
CloudWatch Logs に発行するログの指定

コンソールで発行するログを指定します。AWS Identity and Access Management (IAM) にサービスリンクロールがあることを確認します。サービスにリンクされたロールの詳細については、「Amazon RDS のサービスにリンクされたロールの使用」を参照してください。

発行するログを指定するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択します。
3. 次のいずれかを実行します。
   - [データベースの作成] を選択します。
   - 一覧からデータベースを選択し、[Modify] (変更) を選択します。
4. [Logs exports] (ログのエクスポート) で、発行するログを選択します。

   次の例では、RDS for MySQL DB インスタンスの監査ログ、エラーログ、一般ログ、、スロークエリログを指定します。



CloudWatch Logs でのログの検索とフィルタリング

CloudWatch コンソールを使用して、指定した基準を満たすログエントリを検索することができます。ログには、CloudWatch Logs コンソールにつながる RDS コンソールからアクセスすることも、CloudWatch Logs コンソールから直接アクセスすることもできます。

RDS コンソールを使用して RDS ログを検索するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択します。
3. DB インスタンスを選択します。
4. [設定] を選択します。
5. [Published logs] (発行されたログ) で、表示するデータベースログを選択します。

CloudWatch Logs コンソールを使用して RDS ログを検索するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. フィルタボックスに **`/aws/rds`** と入力します。
4. [ロググループ] で、検索するログストリームを含むロググループの名前を選択します。
5. [ログストリーム] で、検索するログストリームの名前を選択します。
6. [Log Events (ログイベント)] で、使用するフィルター構文を入力します。

詳細については、Amazon CloudWatch Logs ユーザーガイドの「ログデータの検索およびフィルタリング」を参照してください。RDS ログをモニタリングする方法を説明するブログチュートリアルについては、「Amazon CloudWatch Logs、AWS Lambda、および Amazon SNS を使用して Amazon RDS のプロアクティブなデータベースモニタリングを構築する」を参照してください。

REST を用いたログファイルの内容の読み取り

Amazon RDS では、DB インスタンスのログファイルへのアクセスを許可する REST エンドポイントを使用できます。これは、Amazon RDS ログファイルの内容を取り出すアプリケーションを作成される場合に有用です。

構文は次のとおりです。

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

以下のパラメータは必須です。

- *DBInstanceIdentifier*— ダウンロードするログファイルを含む DB インスタンスの名前。
- *LogFileName*—ダウンロードするログファイルの名前。

このレスポンスには、ストリーミングとしてリクエストされたログファイルの内容が含まれます。

次の例では、us-west-2 リージョンの sample-sql という名前の DB インスタンスの log/ERROR.6 という名前のログファイルをダウンロードします。

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

存在しない DB インスタンスを指定した場合、レスポンスは次のエラーになります。

- DBInstanceNotFound—*DBInstanceIdentifier* が既存の DB インスタンスを参照していません。(HTTP ステータスコード: 404)

Amazon RDS for Db2 データベースのログファイル

Amazon RDS コンソール、AWS CLI、または RDS API を使用して、RDS for Db2 の診断ログと通知ログにアクセスできます。ファイルベースのデータベースログの表示、ダウンロード、モニタリングの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

トピック

- 保持期間スケジュール
- Amazon CloudWatch Logs への Db2 ログの発行

保持期間スケジュール

ログファイルは、毎日、および DB インスタンスが再開されるたびにローテーションされます。以下は、Amazon RDS の RDS for Db2 ログの保持期間スケジュールです。

ログタイプ	保持期間スケジュール
診断ログ	Db2 は、インスタンスレベルで設定された保持期間に含まれないログを削除します。Amazon RDS は、`diagsize` パラメータを 1,000 に設定します。
通知ログ	Db2 は、インスタンスレベルで設定された保持期間に含まれないログを削除します。Amazon RDS は、`diagsize` パラメータを 1,000 に設定します。

Amazon CloudWatch Logs への Db2 ログの発行

RDS for Db2 では診断および通知ログイベントを直接 Amazon CloudWatch Logs に発行できます。CloudWatch Logs を使用してログデータを分析し、CloudWatch を使用してアラームを作成し、メトリクスを表示することができます。

CloudWatch Logs では、次のことを実行できます。

- ログは、ユーザーが定義する保持期間で耐久性の高いストレージ領域に保存します。
- ログデータを検索しフィルタリングします。
- アカウント間でログデータを共有します。
- ログを Amazon S3 にエクスポートします。

- Amazon OpenSearch Service へのデータのストリーミング
- Amazon Kinesis Data Streams を使用してログデータをリアルタイムで処理します。詳細については、「Amazon Managed Service for Apache Flink for Apache Flink デベロッパーガイド」内の「Amazon CloudWatch Logs の使用」を参照してください。

Amazon RDS は、各 RDS for Db2 データベースログを、ロググループ内の個別のデータベースストリームとして発行します。例えば、診断ログと通知ログを発行する場合、診断データは `/aws/rds/instance/my_instance/diagnostic` ロググループの診断ログストリームに保存され、通知ログデータは `/aws/rds/instance/my_instance/notify` ロググループに保存されます。

> **Note**
>
> RDS for Db2 ログの CloudWatch Logs への発行は、デフォルトでは有効にされていません。セルフチューニングメモリマネージャー (STMM) とオプティマイザ統計のログの発行はサポートされていません。RDS for Db2 ログの CloudWatch Logs への発行は、アジアパシフィック (香港) を除くすべてのリージョンでサポートされています。

コンソール

AWS マネジメントコンソール から CloudWatch Logs に RDS for Db2 ログを発行するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。

   diag.log、notify.log、またはその両方を選択できます。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

RDS for Db2 ログを発行するには、以下のパラメータを指定して `modify-db-instance` コマンドを使用します。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下のコマンドを使用して RDS for Db2 ログを発行することもできます。

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-to-point-in-time`

Example

次の例では、CloudWatch Logs の発行を有効にした RDS for Db2 DB インスタンスを作成します。`--enable-cloudwatch-logs-exports` 値は、`diag.log`、`notify.log`、または両方を含むことができる JSON 文字列です。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["diag.log","notify.log"]' \
    --db-instance-class db.m4.large \
    --engine db2-se
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports "[\"diag.log\",\"notify.log\"]" ^
    --db-instance-class db.m4.large ^
    --engine db2-se
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

Example

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の RDS for Db2 DB インスタンスを変更します。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `EnableLogTypes` であり、その値は `diag.log`、`notify.log`、または両方を含む文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["diag.log","notify.log"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"EnableLogTypes\":[\"diag.log\",
\"notify.log\"]}"
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

Example

次の例では、診断ログファイルの CloudWatch Logs への発行が無効になるよう既存の RDS for Db2 DB インスタンスを変更します。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `DisableLogTypes` であり、その値は `diag.log`、`notify.log`、または両方を含む文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["diag.log"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"diag.log\"]}"
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

MariaDB データベースのログファイル

MariaDB エラーログ、スロークエリログ、IAM データベース認証エラーログ、一般ログをモニタリングできます。デフォルトで、MariaDB のエラーログは生成されます。DB パラメータグループにパラメータを設定することで、スロークエリログと一般ログを生成できます。Amazon RDS はすべての MariaDB ログファイルをローテーションします。各タイプの間隔は以下のとおりです。

MariaDB ログは、Amazon RDS コンソール、Amazon RDS API、Amazon RDS CLI、または AWS SDK を通じて直接モニタリングできます。また、ログをメインデータベースのデータベーステーブルに書き込み、そのテーブルに対してクエリを実行することで、MariaDB ログにアクセスできます。mysqlbinlog ユーティリティを使用して、バイナリログをダウンロードできます。

ファイルベースのデータベースログの表示、ダウンロード、モニタリングの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

トピック

- MariaDB エラーログにアクセスする
- MariaDB のスロークエリと一般ログにアクセスする
- MariaDB ログを Amazon CloudWatch Logs に発行する
- MariaDB のログのローテーションと保持
- テーブルベースの MariaDB ログを管理する
- MariaDB バイナリログ記録の設定
- MariaDB バイナリログにアクセスする
- MariaDB バイナリログ注釈の有効化

MariaDB エラーログにアクセスする

MariaDB エラーログは `<host-name>.err` ファイルに書き込まれます。Amazon RDS コンソールを使用して、このファイルを表示できます。Amazon RDS API、Amazon RDS CLI、または AWS SDK を使用してログを取得することもできます。`<host-name>.err` ファイルは 5 分ごとにフラッシュされ、その内容は `mysql-error-running.log` に追加されます。その後、`mysql-error-running.log` ファイルは 1 時間ごとにローテーションされ、直前 24 時間内に 1 時間ごとに生成されたファイルが保持されます。各ログファイルには、それぞれ生成された時間 (UTC) がファイル名に付加されます。ログファイルには、タイムスタンプも付加され、ログエントリがいつ書き込まれたかを調べるために役立ちます。

MariaDB では、スタートアップ時、シャットダウン時、エラー検出時にのみエラーログへの書き込みが行われます。DB インスタンスでは、新しいエントリがエラーログに書き込まれないまま、数時間または数日が経過することがあります。最近のエントリがない場合、それは、サーバーにログエントリになるエラーが発生しなかったためです。

MariaDB のスロークエリと一般ログにアクセスする

MariaDB のスロークエリログと一般ログは、DB パラメータグループのパラメータを設定することで、ファイルまたはデータベーステーブルに書き込むことができます。DB パラメータグループの作成と変更の詳細については、「Amazon RDS のパラメータグループ」を参照してください。Amazon RDS コンソール、Amazon RDS API、AWS CLI、または AWS SDK を使用して、スロークエリログまたは一般ログを表示する前に、以下のパラメータを設定する必要があります。

以下のリストに示すパラメータを使用して MariaDB のログ記録を制御できます。

- `slow_query_log` または `log_slow_query`: スロークエリログを作成するには、1 に設定します。デフォルトは 0 です。
- `general_log`: 一般ログを作成するには、1 に設定します。デフォルトは 0 です。
- `long_query_time` または `log_slow_query_time`: 高速で実行されるクエリがスロークエリログに記録されないようにするために、ログに記録されるクエリの最短実行時間の値を秒単位で指定します。デフォルトは 10 秒で、最小値は 0 です。log_output = FILE の場合は、マイクロ秒の精度になるように、浮動小数点値を指定できます。log_output = TABLE の場合は、秒の精度になるように、整数値を指定する必要があります。実行時間が `long_query_time` または `log_slow_query_time` の値を超えたクエリのみがログに記録されます。例えば、`long_query_time` または `log_slow_query_time` を 0.1 に設定すると、実行時間が 100 ミリ秒未満のすべてのクエリはログに記録されなくなります。
- `log_queries_not_using_indexes`: インデックスを使用しないすべてのクエリをスロークエリログに記録するには、このパラメータを 1 に設定します。デフォルトは 0 です。インデックスを使用しないクエリは、その実行時間が `long_query_time` パラメータの値未満であってもログに記録されます。
- `log_output` *option*: `log_output` パラメータに指定できるオプションは、次のとおりです。
  - TABLE (デフォルト) - 一般クエリを `mysql.general_log` テーブルに、スロークエリを `mysql.slow_log` テーブルに書き込みます。
  - FILE - 一般クエリログとスロークエリログの両方をファイルシステムに書き込みます。ログファイルは 1 時間ごとにローテーションされます。
  - NONE - ログ記録を無効にします。

ログ記録が有効になっている場合、Amazon RDS は、テーブルログのローテーションまたはログファイルの削除を定期的に実行します。これは、ログファイルが大きくなることでデータベースが使用できなくなったりパフォーマンスに影響する可能性を低く抑えるための予防措置です。ログ記録の FILE オプションと TABLE オプションでは、ローテーションと削除が次のように行われます。

- FILE ログ記録が有効になっている場合、ログファイルの検査が 1 時間ごとに実行され、作成後 24 時間を超えた古いログファイルは削除されます。場合によっては、削除後の残りのログファイルの合計サイズが、DB インスタンスに割り当てられた領域のしきい値である 2 % を超えることがあります。この場合、ログファイルのサイズがしきい値以下になるまで、最も大きいログファイルから順に削除されます。
- TABLE ロギングを有効化すると、24 時間ごとにログテーブルのローテーションが実行される場合があります。このログテーブルのローテーションは、テーブルログに使用されている領域が、割り当てられたストレージ領域の 20 % を超えると、実行されます。結合されたすべてのログのサイズが 10 GB を超える場合にも発生します。DB インスタンスに使用されている領域が、DB インスタンスに割り当てられたストレージ領域の 90% を超えている場合は、ログのローテーションを実行するためのしきい値が小さくなります。テーブルログに使用されている領域が、割り当てられたストレージ領域の 10% を超えると、ログテーブルのローテーションが実行されます。結合されたすべてのログのサイズが 5 GB を超えると、ログはローテーションされます。

  ログテーブルのローテーションが実行されると、現在のログテーブルがバックアップのログテーブルにコピーされ、現在のログテーブル内にあるエントリは削除されます。バックアップのログテーブルが既に存在する場合は、現在のログテーブルをバックアップにコピーする前に、削除されます。バックアップのログテーブルは、必要に応じて照会することができます。mysql.general_log テーブルに対するバックアップのログテーブルは、mysql.general_log_backup という名前になります。mysql.slow_log テーブルに対するバックアップのログテーブルは、mysql.slow_log_backup という名前になります。

  mysql.general_log テーブルのローテーションは、mysql.rds_rotate_general_log プロシージャを呼び出すことで実行できます。mysql.slow_log テーブルのローテーションは、mysql.rds_rotate_slow_log プロシージャを呼び出すことで実行できます。

  データベースバージョンのアップグレード時にも、テーブルログのローテーションが実行されます。

Amazon RDS では、TABLE ログおよび FILE ログのローテーションが Amazon RDS イベントで記録され、ユーザーに通知が送信されます。

Amazon RDS コンソール、Amazon RDS API、Amazon RDS CLI、または AWS SDK からログを使用するには、`log_output` パラメータを FILE に設定します。MariaDB エラーログと同様、これらのログファイルは 1 時間ごとにローテーションされます。直前 24 時間以内に生成されたログファイルが保持されます。

スロークエリと一般ログの詳細については、MariaDB のドキュメントの以下のトピックを参照してください。

- スロークエリログ
- 一般クエリログ

MariaDB ログを Amazon CloudWatch Logs に発行する

MariaDB DB インスタンスを設定して、ログデータを Amazon CloudWatch Logs のロググループに発行することができます。CloudWatch Logs を使用すると、ログデータのリアルタイム分析や、CloudWatch を使用したアラームの作成、メトリクスの表示を行うことができます。CloudWatch Logs を使用して、耐久性の高いストレージにログレコードを格納できます。

Amazon RDS は、各 MariaDB データベースログを、ロググループの別個のデータストリーミングとして発行します。例えば、エクスポート関数を設定して、スロークエリログを含んでいるとします。次に、スロークエリデータは、`/aws/rds/instance/my_instance/slowquery` ロググループのスロークエリログストリームに保存されます。

エラーログはデフォルトで有効になります。他の MariaDB ログの要件の概要を次の表に示します。

ログ	要件
監査ログ	DB インスタンスは、`MARIADB_AUDIT_PLUGIN` オプションを指定したカスタムオプショングループを使用する必要があります。
全般ログ	DB インスタンスは、パラメータ設定 `general_log = 1` を指定して一般ログを有効にしたカスタムパラメータグループを使用する必要があります。
スロークエリログ	DB インスタンスは、パラメータ設定 `slow_query_log = 1` または `log_slow_`

ログ	要件
	`query = 1` を指定してスロークエリログを有効にしたカスタムパラメータグループを使用する必要があります。
IAM データベース認証エラーログ	DB インスタンスを作成または変更して、DB インスタンスのログタイプ `iam-db-auth-error` を有効にする必要があります。
ログ出力	DB インスタンスは、パラメータ設定 `log_output = FILE` を指定してログをファイルシステムに書き込み、CloudWatch Logs に発行するカスタムパラメータグループを使用する必要があります。

コンソール

コンソールから CloudWatch Logs に MariaDB ログを発行するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

MariaDB ログは、AWS CLI を使用して発行することができます。以下のパラメータを使用して、`modify-db-instance` コマンドを呼び出せます。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下の AWS CLI コマンドを呼び出すことで MariaDB ログを発行することもできます。

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

以下のオプションを使用して、この AWS CLI コマンドの 1 つを実行します。

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`
- `--db-instance-class`
- `--engine`

実行する AWS CLI コマンドに応じて、他のオプションが必要となる場合があります。

Example

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の MariaDB DB インスタンスを変更します。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `EnableLogTypes` であり、値は `audit`、`error`、`general`、および `slowquery` を任意に組み合わせた文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Example

次のコマンドでは、MariaDB DB インスタンスを作成してログファイルを CloudWatch Logs に発行します。`--enable-cloudwatch-logs-exports` 値は、JSON 形式の文字列の配列です。この文字列は `audit`、`error`、`general` および `slowquery` の任意の組み合わせです。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --db-instance-class db.m4.large \
    --engine mariadb
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' ^
    --db-instance-class db.m4.large ^
    --engine mariadb
```

RDS API

MariaDB ログは、RDS API を使用して発行することができます。以下のパラメータを指定して `ModifyDBInstance` オペレーションを呼び出します。

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

> **Note**
>
> `CloudwatchLogsExportConfiguration` パラメータへの変更は常に DB インスタンスに即時適用されます。それで、`ApplyImmediately` パラメータは効果がありません。

以下の RDS API オペレーションを呼び出すことで MariaDB ログを発行することもできます。

- `CreateDBInstance`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceFromS3`
- `RestoreDBInstanceToPointInTime`

以下のパラメータでこの RDS API オペレーションの 1 つを実行します。

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

実行する AWS CLI コマンドに応じて、他のパラメータが必要となる場合があります。

MariaDB のログのローテーションと保持

ログ記録が有効になっている場合、Amazon RDS は、テーブルログのローテーションまたはログファイルの削除を定期的に実行します。これは、ログファイルが大きくなることでデータベースが使用できなくなったりパフォーマンスに影響する可能性を低く抑えるための予防措置です。

MariaDB のスロークエリログ、エラーログ、一般ログファイルのサイズは、DB インスタンスに割り当てられたストレージ領域の 2 パーセント以下に制約されます。このしきい値を維持するために、ログは 1 時間ごとに自動的にローテーションされ、24 時間以上前の古いログファイルは削除されます。古いログファイルを削除した後、ログファイルの合計サイズがしきい値を超えている場合、ログファイルのサイズがしきい値以下になるまで、最も大きいログファイルから順に削除されます。

Amazon RDS は、10 MB を超える IAM データベース認証エラーログファイルをローテーションします。Amazon RDS は、5 日以上経過しているか、100 MB を超える IAM データベース認証エラーログファイルを削除します。

テーブルベースの MariaDB ログを管理する

DB インスタンスのテーブルに一般ログとスロークエリログを移動できます。そのためには、DB パラメータグループを作成し、log_output サーバーパラメータを TABLE に設定します。その後、一般クエリは mysql.general_log テーブルに記録され、スロークエリは mysql.slow_log テーブルに記録されます。それらのテーブルに対してクエリを実行することでログの情報にアクセスできます。このログ記録を有効にすると、データベースに書き込まれるデータの量が増え、パフォーマンスが低下することがあります。

一般ログもスロークエリログもデフォルトで無効になっています。テーブルへのログ記録を有効にするには、以下のサーバーパラメータを 1 に設定する必要もあります。

- general_log
- slow_query_log、または log_slow_query

ログテーブルは、それぞれのログ記録アクティビティのパラメータを 0 にリセットしてログ記録をオフにするまで、拡大し続けます。大量のデータが長期にわたって蓄積されることがよくあり、割り当てストレージ領域の大部分を使い果たすことがあります。Amazon RDS では、ログテーブルを切り詰めることはできませんが、その内容を移動することはできます。テーブルのローテーションにより、その内容がバックアップテーブルに保存され、新しい空のログテーブルが作成されます。以下のコマンドラインプロシージャを使用して、ログテーブルを手動でローテーションされることができます。ここで表示されている PROMPT> はコマンドプロンプトです。

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

以前のデータを完全に削除し、ディスク領域を再利用するには、該当するプロシージャを 2 回連続で呼び出します。

MariaDB バイナリログ記録の設定

バイナリログは、MariaDB サーバーインスタンスで行われたデータ変更に関する情報を含む、一連のログファイルです。バイナリログには、以下のような情報が含まれています。

- テーブルの作成や行の変更など、データベースの変更が記述されたイベント
- データを更新した各ステートメントの実行時間に関する情報
- データを更新する可能性があったものの、それが実行されていないステートメントのイベント

バイナリログには、レプリケーション中に送信されるステートメントが記録されます。また、一部のリカバリオペレーションにもバイナリログが必要です。詳細については、MariaDB のドキュメントの「Binary Log」を参照してください。

自動バックアップ機能では、MariaDB のバイナリログ記録を有効にするか無効にするかを決定します。次のオプションがあります。

バイナリログ記録を有効にするには

バックアップ保持期間を 0 以外の正の値に設定します。

バイナリログ記録を無効にするには

[バックアップ保持期間] を 0 に設定します。

詳細については、「自動バックアップの有効化」を参照してください。

Amazon RDS の MariaDB は行ベース、ステートメントベース、および混合のバイナリログ記録形式をサポートしています。デフォルトのバイナリログ形式は混合です。さまざまな MariaDB のバイナリログ形式の詳細については、MariaDB ドキュメントの「バイナリログ形式」を参照してください。

レプリケーションを使用する予定の場合は、バイナリログ形式が重要です。これは、ソースに記録されてレプリケーションターゲットに送信されるデータ変更記録が決定されるからです。レプリケーション用のさまざまなバイナリログ記録形式の利点と欠点についての詳細は、MySQL ドキュメントの「Advantages and Disadvantages of Statement-Based and Row-Based Replication」を参照してください。

> **Important**
>
> バイナリログ形式を行ベースに設定すると、バイナリログファイルが巨大になることがあります。巨大なバイナリログファイルにより、DB インスタンスの使用可能なストレージの量が減ります。また、DB インスタンスの復元オペレーションの実行にかかる時間が長くなることがあります。
>
> ステートメントベースのレプリケーションは、ソース DB インスタンスとリードレプリカの間の不整合の原因になります。詳細については、MariaDB ドキュメントの「Unsafe Statements for Statement-based Replication」を参照してください。

バイナリログを有効にすると、DB インスタンスへの書き込みディスク I/O 操作の回数が増えます。`WriteIOPS` CloudWatch メトリクスを使用して、IOPS の使用状況をモニタリングできます。

MariaDB バイナリログ形式を設定するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. 変更する DB インスタンスに使用されているパラメータグループを選択します。

   デフォルトのパラメータグループを変更することはできません。DB インスタンスがデフォルトのパラメータグループを使用している場合、新しいパラメータグループを作成し DB インスタンスと関連付けます。

   DB パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。
4. [Parameter group actions (パラメータグループのアクション)] で、[編集] を選択します。
5. `binlog_format` パラメータを、選択したバイナリログ記録形式 (ROW、STATEMENT、または MIXED) に設定します。

   DB インスタンスのバックアップ保持期間をゼロに設定することでバイナリログを無効にできますが、これによって毎日の自動バックアップは無効になります。自動バックアップを無効にすると、`log_bin` セッション変数がオフまたは無効になります。これにより、RDS for MariaDB DB インスタンスのバイナリログ記録が無効になり、`binlog_format` セッション変数がデータベースのデフォルト値の `ROW` にリセットされます。バックアップを無効にしないことをお勧めします。バックアップ保持期間の設定の詳細については、「DB インスタンスの設定」を参照してください。
6. [変更の保存] を選択して、更新を DB パラメータグループに保存します。

`binlog_format` パラメータは RDS for MariaDB では動的であるため、変更を適用するために DB インスタンスを再起動する必要はありません。

> **Important**
>
> DB パラメータグループを変更すると、そのパラメータグループを使用するすべての DB インスタンスに影響を与えます。AWS リージョン内の異なる MariaDB DB インスタンスに対して異なるバイナリログ形式を指定する場合、DB インスタンスは異なる DB パラメータグループを使用する必要があります。これらのパラメータグループは、さまざまなログ形式を識別します。各 DB インスタンスに適切な DB パラメータグループを割り当てます。

MariaDB バイナリログにアクセスする

mysqlbinlog ユーティリティを使用して、バイナリログをテキスト形式で MariaDB DB インスタンスからダウンロードできます。バイナリログは、お使いのコンピュータにダウンロードされます。mysqlbinlog ユーティリティの使用の詳細については、MariaDB ドキュメントの「mysqlbinlog を使用する」を参照してください。

Amazon RDS インスタンスに対して mysqlbinlog ユーティリティを実行するには、以下のオプションを使用します。

- `--read-from-remote-server` オプションを指定します。
- `--host`: インスタンスのエンドポイントからの DNS 名を指定します。
- `--port`: インスタンスによって使用されるポートを指定します。
- `--user`: レプリケーションスレーブアクセス許可を付与された MariaDB ユーザーを指定します。
- `--password`: ユーザーのパスワードを指定するか、パスワード値を省略します。省略した場合、ユーティリティによってパスワードの入力を求められます。
- `--result-file`: 出力を受け取るローカルファイルを指定します。
- 1 つ以上のバイナリログファイルの名前を指定します。使用可能なログのリストを取得するには、SQL コマンド SHOW BINARY LOGS を使用します。

mysqlbinlog オプションの詳細については、MariaDB ドキュメントの「mysqlbinlog オプション」を参照してください。

以下に例を示します。

Linux、macOS、Unix の場合:

```
mysqlbinlog \
```

```
    --read-from-remote-server \
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password <password> \
    --result-file=/tmp/binlog.txt
```

Windows の場合:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password <password> ^
    --result-file=/tmp/binlog.txt
```

Amazon RDS では、通常、バイナリログはできる限り早く消去されます。ただし、バイナリログは、mysqlbinlog によってアクセスされるインスタンスで引き続き使用可能である必要があります。RDS がバイナリログを保持する時間数を指定するには、`mysql.rds_set_configuration` ストアドプロシージャを使用します。ログをダウンロードするのに十分な期間を指定してください。保持期間を設定したら、DB インスタンスのストレージ使用状況をモニタリングして、保持されたバイナリログに必要以上の容量が使用されないようにします。

以下の例では、保持期間を 1 日に設定しています。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

現在の設定を表示するには、`mysql.rds_show_configuration` ストアドプロシージャを使用します。

```
call mysql.rds_show_configuration;
```

MariaDB バイナリログ注釈の有効化

MariaDB DB インスタンスでは、`Annotate_rows` イベントを使用して列イベントを引き起こした SQL クエリのコピーで行イベントに注釈を追加できます。この方法では、RDS for MySQL DB インスタンスの `binlog_rows_query_log_events` パラメータを有効にするのと同様の機能を提供します。

カスタムパラメータグループを作成し binlog_annotate_row_events パラメータを **1** に設定することで、バイナリログの注釈をグローバルに有効にすることができます。SET SESSION binlog_annotate_row_events = 1 を呼び出すことで、セッションレベルで注釈を有効化することもできます。バイナリログがレプリカインスタンスで有効になっている場合は、replicate_annotate_row_events を使用してバイナリログの注釈をレプリカインスタンスにレプリケートします。これらの設定に特別な権限を使用する必要はありません。

次に MariaDB での行ベースの処理の例を示します。行ベースログの使用は、トランザクションの分離レベルをコミット済み読み取りに設定することで起動されます。

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
```

注釈なしのトランザクションのバイナリログエントリは次のようになります。

```
BEGIN
/*!*/;
# at 1163
# at 1209
#150922  7:55:57 server id 1855786460  end_log_pos 1209         Table_map:
 `test`.`square` mapped to number 76
#150922  7:55:57 server id 1855786460  end_log_pos 1247         Write_rows: table id 76
 flags: STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 1247
#150922  7:56:01 server id 1855786460  end_log_pos 1274         Xid = 62
COMMIT/*!*/;
```

次のステートメントでは、同じのトランザクションのセッションレベルの注釈を有効にし、トランザクションをコミットした後に無効にしています。

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
```

```
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET SESSION binlog_annotate_row_events = 1;
BEGIN;
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
SET SESSION binlog_annotate_row_events = 0;
```

注釈ありのトランザクションのバイナリログエントリは次のようになります。

```
BEGIN
/*!*/;
# at 423
# at 483
# at 529
#150922  8:04:24 server id 1855786460  end_log_pos 483  Annotate_rows:
#Q> INSERT INTO square(x, y) VALUES(5, 5 * 5)
#150922  8:04:24 server id 1855786460  end_log_pos 529  Table_map: `test`.`square`
 mapped to number 76
#150922  8:04:24 server id 1855786460  end_log_pos 567  Write_rows: table id 76 flags:
 STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 567
#150922  8:04:26 server id 1855786460  end_log_pos 594  Xid = 88
COMMIT/*!*/;
```

Amazon RDS for Microsoft SQL Server データベースのログファイル

Amazon RDS コンソール、AWS CLI、または RDS API を使用して、Microsoft SQL Server のエラーログ、エージェントログ、トレースファイル、ダンプファイルにアクセスできます。ファイルベースのデータベースログの表示、ダウンロード、モニタリングの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

保持期間スケジュール

ログファイルは、毎日、および DB インスタンスが再開されるたびにローテーションされます。以下は、Amazon RDS の Microsoft SQL Server ログの保持期間スケジュールです。

ログタイプ	保持期間スケジュール
エラーログ	最大 30 のエラーログを保持します。Amazon RDS は 7 日を経過したエラーログを削除する場合があります。
エージェントログ	最大 10 のエージェントログを保持します。Amazon RDS は 7 日を経過したエージェントログを削除する場合があります。
トレースファイル	トレースファイルは、DB インスタンスのトレースファイル保持期間に応じて保持されます。トレースファイルのデフォルトの保持期間は 7 日です。DB インスタンスのトレースファイル保持期間を変更するには、「トレースファイルおよびダンプファイルの保持期間を設定する」を参照してください。
ダンプファイル	ダンプファイルは、DB インスタンスのダンプファイル保持期間に応じて保持されます。ダンプファイルのデフォルトの保持期間は 7 日です。DB インスタンスのダンプファイル保持期間を変更するには、「トレースファイルおよびダンプファイルの保持期間を設定する」を参照してください。

rds_read_error_log プロシージャを使用して SQL Server エラーログを表示する

エラーログおよびエージェントログを表示するには、Amazon RDS ストアドプロシージャ `rds_read_error_log` を使用できます。詳細については、「エラーログとエージェントログの表示」を参照してください。

Amazon CloudWatch Logs への SQL Server ログの発行

Amazon RDS for SQL Server ではエラーおよびエージェントログを直接 Amazon CloudWatch Logs に発行できます。CloudWatch Logs を使用してログデータを分析し、CloudWatch を使用してアラームを作成し、メトリクスを表示することができます。

CloudWatch Logs では、次のことを実行できます。

- ログは、ユーザーが定義する保持期間で耐久性の高いストレージ領域に保存します。
- ログデータを検索しフィルタリングします。
- アカウント間でログデータを共有します。
- ログを Amazon S3 にエクスポートします。
- Amazon OpenSearch Service へのデータのストリーミング
- Amazon Kinesis Data Streams を使用してログデータをリアルタイムで処理します。詳細については、「Amazon Managed Service for Apache Flink for Apache Flink デベロッパーガイド」内の「Amazon CloudWatch Logs の使用」を参照してください。

Amazon RDS は、各 SQL Server データベースログを、ロググループの別個のデータストリーミングとして発行します。例えば、エージェントログとエラーログを発行した場合、エラーデータは `/aws/rds/instance/my_instance.node1/error` ロググループのエラーログストリームに保存され、エージェントログデータは `/aws/rds/instance/my_instance.node1/agent` ロググループに保存されます。

マルチ AZ DB インスタンスの場合、Amazon RDS はデータベースログをロググループ内の 2 つの独立したストリームとして公開します。例えば、エラーログを発行する場合、エラーデータは `/aws/rds/instance/my_instance.node1/error` および `/aws/rds/instance/my_instance.node2/error` のエラーログストリームに保存されます。ログストリームはフェイルオーバー中でも変更されず、各ノードのエラーログストリームには、プライマリインスタンスまたはセカンダリインスタンスのエラーログを含めることができます。マルチ AZ では、ログストリームは自動的に作成され、`/aws/rds/instance/my_instance/rds-events` は DB インスタンスのフェイルオーバーなどのイベントデータを保存します。

> **Note**
>
> SQL Server ログの CloudWatch Logs への発行はデフォルトでは有効にされていません。トレースファイルおよびダンプファイルの発行はサポートされていません。SQL Server ログの CloudWatch Logs への発行は、すべてのリージョンでサポートされています。

コンソール

AWS マネジメントコンソール から CloudWatch Logs に SQL Server DB ログを公開するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。

   エージェントログ、エラーログ、またはその両方を選択できます。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

Oracle ログを発行するには、以下のパラメータを指定して `modify-db-instance` コマンドを使用します。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下のコマンドを使用して SQL Server ログを発行することもできます。

- `create-db-instance`

- restore-db-instance-from-db-snapshot
- restore-db-instance-to-point-in-time

Example

次の例では、CloudWatch Logs の発行を有効にした SQL Server DB インスタンスを作成します。--enable-cloudwatch-logs-exports 値は、error、agent、または両方を含むことができる JSON 文字列です。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["error","agent"]' \
    --db-instance-class db.m4.large \
    --engine sqlserver-se
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports "[\"error\",\"agent\"]" ^
    --db-instance-class db.m4.large ^
    --engine sqlserver-se
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

Example

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の SQL Server DB インスタンスを変更します。--cloudwatch-logs-export-configuration 値は JSON オブジェクトです。このオブジェクトのキーは EnableLogTypes であり、その値は error、agent、または両方を含む文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["error","agent"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"EnableLogTypes\":[\"error\",\"agent\"]}"
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

Example

次の例では、既存の SQL Server DB インスタンスを変更して、ログファイルを CloudWatch Logs に発行できないようにします。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `DisableLogTypes` であり、その値は `error`、`agent`、または両方を含む文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["agent"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"agent\"]}"
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

MySQL データベースのログファイル

MySQL ログは、Amazon RDS コンソール、Amazon RDS API、AWS CLI、または AWS SDK を通じて直接モニタリングできます。また、ログをメインデータベースのデータベーステーブルに書き込み、そのテーブルに対してクエリを実行することで、MySQL ログにアクセスできます。mysqlbinlog ユーティリティを使用して、バイナリログをダウンロードできます。

ファイルベースのデータベースログの表示、ダウンロード、モニタリングの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

トピック

- RDS for MySQL データベースログの概要
- Amazon CloudWatch Logs への MySQL ログの発行
- テーブルへの MySQL ログ出力の送信
- インスタンスデプロイの RDS for MySQL バイナリログの設定
- マルチ AZ DB クラスターの MySQL バイナリログ記録の設定
- MySQL バイナリログにアクセスする

RDS for MySQL データベースログの概要

次の種類の RDS for MySQL ログファイルをモニタリングできます。

- エラーログ
- スロークエリログ
- 全般ログ
- 監査ログ
- インスタンスログ
- IAM データベース認証エラーログ

RDS for MySQL のエラーログはデフォルトで生成されます。DB パラメータグループにパラメータを設定することで、低速クエリと一般ログを生成できます。

トピック

- RDS for MySQL エラーログ
- RDS for MySQL のスロークエリと一般ログ

- MySQL 監査ログ
- RDS for MySQL のログのローテーションと保持
- REDO ログのサイズ制限

RDS for MySQL エラーログ

RDS for MySQL は `mysql-error.log` ファイルにエラーを書き込みます。各ログファイルには、それぞれ生成された時間 (UTC) がファイル名に付加されます。ログファイルには、タイムスタンプも付加され、ログエントリがいつ書き込まれたかを調べるために役立ちます。

RDS for MySQL では起動時、シャットダウン時、およびエラー検出時にのみ、エラーログへの書き込みが行われます。DB インスタンスでは、新しいエントリがエラーログに書き込まれないまま、数時間または数日が経過することがあります。最近のエントリがない場合、それは、サーバーにログエントリになり得るエラーが発生しなかったためです。

設計上、エラーログはフィルタリングされ、エラーなどの予期しないイベントのみが表示されます。ただし、エラーログには、クエリの進行状況など、表示されない追加のデータベース情報も含まれています。したがって、実際のエラーがなくても、継続的なデータベースアクティビティのためにエラーログのサイズが増加する可能性があります。また、AWS マネジメントコンソール のエラーログには特定のサイズがバイト単位またはキロバイト単位で表示されている場合がありますが、ダウンロードすると 0 バイトになる場合があります。

RDS for MySQL は 5 分ごとに `mysql-error.log` をディスクに書き込みます。ログの内容が `mysql-error-running.log` に追加されます。

RDS for MySQL は `mysql-error-running.log` ファイルを 1 時間ごとにローテーションします。過去 2 週間に生成されたログが保持されます。

> **Note**
>
> ログの保持期間は、Amazon RDS と Aurora で異なります。

RDS for MySQL のスロークエリと一般ログ

RDS for MySQL のスロークエリログと一般ログを、ファイルまたはデータベーステーブルに書き込めます。このためには、DB パラメータグループにパラメータを設定します。DB パラメータグループの作成と変更の詳細については、「Amazon RDS のパラメータグループ」を参照してください。Amazon RDS コンソール、Amazon RDS API、Amazon RDS CLI、または AWS SDK を使用し

て、スロークエリログまたは一般ログを表示する前に、以下のパラメータを設定する必要があります。

以下のリストに示すパラメータを使用して RDS for MySQL のログ記録を制御できます。

- slow_query_log: スロークエリログを作成するには、1 に設定します。デフォルトは 0 です。
- general_log: 一般ログを作成するには、1 に設定します。デフォルトは 0 です。
- long_query_time: ファストクエリがスロークエリログに記録されないようにするために、ログに記録されるクエリの最短実行時間の値を秒単位で指定します。デフォルトは 10 秒で、最小値は 0 です。log_output = FILE の場合は、マイクロ秒の精度になるように、浮動小数点値を指定できます。log_output = TABLE の場合は、秒の精度になるように、整数値を指定する必要があります。実行時間が long_query_time の値を超えたクエリのみがログに記録されます。例えば、long_query_time を 0.1 に設定すると、実行時間が 100 ミリ秒未満のすべてのクエリはログに記録されなくなります。
- log_queries_not_using_indexes: インデックスを使用しないすべてのクエリをスロークエリログに記録するには、1 に設定します。インデックスを使用しないクエリは、その実行時間が long_query_time パラメータの値未満であってもログに記録されます。デフォルトは 0 です。
- log_output *option*: log_output パラメータに指定できるオプションは、次のとおりです。
  - TABLE (デフォルト) - 一般クエリを mysql.general_log テーブルに、スロークエリを mysql.slow_log テーブルに書き込みます。
  - FILE - 一般クエリログとスロークエリログの両方をファイルシステムに書き込みます。
  - NONE - ログ記録を無効にします。

スロークエリデータを Amazon CloudWatch Logs に表示するには、次の条件を満たす必要があります。

- スロークエリログを含むように CloudWatch Logs を設定する必要があります。
- slow_query_log を有効にする必要があります。
- log_output/ を に設定する必要があります。FILE
- クエリに要する時間が、long_query_time に設定された時間よりも長い必要があります。

スロークエリと一般ログの詳細については、MySQL ドキュメントの以下のトピックを参照してください。

- スロークエリログ

- 一般クエリログ

MySQL 監査ログ

監査ログにアクセスするには、DB インスタンスは MARIADB_AUDIT_PLUGIN オプションを指定してカスタムオプショングループを使用する必要があります。詳細については、「MySQL に対する MariaDB 監査プラグインのサポート」を参照してください。

RDS for MySQL のログのローテーションと保持

ログ記録が有効になっている場合、Amazon RDS は、テーブルログのローテーションまたはログファイルの削除を定期的に実行します。これは、ログファイルが大きくなることでデータベースが使用できなくなったりパフォーマンスに影響する可能性を低く抑えるための予防措置です。RDS for MySQL は、次のようにローテーションと削除を処理します。

- MySQL のスロークエリログ、エラーログ、一般ログファイルのサイズは、DB インスタンスに割り当てられたストレージ領域の 2 パーセント以下に制約されます。このしきい値を維持するために、ログは 1 時間ごとに自動的にローテーションされます。MySQL は、使用が 2 週間を超えたログファイルを削除します。古いログファイルを削除した後、ログファイルの合計サイズがしきい値を超えている場合、ログファイルのサイズがしきい値以下になるまで、最も古いログファイルから順に削除されます。
- FILE ログ記録が有効になっている場合、ログファイルの検査が 1 時間ごとに実行され、作成後 2 週間を超えたログファイルは削除されます。場合によっては、削除後の残りのログファイルの合計サイズが、DB インスタンスに割り当てられた領域のしきい値である 2 % を超えることがあります。この場合、ログファイルのサイズがしきい値以下になるまで、最も古いログファイルから順に削除されます。
- TABLE ロギングを有効化すると、24 時間ごとにログテーブルのローテーションが実行される場合があります。このログテーブルのローテーションは、テーブルログに使用されている領域が、割り当てられたストレージ領域の 20 % を超えると、実行されます。結合されたすべてのログのサイズが 10 GB を超える場合にも発生します。DB インスタンスに使用されている領域が、DB インスタンスに割り当てられたストレージ領域の 90% を超えている場合は、ログのローテーションを実行するためのしきい値が小さくなります。テーブルログに使用されている領域が、割り当てられたストレージ領域の 10% を超えると、ログテーブルのローテーションが実行されます。結合されたすべてのログのサイズが 5 GB を超えると、ログはローテーションされます。low storage イベントカテゴリにサブスクライブして、ログテーブルのローテーションが実行されて領域が解放されたときに通知を受け取ることができます。詳細については、「Amazon RDS イベント通知の操作」を参照してください。

ログテーブルをローテーションすると、現在のログテーブルがまずバックアップのログテーブルにコピーされます。その後、現在のログテーブルのエントリが削除されます。バックアップのログテーブルが既に存在する場合は、現在のログテーブルをバックアップにコピーする前に、削除されます。バックアップのログテーブルは、必要に応じて照会することができます。mysql.general_log テーブルに対するバックアップのログテーブルは、mysql.general_log_backup という名前になります。mysql.slow_log テーブルに対するバックアップのログテーブルは、mysql.slow_log_backup という名前になります。

mysql.general_log テーブルのローテーションは、mysql.rds_rotate_general_log プロシージャを呼び出すことで実行できます。mysql.slow_log テーブルのローテーションは、mysql.rds_rotate_slow_log プロシージャを呼び出すことで実行できます。

データベースバージョンのアップグレード時にも、テーブルログのローテーションが実行されます。

Amazon RDS コンソール、Amazon RDS API、Amazon RDS CLI、または AWS SDK からログを使用するには、log_output パラメータを FILE に設定します。MySQL エラーログと同様、これらのログファイルは 1 時間ごとにローテーションされます。直近の 2 週間に生成されたログファイルが保持されます。Amazon RDS と Aurora で保持期間が異なる点に注意してください。

REDO ログのサイズ制限

RDS for MySQL バージョン 8.0.32 以前の場合、このパラメータのデフォルト値は 256 MB です。この量は、innodb_log_file_size パラメータ (128 MB) のデフォルト値に innodb_log_files_in_group パラメータ (2) のデフォルト値を掛けることによって算出されます。詳細については、「Best practices for configuring parameters for Amazon RDS for MySQL, part 1: Parameters related to performance」を参照してください。

RDS for MySQL バージョン 8.0.33 以降のマイナーバージョンの場合、Amazon RDS では innodb_log_file_size パラメータの代わりに innodb_redo_log_capacity パラメータが使用されます。innodb_redo_log_capacity パラメータの Amazon RDS デフォルト値は 2 GB です。詳細については、MySQL ドキュメントの「MySQL 8.0.30 での変更点」を参照してください。

MySQL 8.4 以降、Amazon RDS ではデフォルトで innodb_dedicated_server パラメータが有効になります。innodb_dedicated_server パラメータを使用すると、データベースエンジンは innodb_buffer_pool_size および innodb_redo_log_capacity パラメータを計算します。詳細については、「MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定」を参照してください。

Amazon CloudWatch Logs への MySQL ログの発行

MySQL DB インスタンスを設定して、ログデータを Amazon CloudWatch Logs のロググループに発行することができます。CloudWatch Logs を使用すると、ログデータのリアルタイム分析や、CloudWatch を使用したアラームの作成、メトリクスの表示を行うことができます。CloudWatch Logs を使用して、耐久性の高いストレージにログレコードを格納できます。

Amazon RDS は、MySQL データベースログを、ロググループの別のデータストリーミングとしてそれぞれ発行します。例えば、エクスポート機能を設定して、スロークエリログを作成すると、スロークエリデータは、`/aws/rds/instance/my_instance/slowquery` ロググループのスロークエリログストリーミングに保存されます。

エラーログはデフォルトで有効になります。他の MySQL ログの要件の概要を次の表に示します。

ログ	要件
監査ログ	DB インスタンスは、`MARIADB_AUDIT_PLUGIN` オプションを指定したカスタムオプショングループを使用する必要があります。
全般ログ	DB インスタンスは、パラメータ設定 `general_log = 1` を指定して一般ログを有効にしたカスタムパラメータグループを使用する必要があります。
スロークエリログ	DB インスタンスは、パラメータ設定 `slow_query_log = 1` を指定してスロークエリログを有効にしたカスタムパラメータグループを使用する必要があります。
IAM データベース認証エラーログ	DB インスタンスを作成または変更して、DB インスタンスのログタイプ `iam-db-auth-error` を有効にする必要があります。
ログ出力	DB インスタンスは、パラメータ設定 `log_output = FILE` を指定してログをファイルシステムに書き込み、CloudWatch Logs に発行するカスタムパラメータグループを使用する必要があります。

コンソール

コンソールを使用して CloudWatch Logs に MySQL ログを発行するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

AWS CLI を使用して MySQL ログを発行することができます。以下のパラメータを使用して、`modify-db-instance` コマンドを呼び出せます。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下の AWS CLI コマンドを呼び出すことで MySQL ログを発行することもできます。

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-from-s3`
- `restore-db-instance-to-point-in-time`

以下のオプションを使用して、この AWS CLI コマンドの 1 つを実行します。

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`

- `--db-instance-class`
- `--engine`

実行する AWS CLI コマンドに応じて、他のオプションが必要となる場合があります。

Example

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の MySQL DB インスタンスを変更します。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `EnableLogTypes` であり、値は `audit`、`error`、`general`、および `slowquery` を任意に組み合わせた文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["audit","error","general","slowquery"]}'
```

Example

次の例では、MySQL DB インスタンスを作成してログファイルを CloudWatch Logs に発行します。`--enable-cloudwatch-logs-exports` 値は、JSON 形式の文字列の配列です。この文字列は `audit`、`error`、`general` および `slowquery` の任意の組み合わせです。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --db-instance-class db.m4.large \
    --engine MySQL
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' ^
    --db-instance-class db.m4.large ^
    --engine MySQL
```

RDS API

RDS API を使用して MySQL ログを発行することができます。以下のパラメータを使用して、ModifyDBInstance アクションを呼び出せます。

- DBInstanceIdentifier
- CloudwatchLogsExportConfiguration

> **Note**
>
> CloudwatchLogsExportConfiguration パラメータへの変更は常に DB インスタンスに即時適用されます。それで、ApplyImmediately パラメータは効果がありません。

以下の RDS API オペレーションを呼び出すことで MySQL ログを発行することもできます。

- CreateDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

以下のパラメータでこの RDS API オペレーションの 1 つを実行します。

- DBInstanceIdentifier
- EnableCloudwatchLogsExports
- Engine

- DBInstanceClass

実行する AWS CLI コマンドに応じて、他のパラメータが必要となる場合があります。

テーブルへの MySQL ログ出力の送信

DB パラメータグループを作成し、log_output サーバーパラメータを TABLE に設定することで、DB インスタンス上のテーブルに一般ログとスロークエリログを書き込むことができます。その後、一般クエリは mysql.general_log テーブルに記録され、スロークエリは mysql.slow_log テーブルに記録されます。それらのテーブルに対してクエリを実行することでログの情報にアクセスできます。このログ記録を有効にすると、データベースに書き込まれるデータの量が増え、パフォーマンスが低下することがあります。

一般ログもスロークエリログもデフォルトで無効になっています。テーブルへのログ記録を有効にするには、general_log と slow_query_log のサーバーパラメータを 1 に設定する必要があります。

ログテーブルは、それぞれのログ記録アクティビティのパラメータを 0 にリセットしてログ記録をオフにするまで、拡大し続けます。大量のデータが長期にわたって蓄積されることがよくあり、割り当てストレージ領域の大部分を使い果たすことがあります。Amazon RDS では、ログテーブルを切り詰めることはできませんが、その内容を移動することはできます。テーブルのローテーションにより、その内容がバックアップテーブルに保存され、新しい空のログテーブルが作成されます。以下のコマンドラインプロシージャを使用して、ログテーブルを手動でローテーションされることができます。ここで表示されている PROMPT> はコマンドプロンプトです。

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

以前のデータを完全に削除し、ディスク領域を再利用するには、該当するプロシージャを 2 回連続で呼び出します。

インスタンスデプロイの RDS for MySQL バイナリログの設定

バイナリログは、RDS for MySQL DB インスタンスで行われたデータ変更に関する情報を含む、一連のログファイルです。バイナリログには、以下のような情報が含まれています。

- テーブルの作成や行の変更など、データベースの変更が記述されたイベント
- データを更新した各ステートメントの実行時間に関する情報
- データを更新する可能性があったものの、それが実行されていないステートメントのイベント

バイナリログには、レプリケーション中に送信されるステートメントが記録されます。また、一部のリカバリオペレーションにもバイナリログが必要です。詳細については、MySQL ドキュメントの「バイナリログ」を参照してください。

自動バックアップ機能では、MySQL のバイナリログ記録を有効にするか無効にするかを決定します。次のオプションがあります。

バイナリログ記録を有効にするには

バックアップ保持期間を 0 以外の正の値に設定します。

バイナリログ記録を無効にするには

[バックアップ保持期間] を 0 に設定します。

詳細については、「自動バックアップの有効化」を参照してください。

Amazon RDS の MySQL では、行ベース、ステートメントベース、および混合のバイナリログ形式がサポートされています。特定バイナリログ形式が必要でない場合は、混合形式を使用することをお勧めします。MySQL の各種バイナリログ形式の詳細については、MySQL ドキュメントの「Binary logging formats」を参照してください。

レプリケーションを使用する予定の場合は、バイナリログ記録形式が重要です。ソースに記録されてレプリケーションターゲットに送信されるデータ変更記録が決定されるからです。レプリケーション用のさまざまなバイナリログ記録形式の利点と欠点についての詳細は、MySQL ドキュメントの「Advantages and Disadvantages of Statement-Based and Row-Based Replication」を参照してください。

> **Important**
>
> MySQL 8.0.34 では、`binlog_format` パラメータが廃止されました。以降の MySQL バージョンでは、このパラメータを削除し、行ベースのレプリケーションのみをサポートする予定です。そのため、新しい MySQL レプリケーション設定には行ベースのログ記録を使用することをお勧めします。詳細については、MySQL ドキュメントの「binlog_format」を参照してください。
> MySQL バージョン 8.0 および 8.4 は、パラメータ `binlog_format` を受け入れます。このパラメータを使用すると、MySQL は非推奨警告を発行します。今後のメジャーリリースで、MySQL はパラメータ `binlog_format` を削除します。
> ステートメントベースのレプリケーションは、ソース DB インスタンスとリードレプリカの間の不整合の原因になります。詳細については、MySQL ドキュメントの「バイナリログ作

成における安全なステートメントと安全でないステートメントの判断」を参照してください。
バイナリログを有効にすると、DB インスタンスへの書き込みディスク I/O 操作の回数が増えます。`WriteIOPS` CloudWatch メトリクスを使用して、IOPS の使用状況をモニタリングできます。

MySQL バイナリログ形式を設定するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Parameter groups] (パラメータグループ) を選択します。
3. 変更する DB インスタンスに関連付ける DB のパラメータグループを選択します。

   デフォルトのパラメータグループを変更することはできません。DB インスタンスがデフォルトのパラメータグループを使用している場合、新しいパラメータグループを作成し DB インスタンスと関連付けます。

   パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。
4. [アクション] から [編集] を選択します。
5. `binlog_format` パラメータを、選択したバイナリログ形式 (`ROW`、`STATEMENT`、または `MIXED`) に設定します。

   DB インスタンスのバックアップ保持期間をゼロに設定することでバイナリログを無効にできますが、これによって毎日の自動バックアップは無効になります。自動バックアップを無効にすると、`log_bin` セッション変数がオフまたは無効になります。これにより、RDS for MySQL DB インスタンスのバイナリログ記録が無効になり、`binlog_format` セッション変数がデータベースのデフォルト値の `ROW` にリセットされます。バックアップを無効にしないことをお勧めします。バックアップ保持期間の設定の詳細については、「DB インスタンスの設定」を参照してください。
6. [変更の保存] を選択して、更新を DB パラメータグループに保存します。

`binlog_format` パラメータは RDS for MySQL では動的であるため、変更を適用するために、DB インスタンスを再起動する必要はありません。(Aurora MySQL では、このパラメータは静的であることに注意してください。詳細については、「Aurora MySQL バイナリログの設定」を参照してください。)

> **Important**
>
> DB パラメータグループを変更すると、そのパラメータグループを使用するすべての DB インスタンスに影響を与えます。AWS リージョン内の異なる MySQL DB インスタンスに対して異なるバイナリログ形式を指定する場合、DB インスタンスは異なる DB パラメータグループを使用する必要があります。これらのパラメータグループは、さまざまなログ形式を識別します。各 DB インスタンスに適切な DB パラメータグループを割り当てます。

マルチ AZ DB クラスターの MySQL バイナリログ記録の設定

Amazon RDS for MySQL マルチ AZ DB クラスターのバイナリログ記録は、レプリケーション、ポイントインタイムリカバリ、監査をサポートするために、すべてのデータベースの変更を記録します。マルチ AZ DB クラスターのバイナリログ記録では、セカンダリノードをプライマリノードと同期し、アベイラビリティーゾーン間のデータ整合性を確保して、シームレスなフェイルオーバーを可能にします。

バイナリログ記録を最適化するために、Amazon RDS ではバイナリログトランザクション圧縮をサポートしています。これにより、バイナリログのストレージ要件が軽減され、レプリケーション効率が向上します。

トピック

- マルチ AZ DB クラスターのバイナリログトランザクション圧縮
- マルチ AZ DB クラスターのバイナリログトランザクション圧縮の設定

マルチ AZ DB クラスターのバイナリログトランザクション圧縮

バイナリログトランザクション圧縮では、zstd アルゴリズムを使用して、バイナリログに保存されるトランザクションデータのサイズを縮小します。有効にすると、MySQL データベースエンジンがトランザクションペイロードを 1 つのイベントに圧縮し、I/O とストレージのオーバーヘッドを最小限に抑えます。この機能によって、データベースのパフォーマンスが向上し、バイナリログのサイズが縮小して、マルチ AZ DB クラスターのログを管理およびレプリケートするためのリソースの使用が最適化されます。

Amazon RDS は、以下のパラメータを使用して RDS for MySQL マルチ AZ DB クラスターのバイナリログトランザクション圧縮を提供します。

- binlog_transaction_compression – 有効にすると (1)、データベースエンジンはトランザクションペイロードを圧縮し、単一のイベントとしてバイナリログに書き込みます。これにより、ストレージの使用量と I/O オーバーヘッドが減少します。このパラメータは、デフォルトでは無効になっています。
- binlog_transaction_compression_level_zstd – バイナリログトランザクションの zstd 圧縮レベルを設定します。値を大きくすると圧縮率が上がり、ストレージ要件はさらに軽減されますが、圧縮のための CPU とメモリの使用量が増加します。デフォルト値は 3 で、範囲は 1 ~ 22 です。

これらのパラメータを使用して、ワークロードの特性とリソースの可用性に基づいてバイナリログ圧縮を微調整できます。詳細については、MySQL ドキュメントの「Binary Log Transaction Compression」を参照してください。

バイナリログトランザクション圧縮には、主に次の利点があります。

- 特にトランザクションが大きいワークロードや書き込みボリュームが多いワークロードでは、圧縮によってバイナリログのサイズが小さくなります。
- バイナリログが小さいほど、ネットワークと I/O のオーバーヘッドが減少し、レプリケーションのパフォーマンスが向上します。
- binlog_transaction_compression_level_zstd パラメータで、圧縮率とリソース消費のバランスを制御できます。

マルチ AZ DB クラスターのバイナリログトランザクション圧縮の設定

RDS for MySQL マルチ AZ DB クラスターのバイナリログトランザクション圧縮を設定するには、ワークロードの要件に合わせて関連するクラスターパラメータ設定を変更します。

コンソール

バイナリログトランザクション圧縮を有効にするには

1. DB クラスターパラメータグループを変更して、binlog_transaction_compression パラメータを 1 に設定します。
2. (オプション) ワークロードの要件とリソースの可用性に基づいて、binlog_transaction_compression_level_zstd パラメータの値を調整します。

詳細については、「the section called “DB クラスターパラメータグループのパラメータの変更”」を参照してください。

AWS CLI

AWS CLI を使用してバイナリログトランザクション圧縮を設定するには、modify-db-cluster-parameter-group コマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name your-cluster-parameter-group \
  --parameters
 "ParameterName=binlog_transaction_compression,ParameterValue=1,ApplyMethod=pending-
reboot"
```

Windows の場合:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name your-cluster-parameter-group ^
  --parameters
 "ParameterName=binlog_transaction_compression,ParameterValue=1,ApplyMethod=pending-
reboot"
```

RDS API

Amazon RDS API を使用してバイナリログトランザクション圧縮を設定するには、ModifyDBClusterParameterGroup オペレーションを使用します。

MySQL バイナリログにアクセスする

mysqlbinlog ユーティリティを使用して、RDS for MySQL DB インスタンスからバイナリログをダウンロードまたはストリーミングできます。バイナリログはローカルコンピュータにダウンロードされ、mysql ユーティリティを使用してログの再生などの操作を実行できます。mysqlbinlog ユーティリティの使用の詳細については、MySQL ドキュメントの「バイナリログファイルのバックアップのための mysqlbinlog の使用」を参照してください。

Amazon RDS インスタンスに対して mysqlbinlog ユーティリティを実行するには、以下のオプションを使用します。

- `--read-from-remote-server` – 必須。
- `--host` - インスタンスのエンドポイントからの DNS 名。
- `--port` - インスタンスによって使用されるポート。
- `--user` – `REPLICATION SLAVE` アクセス許可を付与された MySQL ユーザー。
- `--password` - MySQL ユーザーのパスワード。パスワード値を省略省略した場合、ユーティリティによってパスワードの入力を求められます。
- `--raw` — バイナリ形式のファイルをダウンロードします。
- `--result-file` - raw 出力を受け取るローカルファイル。
- `--stop-never` — バイナリログファイルをストリーミングします。
- `--verbose` — `ROW` binlog 形式を使用するとき、このオプションを含めると、行イベントが疑似 SQL ステートメントとして表示されます。`--verbose` オプションの詳細については、MySQL ドキュメントの「mysqlbinlog row event display」(mysqlbinlog の行イベントの表示) を参照してください。
- 1 つ以上のバイナリログファイルの名前を指定します。使用可能なログのリストを取得するには、SQL コマンド `SHOW BINARY LOGS` を使用します。

mysqlbinlog のオプションの詳細については、MySQL ドキュメントの「mysqlbinlog - バイナリログファイルを処理するためのユーティリティ」を参照してください。

以下の例では、mysqlbinlog ユーティリティの使用方法を示します。

Linux、macOS、Unix の場合:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Windows の場合:

```
mysqlbinlog ^
```

```
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

mysqlbinlog ユーティリティがバイナリログにアクセスするには、バイナリログが DB インスタンスで利用可能な状態である必要があります。可用性を確保するには、mysql.rds_set_configuration ストアドプロシージャを使用して、ログをダウンロードできる十分な期間を指定します。この設定を行わないと、Amazon RDS はバイナリログをできるだけ早く消去するため、mysqlbinlog ユーティリティが取得するバイナリログにギャップが生じます。

以下の例では、保持期間を 1 日に設定しています。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

現在の設定を表示するには、mysql.rds_show_configuration ストアドプロシージャを使用します。

```
call mysql.rds_show_configuration;
```

Amazon RDS for Oracle データベースのログファイル

Amazon RDS コンソールまたは API を使用して、Oracle のアラートログ、監査ファイル、トレースファイルにアクセスできます。ファイルベースのデータベースログの表示、ダウンロード、モニタリングの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

提供される Oracle の監査ファイルは、スタンダードの Oracle 監査ファイルです。Amazon RDS は、Oracle のきめ細かな監査 (FGA) 機能をサポートしています。ただし、ログアクセスは、SYS.FGA_LOG$ テーブルに保存された FGA イベントと DBA_FGA_AUDIT_TRAIL ビューからアクセス可能な FGA イベントへのアクセスを提供しません。

DB インスタンスの使用可能な Oracle ログファイルを一覧表示する DescribeDBLogFiles API オペレーションでは、MaxRecords パラメータが無視され、最大 1,000 件のレコードが返されます。この呼び出しは、ミリ秒単位の POSIX 日付として LastWritten を返します。

トピック

- 保持期間スケジュール
- Oracle トレースファイルを使用する
- Amazon CloudWatch Logs への Oracle ログの発行
- アラートログとリスナーログへのアクセス

保持期間スケジュール

Oracle データベースエンジンは、ログファイルが非常に大きくなるとローテーションする場合があります。監査ファイルまたはトレースファイルを保持するには、まずダウンロードします。ファイルをローカルに保存すると、Amazon RDS ストレージコストが削減され、データ用の領域が増えます。

次の表は、Amazon RDS における Oracle のアラートログ、監査ファイル、トレースファイルの保持スケジュールを示しています。

ログタイプ	保持期間スケジュール
アラートログ	テキストアラートログは、Amazon RDS による管理の 30 日の保持で毎日ローテーションされます。XML アラートのログは 7 日間以上保存されます。ALERTLOG ビューを使用してこのログにアクセスできます。

ログタイプ	保持期間スケジュール
監査ファイル	監査ファイルのデフォルトの保持期間は 7 日です。Amazon RDS は 7 日を経過した監査ファイルを削除する場合があります。
トレースファイル	トレースファイルのデフォルトの保持期間は 7 日です。Amazon RDS は 7 日を経過したトレースファイルを削除する場合があります。
リスナーログ	リスナーログのデフォルトの保持期間は 7 日です。Amazon RDS は 7 日を経過したリスナーログを削除する場合があります。

> **Note**
>
> 監査ファイルとトレースファイルは同じ保持設定を共有します。

Oracle トレースファイルを使用する

トレース ファイルを作成、更新、アクセス、削除する Amazon RDS の手順の説明を以下に示します。

トピック

- ファイルのリスト化
- トレースファイルとトレースセッションを生成する
- トレースファイルを取得する
- トレースファイルを消去する

ファイルのリスト化

2 つのうちいずれかの手順を使用して、background_dump_dest パスのあらゆるファイルへのアクセスを許可できます。初期の手順では、background_dump_dest 内の最新のファイルのリストをすべて含むビューを更新します。

```
EXEC rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

ビューが更新されたら、以下のビューを照会して結果にアクセスします。

```
SELECT * FROM rdsadmin.tracefile_listing;
```

2 つ目の方法では、FROM table を使用して、非リレーショナルデータを表のような形式でストリーミングし、データベースディレクトリの内容を一覧表示します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('BDUMP'));
```

以下のクエリでは、ログファイルのテキストを表示しています。

```
SELECT text FROM
 TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP','alert_dbname.log.date'));
```

リードレプリカで、V$DATABASE.DB_UNIQUE_NAME をクエリして BDUMP ディレクトリの名前を取得します。一意の名前が DATABASE_B の場合、BDUMP ディレクトリは BDUMP_B です。以下の例では、レプリカの BDUMP 名をクエリし、この名前を使用して alert_DATABASE.log.2020-06-23 の内容をクエリします。

```
SELECT 'BDUMP' || (SELECT regexp_replace(DB_UNIQUE_NAME,'.*(_[A-Z])', '\1') FROM V
$DATABASE) AS BDUMP_VARIABLE FROM DUAL;

BDUMP_VARIABLE
--------------
BDUMP_B

SELECT TEXT FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP_B','alert_DATABASE.log.2020-06-23'));
```

トレースファイルとトレースセッションを生成する

ALTER SESSION には制限がないことから、Oracle でトレースファイルを生成するための多くのスタンダード的な方法は、Amazon RDS DB インスタンスでも使用できます。以下に、より高度なアクセス許可の必要なトレースファイル用の手順を示します。

Oracle での方法	Amazon RDS方法
`oradebug hanganalyze 3`	`EXEC rdsadmin.manage_tracefiles.hanganalyze;`

Oracle での方法	Amazon RDS方法
`oradebug dump systemstate 266`	`EXEC rdsadmin.manage_tracefiles.dump_systemstate;`

Amazon RDS の Oracle DB インスタンスに接続する個々のセッションをトレースするには、複数の方法を使用できます。セッションのトレースを有効にするには、DBMS_SESSION や DBMS_MONITOR など、Oracle が提供する PL/SQL パッケージのサブプログラムを実行できます。詳細については、Oracle ドキュメントの「Enabling Tracing for a Session」を参照してください。

トレースファイルを取得する

Amazon RDS で管理される外部テーブルのスタンダード的な SQL クエリを使用して、background_dump_dest 内の任意のトレースファイルを取得できます。この方法を使用するには、このテーブルの場所として特定のトレースファイルを設定するプロシージャを実行する必要があります。

例えば、前述の rdsadmin.tracefile_listing ビューを使用して、システムのすべてのトレースファイルを一覧表示できます。その後、以下のプロシージャを使用して、目的のトレースファイルを参照するように tracefile_table ビューを設定できます。

```
EXEC
 rdsadmin.manage_tracefiles.set_tracefile_table_location('CUST01_ora_3260_SYSTEMSTATE.trc');
```

以下の例では、外部テーブルを最新のスキーマで作成し、このテーブルの場所として特定のファイルを設定しています。内容は、SQL クエリを使用してローカルファイルに取得できます。

```
SPOOL /tmp/tracefile.txt
SELECT * FROM tracefile_table;
SPOOL OFF;
```

トレースファイルを消去する

トレースファイルが蓄積されて、ディスク領域を消費することがあります。Amazon RDS では、7 日を経過すると、トレースファイルはデフォルトで消去され、ログファイルは消去されます。show_configuration プロシージャを使用してトレースファイルの保持期間を表示および設定できます。設定の結果を表示できるように、コマンド SET SERVEROUTPUT ON を実行する必要があります。

以下の例では、現在のトレースファイルの保持期間を表示し、新しいトレースファイルの保持期間を設定しています。

```
# Show the current tracefile retention
SQL> EXEC rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:10080
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.

# Set the tracefile retention to 24 hours:
SQL> EXEC rdsadmin.rdsadmin_util.set_configuration('tracefile retention',1440);
SQL> commit;

#show the new tracefile retention
SQL> EXEC rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:1440
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.
```

定期的な消去プロセスに加えて、background_dump_dest から手動でファイルを削除することもできます。以下の例では、5 分を経過したすべてのファイルを消去しています。

```
EXEC rdsadmin.manage_tracefiles.purge_tracefiles(5);
```

特定のパターンと一致するすべてのファイルを消去することもできます (そうする場合は .trc などのファイル拡張子を含めないでください)。次の例は、SCHPOC1_ora_5935 で始まるすべてのファイルを消去する方法を示しています。

```
EXEC rdsadmin.manage_tracefiles.purge_tracefiles('SCHPOC1_ora_5935');
```

Amazon CloudWatch Logs への Oracle ログの発行

RDS for Oracle DB インスタンスを設定して、ログデータを Amazon CloudWatch Logs のロググループに発行することができます。CloudWatch Logs を使用すると、ログデータの分析や、CloudWatch を使用したアラームの作成、メトリクスの表示を行うことができます。CloudWatch Logs を使用して、耐久性の高いストレージにログレコードを格納できます。

Amazon RDS は、Oracle データベースログを、ロググループの別のデータストリーミングとしてそれぞれ発行します。例えば、エクスポート機能を設定して、監査ログを作成すると、監査データ

は、/aws/rds/instance/my_instance/audit ロググループの監査ログストリーミングに保存されます。次の表は、RDS for Oracle が Amazon CloudWatch Logs にログを発行するための要件をまとめたものです。

ログ名	要件	デフォルト
アラートログ	なし。このログを無効にすることはできません。	有効
トレースログ	trace_enabled パラメータを TRUE に設定するか、デフォルトのままにします。	TRUE
監査ログ	audit_trail パラメータを次のいずれかの許可された値に設定します。 `{ none | os | db [, extended] | xml [, extended] }`	none
リスナーログ	なし。このログを無効にすることはできません。	有効
Oracle Management Agent ログ	なし。このログを無効にすることはできません。	有効

この Oracle Management Agent ログは、次の表に示すロググループで構成されています。

ログ名	CloudWatch ロググループ
emctl.log	oemagent-emctl
emdctlj.log	oemagent-emdctlj
gcagent.log	oemagent-gcagent
gcagent_errors.log	oemagent-gcagent-errors
emagent.nohup	oemagent-emagent-nohup

ログ名	CloudWatch ロググループ
secure.log	oemagent-secure

詳細については、Oracle ドキュメントの「Management Agent ログとトレースファイルの検索」を参照してください。

コンソール

AWS マネジメントコンソール から CloudWatch Logs に Oracle DB ログを公開するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

Oracle ログを発行するには、以下のパラメータを指定して `modify-db-instance` コマンドを使用します。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下のコマンドを使用して Oracle ログを発行することもできます。

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`

- restore-db-instance-from-s3
- restore-db-instance-to-point-in-time

Example

次の例では、CloudWatch Logs の発行を有効にした Oracle DB インスタンスを作成します。--cloudwatch-logs-export-configuration 値は、JSON 形式の文字列の配列です。この文字列は alert、audit、listener および trace の任意の組み合わせです。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration
 '["trace","audit","alert","listener","oemagent"]' \
    --db-instance-class db.m5.large \
    --allocated-storage 20 \
    --engine oracle-ee \
    --engine-version 19.0.0.0.ru-2024-04.rur-2024-04.r1 \
    --license-model bring-your-own-license \
    --master-username myadmin \
    --manage-master-user-password
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration trace alert audit listener oemagent ^
    --db-instance-class db.m5.large ^
    --allocated-storage 20 ^
    --engine oracle-ee ^
    --engine-version 19.0.0.0.ru-2024-04.rur-2024-04.r1 ^
    --license-model bring-your-own-license ^
    --master-username myadmin ^
    --manage-master-user-password
```

Example

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の Oracle DB インスタンスを変更します。--cloudwatch-logs-export-configuration 値は JSON オブジェクトです。

このオブジェクトのキーは EnableLogTypes であり、値は alert、audit、listener、および trace を任意に組み合わせた文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["trace","alert","audit","listener","oemagent"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration EnableLogTypes=\"trace\",\"alert\",\"audit
\",\"listener\",\"oemagent\"
```

Example

次の例では、CloudWatch Logs への監査およびリスナーログファイルの発行が無効になるよう既存の Oracle DB インスタンスを変更します。--cloudwatch-logs-export-configuration 値は JSON オブジェクトです。このオブジェクトのキーは DisableLogTypes であり、値は alert、audit、listener、および trace を任意に組み合わせた文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit","listener"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration DisableLogTypes=\"audit\",\"listener\"
```

RDS API

Oracle DB ログは、RDS API を使用して発行することができます。以下のパラメータを使用して、ModifyDBInstance アクションを呼び出せます。

- DBInstanceIdentifier
- CloudwatchLogsExportConfiguration

> **Note**
>
> CloudwatchLogsExportConfiguration パラメータへの変更は常に DB インスタンスに即時適用されます。それで、ApplyImmediately パラメータは効果がありません。

以下の RDS API オペレーションを呼び出すことで Oracle ログを発行することもできます。

- CreateDBInstance
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

以下のパラメータでこの RDS API オペレーションの 1 つを実行します。

- DBInstanceIdentifier
- EnableCloudwatchLogsExports
- Engine
- DBInstanceClass

実行する RDS オペレーションに応じて、他のパラメータが必要となる場合があります。

アラートログとリスナーログへのアクセス

Amazon RDS コンソールを使用してアラートログを表示できます。次の SQL ステートメントを使用することもできます。

```
SELECT message_text FROM alertlog;
```

Amazon CloudWatch Logs を使用してリスナーログにアクセスします。

> **Note**
>
> Oracle では、アラートログとリスナーログは 10 MB を超えるとローテーションされます。その時点で、Amazon RDS のビューからは使用できなくなります。

RDS for PostgreSQL データベースログファイル

次の種類の ログファイルをモニタリングできます。

- PostgreSQL ログ
- アップグレードログ
- IAM データベース認証エラーログ

> **Note**
>
> IAM データベース認証エラーログを有効にするには、まず RDS for PostgreSQL DB インスタンスの IAM データベース認証を有効にする必要があります。IAM データベース認証の有効化の詳細については、「IAM データベース認証の有効化と無効化」を参照してください。

RDS for PostgreSQL では、データベースアクティビティをデフォルトの PostgreSQL ログファイルに記録します。オンプレミスの PostgreSQL DB インスタンスの場合、これらのメッセージは `log/postgresql.log` にローカルに保存されます。、RDS for PostgreSQL DB インスタンスの場合、ログファイルは Amazon RDS インスタンスにあります。これらのログには、AWS マネジメントコンソール 経由でアクセスすることもできます。ここでは、ログを表示またはダウンロードできます。デフォルトのロギングレベルは、ログインの失敗、致命的なサーバーエラー、デッドロック、およびクエリエラーをキャプチャします。

ファイルベースのデータベースログの表示、ダウンロード、モニタリングの方法の詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。PostgreSQL ログの詳細については、「Amazon RDS および Aurora PostgreSQL ログの操作: パート 1」および「Amazon RDS および Aurora PostgreSQL ログの操作: パート 2」を参照してください。

このトピックで説明した標準の PostgreSQL ログに加えて、 RDS for PostgreSQL は PostgreSQL 監査エクステンション (pgAudit) もサポートしています。規制対象の業界や政府機関のほとんどは、法的要件に準拠するために、データに加えられた変更の監査ログまたは監査証跡を維持する必要があります。pgAudit のインストールおよび使用の詳細については、「pgAudit を使用してデータベースのアクティビティを記録する」を参照してください。

トピック

- RDS for PostgreSQL でのログ記録のパラメータ
- RDS for PostgreSQL DB インスタンスのクエリログ記録をオンにする

- Amazon CloudWatch Logs への PostgreSQL ログの発行

RDS for PostgreSQL でのログ記録のパラメータ

さまざまなパラメータを変更することで、 RDS for PostgreSQL DB インスタンスのロギング動作をカスタマイズできます。次のテーブルには、ログの保存期間、ログをローテーションするタイミング、ログを CSV (カンマ区切り値) 形式で出力するかどうかなどに影響するパラメータがあります。他の設定の中でも、STDERR に送信されたテキスト出力を確認できます。変更可能なパラメータの設定を変更するには、のカスタム DB パラメータグループを使用します。RDS for PostgreSQL インスタンス。詳細については、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。

パラメータ	デフォルト	説明
log_destination	stderr	ログの出力形式を設定します。デフォルトは `stderr` ですが、設定に `csvlog` を追加してカンマ区切り値 (CSV) を指定することもできます。詳細については、「ログの送信先の設定 (stderr、csvlog)」を参照してください。
log_filename	postgresql.log.%Y-%m-%d-%H	ログファイル名のパターンを指定します。デフォルトに加えて、このパラメータはファイル名パターンの `postgresql.log.%Y-%m-%d` と `postgresql.log.%Y-%m-%d-%H%M` をサポートします。
log_line_prefix	%t:%r:%u@%d:[%p]:	時間 (%t)、リモート ホスト (%r)、ユーザー (%u)、データベース (%d)、およびプロセス ID (%p) を記録するために、`stderr` に書き込まれる各ログ行のプレフィックスを定義します。
log_rotation_age	60	ログファイルが自動的にローテーションされるまでの分数。この値は、1 ~ 1,440 分の間で変更できます。詳細については、「ログファイルのローテーションの設定」を参照してください。

パラメータ	デフォルト	説明
log_rotation_size	–	ログが自動的にローテーションされるサイズ (kB)。ログは `log_rotation_age` パラメータに基づいてローテーションされるため、デフォルトではこのパラメータは使用されません。詳細については[ログファイルのローテーションの設定]を参照してください。
rds.log_retention_period	4320	指定した時間 (分) より古い PostgreSQL ログは削除されます。デフォルト値の 4,320 分では、3 日後にログファイルが削除されます。詳細については、「[ログの保持期間の設定]」を参照してください。

アプリケーションの問題を特定するには、ログでクエリの失敗、ログインの失敗、デッドロック、および致命的なサーバーエラーを探すことができます。例えば、従来のアプリケーションを Oracle から Amazon RDS PostgreSQL に変換したが、一部のクエリは正しく変換されなかったとします。これらの誤った形式のクエリは、ログにエラーメッセージを生成し、ログから問題を特定することができます。クエリログの詳細については、「[RDS for PostgreSQL DB インスタンスのクエリログ記録をオンにする]」参照してください。

次のトピックでは、PostgreSQL ログの基本的な詳細を制御するさまざまなパラメータの設定方法について説明します。

トピック

- [ログの保持期間の設定]
- [ログファイルのローテーションの設定]
- [ログの送信先の設定 (stderr、csvlog)]
- [log_line_prefix パラメータの概要]

ログの保持期間の設定

`rds.log_retention_period` パラメータは、 RDS for PostgreSQL DB インスタンスがログファイルを保持する期間を指定します。デフォルトの設定は 3 日 (4,320 分) ですが、この値を 1 日

(1,440 分) から 7 日 (10,080 分) までの任意の時間に設定できます。 RDS for PostgreSQL DB インスタンスに、一定期間ログファイルを保持するのに十分なストレージがあることを確認してください。

ログを定期的に Amazon CloudWatch Logs に公開することをお勧めします。これにより、ログが から削除された後も、システムデータを表示して分析できます。RDS for PostgreSQL DB インスタンス。詳細については、「Amazon CloudWatch Logs への PostgreSQL ログの発行」を参照してください。

ログファイルのローテーションの設定

Amazon RDS は、デフォルトで 1 時間ごとに新しいログファイルを作成します。このタイミングは、`log_rotation_age` パラメータによって制御されます。このパラメータのデフォルト値は 60 (分) ですが、1 分から 24 時間 (1,440 分) までの任意の時間に設定できます。ローテーションの時期になると、新しい個別のログファイルが作成されます。ファイルには、`log_filename` パラメータによって指定されたパターンに従って名前が付けられます。

ログファイルは、`log_rotation_size` パラメータで指定されたサイズに従ってローテーションすることもできます。このパラメータは、ログが指定されたサイズ (キロバイト単位) に達したときにローテーションされるように指定します。RDS for PostgreSQL DB インスタンスの場合、`log_rotation_size` は未設定です。つまり、値が指定されていません。ただし、このパラメータは 0 ~ 2,097,151 KB (キロバイト) の範囲で設定できます。

ログファイル名は、`log_filename` パラメータで指定されたファイル名のパターンに基づきます。このパラメータに使用できる設定は次のとおりです。

- `postgresql.log.%Y-%m-%d` — ログファイル名のデフォルトフォーマット。年、月、日をログファイルの名前に含めます。
- `postgresql.log.%Y-%m-%d-%H` — ログファイル名形式に時間を含めます。

詳細については、PostgreSQL ドキュメントの「`log_rotation_age`」と「`log_rotation_size`」を参照してください。

ログの送信先の設定 (**`stderr`**、**`csvlog`**)

デフォルトでは、 Amazon RDS PostgreSQL はスタンダードエラー (stderr) 形式でログを生成します。この形式は、`log_destination` パラメータのデフォルト設定です。各メッセージには、`log_line_prefix` パラメータで指定したパターンを使用してプレフィックスが付きます。詳細については、「log_line_prefix パラメータの概要」を参照してください。

RDS for PostgreSQL は、csvlog フォーマットでログを生成することもできます。csvlog は、ログデータをカンマ区切り値 (CSV) データとして分析する場合に便利です。例えば、log_fdw 拡張機能を使用して外部テーブルとしてログを使用するとします。stderr ログファイルについて作成された外部テーブルには、ログイベントデータを含む 1 つの列が含まれます。log_destination パラメータに csvlog を追加すると、外部テーブルの複数の列の区切りを含む CSV 形式のログファイルが取得できます。ログをより簡単に分類して分析できるようになりました。log_fdw を csvlog を指定して使用する方法については、「SQL を使用した DB ログのアクセスのための log_fdw 拡張機能の使用」を参照してください。

このパラメータに csvlog を指定する場合、stderr ファイル と csvlog ファイル の両方が生成されることに注意してください。ログのストレージと回転率に影響する rds.log_retention_period とその他の設定を考慮し、ログによって消費されるストレージに注意してください。stderr と csvlog を使用すると、ログで消費されるストレージが 2 倍以上になります。

log_destination に csvlog を追加して、stderr だけに戻す場合は、パラメータをリセットする必要があります。そのためには、Amazon RDS コンソールを開いて、インスタンスのカスタム DB パラメータグループを開きます。log_destination パラメータを選択し、[Edit parameter] (パラメータの編集) を選択し、[Reset] (リセット) を選択します。

ログの設定の詳細については、「Amazon RDS および Aurora PostgreSQL ログの操作:パート 1」を参照してください。

log_line_prefix パラメータの概要

stderr ログ形式では、各ログメッセージの先頭に log_line_prefix パラメータで指定した詳細が追加されます。デフォルト値は次のとおりです。

```
%t:%r:%u@%d:[%p]:t
```

Aurora PostgreSQL バージョン 16 以降では、以下を選択することもできます。

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

stderr に送信される各ログエントリには、選択した値に応じて次の情報が含まれます。

- %t – ミリ秒なしのログエントリの時刻
- %m – ミリ秒付きのログエントリの時刻
- %r – リモートホストのアドレス。

- %u@%d – ユーザー名 @ データベース名。
- [%p] – プロセス ID (使用可能な場合)。
- %l – セッションあたりのログ行数
- %e – SQL エラーコード
- %s – プロセス開始時間
- %v – 仮想トランザクション ID
- %x – トランザクション ID
- %c – セッション ID
- %q – セッション以外のターミネータ
- %a – アプリケーション名

RDS for PostgreSQL DB インスタンスのクエリログ記録をオンにする

次のテーブルに示すパラメータの一部を設定することで、クエリ、ロック待ちのクエリ、チェックポイント、その他多くの詳細を含む、データベースアクティビティに関するより詳細な情報を収集できます。このトピックでは、クエリのログ記録に焦点を当てます。

パラメータ	デフォルト	説明
log_connections	–	成功した各接続をログに記録します。
log_disconnections	–	各セッションの終了とその期間を記録します。
log_checkpoints	1	各チェックポイントをログに記録します。
log_lock_waits	–	長期間にわたるロックの待機をログに記録します。デフォルトでは、このパラメータは設定されていません。
log_min_duration_sample	–	(ms) ステートメントのサンプリングに関する最小実行時間を設定します。この値を超えるとステートメントがサンプリングされてログに記録されます。サンプルサイズは、log_statement_sample_rate パラメータを使用して設定されます。

パラメータ	デフォルト	説明
log_min_duration_statement	–	少なくとも指定された時間以上実行された SQL ステートメントはすべてログに記録されます。デフォルトでは、このパラメータは設定されていません。このパラメータを有効にすると、最適化されていないクエリを見つけるために役立ちます。
log_statement	–	ログに記録するステートメントのタイプを設定します。デフォルトでは、このパラメータは設定されていませんが、all、ddl、または mod に変更して、ログに記録する SQL ステートメントのタイプを指定できます。このパラメータの none 以外を指定する場合は、ログファイル内のパスワードが漏洩しないように、追加の手順も実行する必要があります。詳細については、「クエリのログ記録を使用する際のパスワード漏洩リスクの軽減」を参照してください。
log_statement_sample_rate	–	log_min_duration_sample で指定された時間を超えるステートメントがログに記録される割合で、0.0 から 1.0 の間の浮動小数点値で表されます。
log_statement_stats	–	累積処理のパフォーマンスの統計情報をサーバーログに書き込みます。

ログ記録を使用してパフォーマンスの低いクエリを見つける

SQL ステートメントとクエリをログに記録すると、パフォーマンスの悪いクエリを見つけるのに役立ちます。この機能を有効にするには、このセクションで説明されているとおり、log_statement および log_min_duration パラメータの設定を変更します。 RDS for PostgreSQL DB インスタンス、のクエリログ記録を有効にする前に、ログにパスワードが漏洩する可能性と、そのリスクを軽減する方法について知っておく必要があります。詳細については、「クエリのログ記録を使用する際のパスワード漏洩リスクの軽減」を参照してください。

`log_statement` および `log_min_duration` のパラメータに関する参照情報は、以下を参照してください。

log_statement

このパラメータは、ログに送信する SQL ステートメントのタイプを指定します。デフォルト値は none です。このパラメータを `all`、`ddl`、または `mod` に変更する場合は、ログにパスワードが漏洩するリスクを軽減するために、必ず推奨アクションを適用してください。詳細については、「クエリのログ記録を使用する際のパスワード漏洩リスクの軽減」を参照してください。

すべて

すべてのステートメントを記録します。この設定はデバッグ目的での使用を推奨します。

ddl

CREATE、ALTER、DROP などのすべてのデータ定義言語 (DDL) ステートメントをログに記録します。

mod

データを変更する DDL ステートメントと、INSERT、UPDATE、DELETE などのデータ操作言語 (DML) ステートメントをすべてログに記録します。

なし

SQL ステートメントはログに記録されません。ログにパスワードが漏れてしまうリスクを避けるため、この設定をお勧めします。

log_min_duration_statement

少なくとも指定された時間以上実行された SQL ステートメントはすべてログに記録されます。デフォルトでは、このパラメータは設定されていません。このパラメータを有効にすると、最適化されていないクエリを見つけるために役立ちます。

–1–2147483647

ステートメントがログに記録される実行時間のミリ秒 (ms) 数。

クエリのログ記録を設定するには

これらのステップは、RDS for PostgreSQL DB インスタンスはカスタム DB パラメータグループを使用します。

1. log_statement パラメータを all に設定します。以下の例に示しているのは、このパラメータ設定で postgresql.log ファイルに書き込まれる情報です。

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. log_min_duration_statement パラメータを設定します。以下の例に示しているのは、パラメータを postgresql.log に設定したときに 1 ファイルに書き込まれる情報です。

   log_min_duration_statement パラメータで指定された期間を超えるクエリはログに記録されます。例を以下に示します。 RDS for PostgreSQL DB インスタンスのログファイルは Amazon RDS コンソールで表示できます。

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
```

```
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
---------------------- END OF LOG ----------------------
```

クエリのログ記録を使用する際のパスワード漏洩リスクの軽減

パスワードが漏洩しないように、log_statement を none に設定したままにしておくことをお勧めします。log_statement を all、ddl、または mod に設定した場合は、次の手順を 1 つ以上実行することをお勧めします。

- クライアントの場合は、機密情報を暗号化します。詳細については、PostgreSQL ドキュメントの「暗号化オプション」を参照してください。CREATE および ALTER ステートメントの ENCRYPTED (およびUNENCRYPTED) オプションを使用してください。詳細については、PostgreSQL のドキュメントの「CREATE USER」を参照してください。
- RDS for PostgreSQL DB インスタンスでは 、PostgreSQL 監査 (pgAudit) 拡張機能をセットアップして使用します。この拡張機能は、ログに送信された CREATE および ALTER ステートメントの機密情報を編集します。詳細については、「pgAudit を使用してデータベースのアクティビティを記録する」を参照してください。
- CloudWatch ログへのアクセスを制限します。
- IAM など、より強力な認証メカニズムを使用してください。

Amazon CloudWatch Logs への PostgreSQL ログの発行

PostgreSQL ログレコードを高い耐久性の高いのストレージに保存するには、Amazon CloudWatch Logs を使用できます。CloudWatch Logs では、ログデータのリアルタイム分析を実行したり、CloudWatch を使用してメトリクスを表示したり、アラームを作成したりすることもできます。例えば、log_statement を ddl に設定した場合、DDL ステートメントが実行されるたびに警告するアラームを設定できます。RDS for PostgreSQL DB インスタンスの作成プロセス中に PostgreSQL が CloudWatch Logs のログをアップロードするように選択できます。その時点でログをアップロードしないことを選択した場合は、後でインスタンスを変更してその時点からログのアップロードを開始できます。つまり、既存のログはアップロードされません。変更した RDS for PostgreSQL DB インスタンスで作成された新しいログのみがアップロードされます。

現在の PostgreSQL バージョンで利用可能なすべての RDS は、CloudWatch Logs へのログファイルの発行をサポートしています。詳細については、「Amazon RDS for PostgreSQL リリースノート」の「Amazon RDS for PostgreSQL の更新」を参照してください。

CloudWatch Logs を操作するには、ログデータをロググループに発行するように PostgreSQL DB インスタンス用 RDS を設定します。

RDS for PostgreSQL では、次のログの種類を CloudWatch Logs に発行できます。

- PostgreSQL ログ
- アップグレードログ
- IAM データベース認証エラーログ

設定が完了すると、Amazon RDS はログイベントを CloudWatch ロググループのログストリーミングに発行します。例えば、PostgreSQL ログデータは `/aws/rds/instance/my_instance/postgresql` ロググループに保存されます。ログを表示するには、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。

コンソール

コンソールを使用して CloudWatch Logs に PostgreSQL ログを発行するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択してから、[変更] を選択します。
4. [ログのエクスポート] セクションで、CloudWatch Logs に公開するログを選択します。

   [ログのエクスポート] セクションは、CloudWatch Logs への発行をサポートしている PostgreSQL バージョンでのみ使用できます。
5. [続行] を選択し、概要ページで [Modify DB Instance] (DB インスタンスの変更) を選択します。

AWS CLI

PostgreSQL ログは、AWS CLI を使用して発行することができます。以下のパラメータを使用して、`modify-db-instance` コマンドを呼び出せます。

- `--db-instance-identifier`
- `--cloudwatch-logs-export-configuration`

> **Note**
>
> `--cloudwatch-logs-export-configuration` オプションへの変更は常に DB インスタンスに即時適用されます。それで、`--apply-immediately` と `--no-apply-immediately` オプションは効果がありません。

以下の CLI コマンドを呼び出すことで PostgreSQL ログを発行することもできます。

- `create-db-instance`
- `restore-db-instance-from-db-snapshot`
- `restore-db-instance-to-point-in-time`

以下のオプションを使用して、これらの CLI コマンドの 1 つを実行します。

- `--db-instance-identifier`
- `--enable-cloudwatch-logs-exports`
- `--db-instance-class`
- `--engine`

実行する CLI コマンドに応じて、他のオプションが必要となる場合があります。

Example CloudWatch Logs にログを発行するようにインスタンスを変更する

次の例では、ログファイルが CloudWatch Logs に発行されるよう既存の PostgreSQL DB インスタンスを変更します。`--cloudwatch-logs-export-configuration` 値は JSON オブジェクトです。このオブジェクトのキーは `EnableLogTypes` であり、値は `postgresql` と `upgrade` を任意に組み合わせた文字列の配列です。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["postgresql",
 "upgrade"]}'
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","upgrade"]}'
```

Example CloudWatch Logs にログを発行するようにインスタンスを作成する

次の例では、PostgreSQL DB インスタンスを作成してログファイルを CloudWatch Logs に発行します。`--enable-cloudwatch-logs-exports` 値は、JSON 形式の文字列の配列です。この文字列は `postgresql` と `upgrade` の任意の組み合わせです。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --enable-cloudwatch-logs-exports '["postgresql","upgrade"]' \
    --db-instance-class db.m4.large \
    --engine postgres
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --enable-cloudwatch-logs-exports '["postgresql","upgrade"]' ^
    --db-instance-class db.m4.large ^
    --engine postgres
```

RDS API

PostgreSQL ログは、RDS API を使用して発行することができます。以下のパラメータを使用して、ModifyDBInstance アクションを呼び出せます。

- `DBInstanceIdentifier`
- `CloudwatchLogsExportConfiguration`

> **Note**
>
> `CloudwatchLogsExportConfiguration` パラメータへの変更は常に DB インスタンスに即時適用されます。それで、`ApplyImmediately` パラメータは効果がありません。

以下の RDS API オペレーションを呼び出すことで PostgreSQL ログを発行することもできます。

- `CreateDBInstance`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceToPointInTime`

以下のパラメータでこの RDS API オペレーションの 1 つを実行します。

- `DBInstanceIdentifier`
- `EnableCloudwatchLogsExports`
- `Engine`
- `DBInstanceClass`

実行するオペレーションに応じて、他のパラメータが必要となる場合があります。

AWS CloudTrail での Amazon RDS API コールのモニタリング

AWS CloudTrail は、AWS アカウントの監査に役立つ AWS のサービスです。AWS CloudTrail は、AWS アカウントを作成すると、そのアカウントでオンに切り替わります。CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

トピック

- CloudTrail と Amazon RDS の統合
- Amazon RDS ログファイルエントリ

CloudTrail と Amazon RDS の統合

すべての Amazon RDS アクションが、CloudTrail によってログ記録されます。CloudTrail では、Amazon RDS のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を確認できます。

CloudTrail のイベント

CloudTrail が、Amazon RDS のAPI コールをイベントとしてキャプチャします。イベント は、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。イベントには、Amazon RDS コンソールからの呼び出しと、Amazon RDS API 操作へのコード呼び出しが含まれます。

Amazon RDS アクティビティは、[Event history] (イベント履歴) の CloudTrail イベントに記録されます。CloudTrail コンソールを使用して、AWS リージョンの過去 90 日間に記録された API アクティビティとイベントを表示できます。詳細については、CloudTrail イベント履歴でのイベントの表示を参照してください。

CloudTrail 証跡

AWS アカウントのイベント (Amazon RDS のイベントなど) を継続的に記録するには、証跡を作成します。証跡とは、指定した Amazon S3 バケットにイベントを配信するという設定です。CloudTrail は、通常、アカウントアクティビティから 15 分以内にログファイルを配信します。

> **Note**
>
> 追跡を設定しない場合でも、CloudTrail コンソールの イベント履歴で最新のイベントを表示できます。

AWS アカウントには、すべてのリージョンに適用される証跡と、1 つのリージョンに適用される証跡の 2 種類の証跡を作成できます。デフォルトでは、コンソールで証跡を作成すると、すべてのリージョンに証跡が適用されます。

さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 証跡を作成するための概要
- 「CloudTrail がサポートされているサービスと統合」
- 「CloudTrail の アマゾン SNS 通知の設定」
- CloudTrail ログファイルを複数のリージョンから受け取ると複数のアカウントから CloudTrail ログファイルを受け取る

Amazon RDS ログファイルエントリ

CloudTrail のログファイルには、単一か複数のログエントリがあります。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、`CreateDBInstance` アクションを示す CloudTrail ログエントリの例です。

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
```

```
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
```

```
                "subnetStatus": "Active"
            },
            {
                "subnetAvailabilityZone": {"name": "us-east-1e"},
                "subnetIdentifier": "subnet-d7c825e8",
                "subnetStatus": "Active"
            },
            {
                "subnetAvailabilityZone": {"name": "us-east-1f"},
                "subnetIdentifier": "subnet-6746046b",
                "subnetStatus": "Active"
            },
            {
                "subnetAvailabilityZone": {"name": "us-east-1c"},
                "subnetIdentifier": "subnet-bac383e0",
                "subnetStatus": "Active"
            },
            {
                "subnetAvailabilityZone": {"name": "us-east-1d"},
                "subnetIdentifier": "subnet-42599426",
                "subnetStatus": "Active"
            },
            {
                "subnetAvailabilityZone": {"name": "us-east-1a"},
                "subnetIdentifier": "subnet-da327bf6",
                "subnetStatus": "Active"
            }
        ],
        "vpcId": "vpc-136a4c6a",
        "subnetGroupStatus": "Complete"
    },
    "masterUsername": "myawsuser",
    "multiAZ": false,
    "autoMinorVersionUpgrade": true,
    "engine": "mysql",
    "cACertificateIdentifier": "rds-ca-2015",
    "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
    "dBSecurityGroups": [],
    "pendingModifiedValues": {
        "masterUserPassword": "****",
        "pendingCloudwatchLogsExports": {
            "logTypesToEnable": [
                "audit",
                "error",
```

```
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


前の例の userIdentity 要素に示すように、すべてのイベントまたはログエントリには、誰がリクエストを生成したかに関する情報が含まれています。この ID 情報は以下のことを確認するのに役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

userIdentity の詳細については、「CloudTrail userIdentity 要素」を参照してください。CreateDBInstance およびその他の Amazon RDS アクションの詳細については、「Amazon RDS API リファレンス」を参照してください。

データベースアクティビティストリームを使用した Amazon RDS のモニタリング

データベースアクティビティストリームを使用すると、データベースアクティビティのストリームをほぼリアルタイムでモニタリングできます。

トピック

- データベースアクティビティストリーミングの概要
- Oracle Database の統合監査の設定
- Amazon RDS for Microsoft SQL Server の監査ポリシーの設定
- データベースアクティビティストリーミングのスタート
- Amazon RDS のデータベースアクティビティストリーミングの変更
- データベースアクティビティストリーミングのステータスの取得
- データベースアクティビティストリーミングの停止
- データベースアクティビティストリーミングのモニタリング
- データベースアクティビティストリームの IAM ポリシーの例

データベースアクティビティストリーミングの概要

Amazon RDS データベース管理者として、データベースを保護し、コンプライアンスおよび規制要件を満たす必要があります。1 つの戦略は、データベースアクティビティストリーミングをモニタリングツールに統合することです。このようにして、データベースでモニタリングを行い、監査アクティビティのアラームを設定します。

セキュリティの脅威は、外部と内部の両方です。内部の脅威から保護するために、データベースアクティビティストリーミング機能を設定して、データストリームへの管理者アクセスを制御できます。Amazon RDS DBA には、ストリームの収集、送信、保存、および処理に対するアクセス権限がありません。

目次

- データベースアクティビティストリーミングの機能
- Oracle データベースおよび Microsoft SQL Server データベースでの監査
  - Oracle Database での統合監査
  - Microsoft SQL Server での監査

  - Oracle Database および SQL Server の非ネイティブ監査フィールド
  - DB パラメータグループオーバーライド
- データベースアクティビティストリーミングの非同期モード
- データベースアクティビティストリームの要件と制限
- 利用可能なリージョンとバージョン
- データベースアクティビティストリーミングでサポートされる DB インスタンスクラス

データベースアクティビティストリーミングの機能

Amazon RDS は、アクティビティをほぼリアルタイムで Amazon Kinesis データストリームにプッシュします。Kinesis ストリーミングが自動的に作成されます。Kinesis から、Amazon Data Firehose や AWS Lambda などの AWS サービスを設定して、ストリーミングを消費し、データを保存できます。

> **Important**
>
> Amazon RDS のデータベースアクティビティストリーム機能は無料でご利用いただけますが、Amazon Kinesis のデータストリームに対しては課金されます。詳細については、「Amazon Kinesis Data Streams の料金表」を参照してください。

コンプライアンス管理のためのアプリケーションを設定して、データベースアクティビティストリーミングを使用できます。このようなアプリケーションでストリームを使用して、データベースについてのアラートと監査アクティビティを生成できます。

Amazon RDS は、マルチ AZ 配置でのデータベースアクティビティストリームをサポートしています。この場合、データベースアクティビティストリーミングはプライマリインスタンスとスタンバイインスタンスの両方を監査します。

Oracle データベースおよび Microsoft SQL Server データベースでの監査

監査とは、設定されたデータベースアクションをモニタリングし記録することです。Amazon RDS は、デフォルトではデータベースアクティビティをキャプチャしません。データベースで監査ポリシーを自分で作成し、管理します。

トピック

- Oracle Database での統合監査

- Microsoft SQL Server での監査
- Oracle Database および SQL Server の非ネイティブ監査フィールド
- DB パラメータグループオーバーライド

Oracle Database での統合監査

Oracle データベースでは、統合監査ポリシーとは、ユーザーの動作のある面を監査するために使用できる監査設定の名前付きグループです。単一のユーザーのアクティビティを監査するといったシンプルなポリシーもあり得ます。条件を使用する複雑な監査ポリシーも作成できます。

Oracle データベースは、SYS 監査レコードを含む監査レコードを統合監査証跡に書き込みます。例えば、INSERT ステートメントでエラーが発生した場合、標準監査では、エラー番号と実行された SQL が示されます。監査証跡は、AUDSYS スキーマの読み取り専用テーブルに格納されます。これらのレコードにアクセスするには、UNIFIED_AUDIT_TRAIL データディクショナリビューにクエリを実行します。

通常、データベースアクティビティストリーミングは次のように設定します。

1. CREATE AUDIT POLICY コマンドを使用して、Oracle Database の監査ポリシーを作成します。

   Oracle Database は、監査レコードを生成します。
2. AUDIT POLICY コマンドを使用して、監査ポリシーを有効にします。
3. データベースアクティビティストリーミングを設定します。

   Oracle Database の監査ポリシーに適合するアクティビティのみがキャプチャされ、Amazon Kinesis データストリーミングに送信されます。データベースアクティビティストリーミングが有効な場合、Oracle データベース管理者は監査ポリシーを変更したり、監査ログを削除したりできません。

統合監査ポリシーの詳細については、Oracle Database セキュリティガイドの「統合監査ポリシーおよび AUDIT を使用した監査アクティビティについて」を参照してください。

Microsoft SQL Server での監査

データベースアクティビティストリームは、SQLAudit 機能を使用して SQL Server データベースを監査します。

RDS for SQL Server インスタンスには、以下が含まれます。

- サーバー監査 — SQL サーバー監査では、サーバーレベルまたはデータベースレベルのアクションの単一のインスタンスと、監視するアクションのグループを収集します。サーバーレベルの監査 RDS_DAS_AUDIT と RDS_DAS_AUDIT_CHANGES は、RDS によって管理されます。
- サーバー監査仕様 — サーバー監査仕様は、サーバーレベルのイベントを記録します。RDS_DAS_SERVER_AUDIT_SPEC 仕様は変更できます。この仕様はサーバー監査 RDS_DAS_AUDIT にリンクされています。RDS_DAS_CHANGES_AUDIT_SPEC 仕様は RDS によって管理されます。
- データベース監査仕様 — データベース監査仕様は、データベースレベルのイベントを記録します。データベース監査仕様 RDS_DAS_DB_<name> を作成し、それを RDS_DAS_AUDIT サーバー監査にリンクできます。

データベースアクティビティストリームは、コンソールまたは CLI を使用して設定できます。通常、データベースアクティビティストリーミングは次のように設定します。

1. (オプション) CREATE DATABASE AUDIT SPECIFICATION コマンドを使用してデータベース監査仕様を作成し、それを RDS_DAS_AUDIT サーバー監査にリンクします。
2. (オプション) ALTER SERVER AUDIT SPECIFICATION コマンドを使用してサーバー監査仕様を変更し、ポリシーを定義します。
3. データベースおよびサーバー監査ポリシーを有効にします。例えば、次のようになります。

   ```
   ALTER DATABASE AUDIT SPECIFICATION [<Your database specification>] WITH
   (STATE=ON)

   ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC] WITH
   (STATE=ON)
   ```
4. データベースアクティビティストリーミングを設定します。

   サーバーおよびデータベース監査ポリシーに一致するアクティビティのみがキャプチャされ、Amazon Kinesis データストリームに送信されます。データベースアクティビティストリームが有効であり、ポリシーがロックされているときには、データベース管理者は監査ポリシーを変更したり、監査ログを削除したりできません。

> **Important**
>
> 特定のデータベースのデータベース監査仕様が有効であり、ポリシーがロック状態の場合、データベースを削除することはできません。

詳細については、Microsoft SQL Server ドキュメントの「SQL Server 監査コンポーネント」を参照してください。

Oracle Database および SQL Server の非ネイティブ監査フィールド

データベースアクティビティストリーミングをスタートすると、すべてのデータベースイベントが対応するアクティビティストリーミングイベントを生成します。例えば、データベースユーザーが SELECT ステートメントと INSERT ステートメントを実行したとします。データベースは、これらのイベントを監査し、Amazon Kinesis データストリーミングに送信します。

イベントは、ストリーミングの中で、JSON オブジェクトとして表されます。JSON オブジェクトには、DatabaseActivityMonitoringRecord が含まれ、これには databaseActivityEventList アレイが含まれています。アレイ内の定義済みフィールドには、class、clientApplication、commandなどがあります。

デフォルトでは、アクティビティストリーミングにはエンジンネイティブの監査フィールドは含まれません。Amazon RDS for Oracle と SQL Server を設定して、これらの追加フィールドを engineNativeAuditFields JSON オブジェクトに含めることができます。

Oracle Database では、統合監査証跡のほとんどのイベントは、RDS データアクティビティストリームのフィールドにマップされます。例えば、統合監査の UNIFIED_AUDIT_TRAIL.SQL_TEXT フィールドは、データベースアクティビティストリーミング内の commandText フィールドにマッピングされます。ただし、OS_USERNAME などの Oracle Database 監査フィールドは、データベースアクティビティストリーミングの定義済みフィールドにマッピングされません。

SQL Server では、SQLAudit によって記録されるイベントのフィールドのほとんどが RDS データベースアクティビティストリームのフィールドにマップされます。例えば、監査の sys.fn_get_audit_file のcode フィールドは、データベースアクティビティストリームの commandText フィールドにマップされます。ただし、permission_bitmask などの SQL Server データベース監査フィールドは、データベースアクティビティストリームの定義済みフィールドにマップされません。

databaseActivityEventList の詳細については、「データベースアクティビティストリームの databaseActivityEventList JSON 配列」を参照してください。

DB パラメータグループオーバーライド

通常、RDS for Oracle で統一監査を有効にするには、パラメータグループをアタッチします。ただし、データベースアクティビティストリームには、追加の設定が必要です。カスタマーエクスペリエンスを向上させるため、Amazon RDS では以下を実行しています。

- アクティビティストリームを有効にすると、RDS for Oracle はパラメータグループの監査パラメータを無視します。
- アクティビティストリームを無効にすると、RDS for Oracle は監査パラメータの無視を停止します。

SQL Server のデータベースアクティビティストリームは、SQL Audit オプションで設定したパラメータとは無関係です。

データベースアクティビティストリーミングの非同期モード

Amazon RDS のアクティビティストリームは、常に非同期です。データベースセッションでアクティビティストリーミングイベントが生成されると、セッションは直ちに通常のアクティビティに戻ります。バックグラウンドでは、Amazon RDS によりアクティビティストリームイベントが永続的なレコードになります。

バックグラウンドタスクでエラーが発生した場合は、Amazon RDS はイベントを生成します。このイベントは、アクティビティストリーミングのイベントレコードが失われた可能性がある時間枠のスタートと終了を示します。非同期モードでは、アクティビティストリーミングの精度よりもデータベースのパフォーマンスが優先されます。

データベースアクティビティストリームの要件と制限

RDS では、データベースアクティビティストリームに、次の要件と制限があります。

- データベースアクティビティストリーミングには、Amazon Kinesis が必要です。
- 常に暗号化されているため、アクティビティストリーミングには AWS Key Management Service (AWS KMS) が必要です。
- Amazon Kinesis データストリームに追加の暗号化を適用することは、AWS KMS キーで暗号化済みのデータベースアクティビティストリーミングと互換性がありません。
- 監査ポリシーは自分で作成して、管理します。Amazon Aurora とは異なり、RDS for Oracle はデフォルトでデータベースアクティビティをキャプチャしません。
- 監査ポリシーまたは仕様は自分で作成して、管理します。Amazon Aurora とは異なり、Amazon RDS はデフォルトではデータベースアクティビティをキャプチャしません。
- マルチ AZ 配置では、プライマリ DB インスタンスのみでデータベースアクティビティストリームを開始してください。アクティビティストリーミングは、プライマリインスタンスとスタンバイ DB インスタンスの両方を自動的に監査します。フェイルオーバー中に追加のステップは必要ありません。

- DB インスタンスの名前を変更しても、新しい Kinesis ストリームは作成されません。
- CDB は、RDS for Oracle ではサポートされません。
- リードレプリカはサポートされていません。

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。データベースアクティビティストリーミングでのバージョンとリージョンの可用性の詳細については、「Amazon RDS のデータベースアクティビティストリームでサポートされているリージョンと DB エンジン」を参照してください。

データベースアクティビティストリーミングでサポートされる DB インスタンスクラス

RDS for Oracle では、次の DB インスタンスクラスでデータベースアクティビティストリームを使用できます。

- db.m4.*large
- db.m5.*large
- db.m5d.*large
- db.m6i.*large
- db.r4.*large
- db.r5.*large
- db.r5.*large.tpc*.mem*x
- db.r5b.*large
- db.r5b.*large.tpc*.mem*x
- db.r5d.*large
- db.r6i.*large
- db.r6i.*large.tpc*.mem*x
- db.x2idn.*large
- db.x2iedn.*large
- db.x2iezn.*large

- db.z1d.*large

RDS for Oracle では、次の DB インスタンスクラスでデータベースアクティビティストリームを使用できます。

- db.m4.*large
- db.m5.*large
- db.m5d.*large
- db.m6i.*large
- db.r4.*large
- db.r5.*large
- db.r5b.*large
- db.r5d.*large
- db.r6i.*large
- db.x1e.*large
- db.x2iedn.*large
- db.z1d.*large

インスタンスクラスのタイプの詳細については、「DB インスタンスクラス」を参照してください。

Oracle Database の統合監査の設定

データベースアクティビティストリーミングで使用する統合監査を設定すると、次のような状況が発生する可能性があります。

- Oracle データベースに対して統合監査は設定されていません。

  この場合、新しいポリシーを CREATE AUDIT POLICY コマンドで作成し、それを AUDIT POLICY コマンドで有効にします。次の例では、特定の権限とロールを持つユーザーをモニタリングするポリシーの作成および有効化を行います。

```
CREATE AUDIT POLICY table_pol
PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
ROLES emp_admin, sales_admin;

AUDIT POLICY table_pol;
```

詳細な手順については、Oracle Database のドキュメントの「監査ポリシーの設定」を参照してください。

- Oracle データベースに対して統合監査が設定されます。

  データベースアクティビティストリームを有効にすると、RDS for Oracle によって既存の監査データが自動的にクリアされます。また、監査証跡権限も取り消されます。RDS for Oracle では、次の処理を実行できなくなります。

  - 統合監査証跡レコードを消去する。
  - 統合監査ポリシーを追加、削除、または変更する。
  - 最後にアーカイブされたタイムスタンプを更新します。

  > **Important**
  >
  > データベースアクティビティストリームを有効にする前に、監査データをバックアップすることを強くお勧めします。

  `UNIFIED_AUDIT_TRAIL` ビューについては、「UNIFIED_AUDIT_TRAIL」を参照してください。Oracle Support のアカウントをお持ちの場合は、「UNIFIED AUDIT TRAIL を消去する方法」を参照してください。

Amazon RDS for Microsoft SQL Server の監査ポリシーの設定

SQL Server データベースインスタンスにはサーバー監査 `RDS_DAS_AUDIT` があり、Amazon RDS によって管理されます。サーバー監査仕様 `RDS_DAS_SERVER_AUDIT_SPEC` にサーバーイベントを記録するポリシーを定義できます。`RDS_DAS_DB_<name>` などのデータベース監査仕様を作成し、データベースイベントを記録するポリシーを定義できます。サーバーレベルとデータベースレベルの監査アクショングループのリストについては、Microsoft SQL Server ドキュメントの「SQL Server 監査アクショングループとアクション」を参照してください。

デフォルトのサーバーポリシーは、失敗したログインと、データベースアクティビティストリームのデータベースまたはサーバー監査仕様の変更のみを監視します。

監査および監査仕様の制限には以下が含まれます。

- データベースアクティビティストリームがロック状態の場合、サーバーまたはデータベースの監査仕様を変更することはできません。

- サーバー監査 RDS_DAS_AUDIT の仕様は変更できません。
- SQL Server 監査 RDS_DAS_CHANGES または関連するサーバー監査仕様 RDS_DAS_CHANGES_AUDIT_SPEC は変更できません。
- データベース監査仕様を作成するときには、RDS_DAS_DB_databaseActions などの形式 RDS_DAS_DB_<name> を使用する必要があります。

> **Important**
>
> 小規模なインスタンスクラスでは、すべてを監査せず、必要なデータだけを監査することをお勧めします。これにより、これらのインスタンスクラスに対するデータベースアクティビティストリームのパフォーマンスへの影響を軽減できます。

次のサンプルコードは、サーバー監査仕様 RDS_DAS_SERVER_AUDIT_SPEC を変更し、ログアウトと成功したログインアクションを監査します。

```
ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC]
      WITH (STATE=OFF);
ALTER SERVER AUDIT SPECIFICATION [RDS_DAS_SERVER_AUDIT_SPEC]
      ADD (LOGOUT_GROUP),
      ADD (SUCCESSFUL_LOGIN_GROUP)
      WITH (STATE = ON );
```

次のサンプルコードは、データベース監査仕様 RDS_DAS_DB_database_spec を作成し、それをサーバー監査 RDS_DAS_AUDIT に添付します。

```
USE testDB;
CREATE DATABASE AUDIT SPECIFICATION [RDS_DAS_DB_database_spec]
     FOR SERVER AUDIT [RDS_DAS_AUDIT]
     ADD ( INSERT, UPDATE, DELETE
          ON testTable BY testUser )
     WITH (STATE = ON);
```

監査仕様を設定したら、仕様 RDS_DAS_SERVER_AUDIT_SPEC および RDS_DAS_DB_<name> が ON の状態に設定されていることを確認します。これで、監査データをデータベースアクティビティストリームに送信できます。

データベースアクティビティストリーミングのスタート

DB インスタンスのアクティビティストリームを開始すると、監査ポリシーで設定したデータベースアクティビティイベントごとに、アクティビティストリームイベントが生成されます。アクセスイベントは CONNECT、SELECT などの SQL コマンドから生成されます。変更イベントは CREATE、INSERT などの SQL コマンドから生成されます。

> **Important**
>
> Oracle DB インスタンスのアクティビティストリーミングを有効にすると、既存の監査データが消去されます。また、監査証跡権限も取り消されます。ストリーミングを有効にすると、RDS for Oracle では次の処理を実行できなくなります。
>
> - 統合監査証跡レコードを消去する。
> - 統合監査ポリシーを追加、削除、または変更する。
> - 最後にアーカイブされたタイムスタンプを更新する。

Console

データベースアクティビティストリーミングをスタートするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択します。
3. アクティビティストリームを有効にする Amazon RDS データベースインスタンスを選択します。マルチ AZ 配置では、プライマリインスタンスのみでストリーミングをスタートします。アクティビティストリーミングは、プライマリインスタンスとスタンバイインスタンスの両方を監査します。
4. [アクション] で [アクティビティストリーミングの開始] を選択します。

   [データベースアクティビティストリーミングの開始: *##*] ウィンドウが表示されます。ここで、*name* は RDS インスタンスです。
5. 以下の設定を入力します。

   - [AWS KMS key] では、 AWS KMS keys のリストからキーを選択します。

     Amazon RDS は、KMS キーを使用してキーを暗号化し、それによってデータベースアクティビティを暗号化します。デフォルトキー以外の KMS キーを選択します。暗号化キー

と AWS KMS の詳細については、AWS Key Management Service デベロッパーガイドの「AWS Key Management Service とは？」を参照してください。

- [データベースアクティビティイベント] で、[エンジンネイティブ監査フィールドを有効にする] を選択して、エンジン固有の監査フィールドを含めます。
- [すぐに適用] を選択します。

  [直ちに] を選択する場合直ちにとすると、RDS インスタンスがすぐに再起動します。を選択すると次のメンテナンス時間中とすると、RDS インスタンスはすぐには再起動しません。この場合、データベースアクティビティストリーミングは、次のメンテナンスウィンドウまでスタートされません。

6. [Start database activity stream] (データベースアクティビティストリームを開始) を選択します。

   データベースのステータスは、アクティビティストリームが開始していることを示します。

> **Note**
>
> `You can't start a database activity stream in this configuration` というエラーが表示された場合は、データベースアクティビティストリーミングでサポートされる DB インスタンスクラス で RDS インスタンスがサポートされているインスタンスクラスを使用しているかを確認してください。

AWS CLI

DB インスタンスのデータベースアクティビティストリームを開始するには、AWS CLI コマンド start-activity-stream を使用して データベースを設定します。

- `--resource-arn` *`arn`* - DB インスタンスの Amazon リソースネーム (ARN) を指定します。
- `--kms-key-id` *`key`* - データベースアクティビティストリーミング内のメッセージを暗号化するための KMS キー識別子を指定します。AWS KMS キー識別子は、キー ARN、キー ID、エイリアス ARN、または AWS KMS key のエイリアス名です。
- `--engine-native-audit-fields-included` - エンジン固有の監査フィールドをデータストリームに含めます。これらのフィールドを除外するには、`--no-engine-native-audit-fields-included` (デフォルト) を指定します。

次の例では、非同期モードで DB インスタンスのデータベースアクティビティストリームを開始します。

Linux、macOS、Unix の場合:

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-instance-arn \
    --engine-native-audit-fields-included \
    --apply-immediately
```

Windows の場合:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-instance-arn ^
    --engine-native-audit-fields-included ^
    --apply-immediately
```

Amazon RDS API

DB インスタンスのデータベースアクティビティストリームを開始するには、StartActivityStream オペレーションを使用してインスタンスを設定します。

以下のパラメータを使用してアクションを呼び出します。

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`
- `EngineNativeAuditFieldsIncluded`

Amazon RDS のデータベースアクティビティストリーミングの変更

アクティビティストリームの開始時に Amazon RDS 監査ポリシーをカスタマイズしたい場合があります。アクティビティストリーミングを停止して時間とデータを失いたくない場合は、ポリシーの状態の監査を以下のいずれかの設定に変更します。

ロック (デフォルト)

データベース内の監査ポリシーは読み取り専用です。

ロック解除

データベース内の監査ポリシーは、読み取り/書き込みが可能です。

基本的なステップは次のとおりです。

1. 監査ポリシーの状態をロック解除に変更します。
2. 監査ポリシーをカスタマイズします。
3. 監査ポリシーの状態をロックに変更します。

コンソール

アクティビティストリーミングの監査ポリシーステータスを変更するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. [Actions] (アクション) で、選択してください[Modify database activity stream] (データベースアクティビティストリーミングの変更) を選択します。

   [Modify database activity stream: ***name***] (データベースアクティビティストリームの変更: 名前) ウィンドウが表示されます。ここで、##は RDS インスタンスです。
4. 次のいずれかのオプションを選択します。

   ロック

   監査ポリシーをロックすると、読み取り専用になります。ポリシーをロック解除するか、アクティビティストリーミングを停止しない限り、監査ポリシーを編集することはできません。

   ロック解除

   監査ポリシーをロック解除すると、読み取り/書き込みが可能になります。アクティビティストリーミングの起動中に監査ポリシーを編集できます。
5. [Modify DB activity stream] (DB アクティビティストリーミングの変更) を選択します。

Amazon RDS データベースのステータスは、[アクティビティストリームの設定中] と表示されます。

6. (オプション) DB インスタンスリンクを選択します。そして、[Configuration (設定)] タブを選択します。

   [Audit policy status] (ポリシーステータスの監査) フィールドには、次のいずれかの値が表示されます。

   - ロック
   - ロック解除
   - ロックのポリシー
   - ロック解除のポリシー

AWS CLI

データベースインスタンスのアクティビティストリームの状態を変更するには、modify-activity-stream AWS CLI コマンドを使用します。

オプション	必須?	[Description] (説明)
`--resource-arn` *my-instance-ARN*	可能	RDS データベースインスタンスの Amazon リソースネーム (ARN)。
`--audit-policy-state`	不可	インスタンスのデータベースアクティビティストリーミングの監査ポリシーの新しい状態: `locked` または `unlocked`。

次の例では、*my-instance-ARN* で起動されたアクティビティストリーミングの監査ポリシーをロック解除します。

Linux、macOS、Unix の場合:

```
aws rds modify-activity-stream \
    --resource-arn my-instance-ARN \
    --audit-policy-state unlocked
```

Windows の場合:

```
aws rds modify-activity-stream ^
    --resource-arn my-instance-ARN ^
    --audit-policy-state unlocked
```

次の例ではインスタンス *my-instance* について説明します。出力例の一部は、監査ポリシーがロック解除されていることを示しています。

```
aws rds describe-db-instances --db-instance-identifier my-instance

{
    "DBInstances": [
        {
            ...
            "Engine": "oracle-ee",
            ...
            "ActivityStreamStatus": "started",
            "ActivityStreamKmsKeyId": "ab12345e-1111-2bc3-12a3-ab1cd12345e",
            "ActivityStreamKinesisStreamName": "aws-rds-das-db-
AB1CDEFG23GHIJK4LMNOPQRST",
            "ActivityStreamMode": "async",
            "ActivityStreamEngineNativeAuditFieldsIncluded": true,
            "ActivityStreamPolicyStatus": "unlocked",
            ...
        }
    ]
}
```

RDS API

データベースアクティビティストリーミングのポリシーステートを変更するには、ModifyActivityStream オペレーションを使用します。

以下のパラメータを使用してアクションを呼び出します。

- `AuditPolicyState`
- `ResourceArn`

データベースアクティビティストリーミングのステータスの取得

Amazon RDS データベースインスタンスのアクティビティストリームのステータスを取得するには、コンソールまたは AWS CLI を使用します。

コンソール

データベースアクティビティストリーミングのステータスを取得するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、DB インスタンスのリンクを選択します。
3. [設定] タブを選択して、ステータスを取得する [データベースアクティビティストリーミング] をオンにします。

AWS CLI

describe-db-instances CLI リクエストに対するレスポンスとして、データベースインスタンスのアクティビティストリーム設定を取得できます。

次の例は、 *my-instance* を説明します。

```
aws rds --region my-region describe-db-instances --db-instance-identifier my-db
```

以下は JSON レスポンスの例です。次のフィールドが表示されます。

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`
- `ActivityStreamMode`
- `ActivityStreamPolicyStatus`

```
{
    "DBInstances": [
        {
            ...
            "Engine": "oracle-ee",
            ...
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "ab12345e-1111-2bc3-12a3-ab1cd12345e",
            "ActivityStreamKinesisStreamName": "aws-rds-das-db-
AB1CDEFG23GHIJK4LMNOPQRST",
            "ActivityStreamMode": "async",
            "ActivityStreamEngineNativeAuditFieldsIncluded": true,
```

```
            "ActivityStreamPolicyStatus": locked",
            ...
        }
    ]
}
```

RDS API

DescribeDBInstances オペレーションに対するレスポンスとして、データベースのアクティビティストリーミングの設定を取得できます。

データベースアクティビティストリーミングの停止

アクティビティストリーミングを停止するには、コンソールまたは AWS CLI を使用します。

Amazon RDS データベースインスタンスを削除すると、アクティビティストリームが停止し、基になる Amazon Kinesis ストリームが自動的に削除されます。

コンソール

アクティビティストリーミングを停止するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. データベースアクティビティストリーミングを停止する データベースを選択します。
4. [アクション] で [アクティビティストリーミングの停止] を選択します。[データベースアクティビティストリーミング] ウィンドウが表示されます。

    a. [すぐに適用] を選択します。

        [直ちに] を選択する場合直ちにとすると、RDS インスタンスがすぐに再起動します。を選択すると次のメンテナンス時間中とすると、RDS インスタンスはすぐには再起動しません。この場合、データベースアクティビティストリーミングは、次のメンテナンスウィンドウまで停止しません。

    b. [続行] を選択します。

AWS CLI

データベースのデータベースアクティビティストリーミングを停止するには、AWS CLI コマンドの stop-activity-stream を使用して DB インスタンスを設定します。のDB インスタンスの AWS リー

ジョンを識別するには、--region パラメータを指定します。--apply-immediately パラメータはオプションです。

Linux、macOS、Unix の場合:

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_DB_ARN \
    --apply-immediately
```

Windows の場合:

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_DB_ARN ^
    --apply-immediately
```

RDS API

データベースのデータベースアクティビティストリーミングを停止するには、StopActivityStream オペレーションを使用して、DB インスタンスを設定します。のDB インスタンスの AWS リージョンを識別するには、Region パラメータを指定します。ApplyImmediately パラメータはオプションです。

データベースアクティビティストリーミングのモニタリング

データベースアクティビティストリーミングは、アクティビティをモニタリングして報告します。アクティビティのストリーミングは、収集後、Amazon Kinesis に送信されます。Kinesis から、アクティビティストリーミングをモニタリングしたり、他のサービスやアプリケーションがアクティビティストリーミングを使用して詳細な分析を行うことができます。基礎となる Kinesis ストリーム名は、AWS CLI コマンドの describe-db-instances または RDS API DescribeDBInstances オペレーションを使用して検索できます。

Amazon RDS は、Kinesis ストリーミングを次のように管理します。

- Amazon RDS は、24 時間の保存期間の Kinesis ストリーミングを自動的に作成します。
- Amazon RDS は、必要に応じて Kinesis ストリーミングをスケールします。
- データベースアクティビティストリーミングを停止したり、DB インスタンスを削除したりすると、Amazon RDS によって Kinesis ストリームが削除されます。

以下のカテゴリのアクティビティがモニタリングされ、アクティビティストリーミングの監査ログに追加されます。

- SQL コマンド - すべての SQL コマンドに加えて、準備済みステートメント、組み込み関数、および PL/SQL の関数も監査されます。ストアドプロシージャへの呼び出しが監査されます。ストアドプロシージャまたは関数内で発行された SQL ステートメントも監査されます。
- 他のデータベース情報 - モニタリングされるアクティビティには、完全な SQL ステートメント、DML コマンドから影響を受ける行の行数、アクセスされたオブジェクト、および一意のデータベース名が含まれます。また、データベースアクティビティストリーミングは、バインド可変とストアドプロシージャパラメータもモニタリングします。

  > **Important**
  >
  > 各ステートメントの完全な SQL テキストは、機密データを含むアクティビティストリーミング監査ログに表示されます。ただし、Oracle が次の SQL ステートメントのようにコンテキストから判断できる場合、データベースユーザーのパスワードは訂正されます。
  >
  > ```
  > ALTER ROLE role-name WITH password
  > ```

- 接続情報 - モニタリングされるアクティビティには、セッションとネットワークの情報、サーバープロセス ID、および終了コードなどがあります。

DB インスタンスのモニタリング中にアクティビティストリーミングに障害が発生した場合は、RDS イベントを通じて通知されます。

以下のセクションでは、データベースアクティビティストリームへのアクセス、監査、処理を行うことができます。

トピック

- Amazon Kinesis からのアクティビティストリーミングへのアクセス
- データベースアクティビティストリームの監査ログコンテンツおよび例
- データベースアクティビティストリームの databaseActivityEventList JSON 配列
- AWS SDK を使用したデータベースアクティビティストリーミングの処理

Amazon Kinesis からのアクティビティストリーミングへのアクセス

データベースのアクティビティストリーミングを有効にすると、Kinesis ストリーミングが作成されます。データベースのアクティビティは、Kinesis からリアルタイムでモニタリングできます。データベースのアクティビティを詳細に分析するには、Kinesis ストリーミングをコンシューマーアプリケーションに接続します。また、IBM の Security Guardium または Imperva の SecureSphere Database Audit and Protection などのコンプライアンス管理アプリケーションにストリーミングを接続することもできます。

Kinesis ストリームには、RDS コンソールまたは Kinesis コンソールからアクセスできます。

RDS コンソールを使用して、Kinesis からアクティビティストリーミングにアクセスするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. アクティビティストリームを開始する Amazon RDS データベースインスタンスを選択します。
4. [設定] を選択します。
5. [Database activity stream] (データベースアクティビティストリーム) で、[Kinesis stream] (Kinesis ストリーム) の下のリンクを選択します。
6. データベースアクティビティの観察を開始するには、Kinesis コンソールで [Monitoring] (モニタリング) を選択します。

Kinesis コンソールを使用して、Kinesis からアクティビティストリーミングにアクセスするには

1. Kinesis コンソール (https://console.aws.amazon.com/kinesis) を開きます。
2. Kinesis ストリーミングのリストからアクティビティストリーミングを選択します。

   アクティビティストリーミングの名前には、プレフィックス `aws-rds-das-db-` が付き、その後に データベースのリソース ID が続きます。次に例を示します。

   ```
   aws-rds-das-db-NHVOV4PCLWHGF52NP
   ```

   Amazon RDS コンソールを使用して データベースのリソース ID を検索するには、データベースのリストから DB インスタンスを選択した上で、[設定] タブを選択します。

AWS CLI を使用して、アクティビティストリーミングの Kinesis ストリームの完全な名前を検索するには、 describe-db-instances CLI リクエストを使用し、そのレスポンスに含まれる `ActivityStreamKinesisStreamName` の値を書き留めます。

3. データベースアクティビティの観察をスタートするには、[モニタリング] を選択します。

Amazon Kinesis の使用の詳細については、「Amazon Kinesis Data Streams とは」を参照してください。

データベースアクティビティストリームの監査ログコンテンツおよび例

モニタリングされるイベントは、データベースアクティビティストリーミングでは JSON 文字列として表されます。この構造は、`DatabaseActivityMonitoringRecord` を含む JSON オブジェクトで構成されます。このオブジェクトには、アクティビティイベントの `databaseActivityEventList` 配列が含まれます。

> **Note**
>
> データベースアクティビティストリームの場合、`paramList` JSON 配列には Hibernate アプリケーションからの null 値は含まれません。

トピック

- アクティビティストリーミングの監査ログの例
- DatabaseActivityMonitoringRecords JSON オブジェクト
- databaseActivityEvents JSON オブジェクト

アクティビティストリーミングの監査ログの例

以下に、アクティビティイベントレコードの復号されたサンプルの JSON 監査ログを示します。

Example CONNECT SQL ステートメント のアクティビティイベントレコード

次のアクティビティイベントレコードは、Oracle DB について、JDBC Thin Client (`clientApplication`) による `CONNECT` SQL ステートメント (`command`) を使用したログインを示しています。

```
{
```

```
    "class": "Standard",
    "clientApplication": "JDBC Thin Client",
    "command": "LOGON",
    "commandText": null,
    "dbid": "0123456789",
    "databaseName": "ORCL",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:15:36.233787",
    "netProtocol": "tcp",
    "objectName": null,
    "objectType": null,
    "paramList": [],
    "pid": 17904,
    "remoteHost": "123.456.789.012",
    "remotePort": "25440",
    "rowCount": null,
    "serverHost": "987.654.321.098",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 987654321,
    "startTime": null,
    "statementId": 1,
    "substatementId": null,
    "transactionId": "0000000000000000",
    "engineNativeAuditFields": {
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": "CREATE SESSION",
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DBID": 123456789
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
```

```
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((ADDRESS
\u003d(PROTOCOL\u003dtcp)(HOST\u003d205.251.233.183)(PORT\u003d25440))));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "a1b2c3d4e5f6.amazon.com",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
        "TERMINAL": "unknown",
        "OS_USERNAME": "sumepate",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
```

```
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 1,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5124715
    }
}
```

次のアクティビティイベントレコードは、SQL Server DB のログイン失敗を示しています。

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "LOGIN",
            "clientApplication": "Microsoft SQL Server Management Studio",
            "command": "LOGIN FAILED",
            "commandText": "Login failed for user 'test'. Reason: Password did not
 match that for the login provided. [CLIENT: local-machine]",
            "databaseName": "",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
```

```
            "errorMessage": null,
            "exitCode": 0,
            "logTime": "2022-10-06 21:34:42.7113072+00",
            "netProtocol": null,
            "objectName": "",
            "objectType": "LOGIN",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 0,
            "startTime": null,
            "statementId": "0x1eb0d1808d34a94b9d3dcf5432750f02",
            "substatementId": 1,
            "transactionId": "0",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 0,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "",
                "target_server_principal_name": "",
                "schema_name": "",
                "is_column_permission": false,
                "object_id": 0,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "<action_info "xmlns=\"http://
schemas.microsoft.com/sqlserver/2008/sqlaudit_data\"><pooled_connection>0</
pooled_connection><error>0x00004818</error><state>8</state><address>local machine</
address><PasswordFirstNibbleHash>B</PasswordFirstNibbleHash></action_info>"-->,
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000000",
                "data_sensitivity_information": "",
                "session_server_principal_name": "",
                "connection_id": "98B4F537-0F82-49E3-AB08-B9D33B5893EF",
```

```
                "audit_schema_version": 1,
                "database_principal_id": 0,
                "server_principal_sid": null,
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

> **Note**
>
> データベースアクティビティストリーミングが有効になっていない場合、JSON ドキュメントの最後のフィールドは "engineNativeAuditFields": { } になります。

Example CREATE TABLE ステートメントのアクティビティイベントレコード

次の例は、Oracle データベースの CREATE TABLE イベントを示しています。

```
{
    "class": "Standard",
    "clientApplication": "sqlplus@ip-12-34-5-678 (TNS V1-V3)",
    "command": "CREATE TABLE",
    "commandText": "CREATE TABLE persons(\n    person_id NUMBER GENERATED BY DEFAULT AS
 IDENTITY,\n    first_name VARCHAR2(50) NOT NULL,\n    last_name VARCHAR2(50) NOT NULL,
\n    PRIMARY KEY(person_id)\n)",
    "dbid": "0123456789",
    "databaseName": "ORCL",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:22:49.535239",
    "netProtocol": "beq",
    "objectName": "PERSONS",
    "objectType": "TEST",
    "paramList": [],
    "pid": 17687,
    "remoteHost": "123.456.789.0",
    "remotePort": null,
```

```
    "rowCount": null,
    "serverHost": "987.654.321.01",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 1234567890,
    "startTime": null,
    "statementId": 43,
    "substatementId": null,
    "transactionId": "090011007F0D0000",
    "engineNativeAuditFields": {
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": "CREATE SEQUENCE, CREATE TABLE",
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((PROTOCOL
\u003dbeq)(HOST\u003d123.456.789.0)));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
```

```
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "ip-10-13-0-122",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
        "TERMINAL": "pts/1",
        "OS_USERNAME": "rdsdb",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 12,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
```

```
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5133083
    }
}
```

次の例は、SQL Server データベースの CREATE TABLE イベントを示しています。

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "SCHEMA",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "ALTER",
            "commandText": "Create table [testDB].[dbo].[TestTable2](\r\ntextA
 varchar(6000),\r\n    textB varchar(6000)\r\n)",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:44:38.4120677+00",
            "netProtocol": null,
            "objectName": "dbo",
            "objectType": "SCHEMA",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 84,
            "startTime": null,
            "statementId": "0x5178d33d56e95e419558b9607158a5bd",
            "substatementId": 1,
            "transactionId": "4561864",
```

```
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "",
                "is_column_permission": false,
                "object_id": 1,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000000",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "EE1FE3FD-EF2C-41FD-AF45-9051E0CD983A",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x010500000000000515000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```


Example SELECT ステートメントのアクティビティイベントレコード

次の例は、Oracle DB の SELECT イベントを示しています。


```
{
    "class": "Standard",
    "clientApplication": "sqlplus@ip-12-34-5-678 (TNS V1-V3)",
    "command": "SELECT",
    "commandText": "select count(*) from persons",
    "databaseName": "1234567890",
    "dbProtocol": "oracle",
    "dbUserName": "TEST",
```

```
    "endTime": null,
    "errorMessage": null,
    "exitCode": 0,
    "logTime": "2021-01-15 00:25:18.850375",
    "netProtocol": "beq",
    "objectName": "PERSONS",
    "objectType": "TEST",
    "paramList": [],
    "pid": 17687,
    "remoteHost": "123.456.789.0",
    "remotePort": null,
    "rowCount": null,
    "serverHost": "987.654.321.09",
    "serverType": "oracle",
    "serverVersion": "19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3",
    "serviceName": "oracle-ee",
    "sessionId": 1080639707,
    "startTime": null,
    "statementId": 44,
    "substatementId": null,
    "transactionId": null,
    "engineNativeAuditFields": {
        "UNIFIED_AUDIT_POLICIES": "TEST_POL_EVERYTHING",
        "FGA_POLICY_NAME": null,
        "DV_OBJECT_STATUS": null,
        "SYSTEM_PRIVILEGE_USED": null,
        "OLS_LABEL_COMPONENT_TYPE": null,
        "XS_SESSIONID": null,
        "ADDITIONAL_INFO": null,
        "INSTANCE_ID": 1,
        "DV_COMMENT": null,
        "RMAN_SESSION_STAMP": null,
        "NEW_NAME": null,
        "DV_ACTION_NAME": null,
        "OLS_PROGRAM_UNIT_NAME": null,
        "OLS_STRING_LABEL": null,
        "RMAN_SESSION_RECID": null,
        "OBJECT_PRIVILEGES": null,
        "OLS_OLD_VALUE": null,
        "XS_TARGET_PRINCIPAL_NAME": null,
        "XS_NS_ATTRIBUTE": null,
        "XS_NS_NAME": null,
        "DBLINK_INFO": null,
```

```
        "AUTHENTICATION_TYPE": "(TYPE\u003d(DATABASE));(CLIENT ADDRESS\u003d((PROTOCOL
\u003dbeq)(HOST\u003d123.456.789.0)));",
        "OBJECT_EDITION": null,
        "OLS_PRIVILEGES_GRANTED": null,
        "EXCLUDED_USER": null,
        "DV_ACTION_OBJECT_NAME": null,
        "OLS_LABEL_COMPONENT_NAME": null,
        "EXCLUDED_SCHEMA": null,
        "DP_TEXT_PARAMETERS1": null,
        "XS_USER_NAME": null,
        "XS_ENABLED_ROLE": null,
        "XS_NS_ATTRIBUTE_NEW_VAL": null,
        "DIRECT_PATH_NUM_COLUMNS_LOADED": null,
        "AUDIT_OPTION": null,
        "DV_EXTENDED_ACTION_CODE": null,
        "XS_PACKAGE_NAME": null,
        "OLS_NEW_VALUE": null,
        "DV_RETURN_CODE": null,
        "XS_CALLBACK_EVENT_TYPE": null,
        "USERHOST": "ip-12-34-5-678",
        "GLOBAL_USERID": null,
        "CLIENT_IDENTIFIER": null,
        "RMAN_OPERATION": null,
        "TERMINAL": "pts/1",
        "OS_USERNAME": "rdsdb",
        "OLS_MAX_READ_LABEL": null,
        "XS_PROXY_USER_NAME": null,
        "XS_DATASEC_POLICY_NAME": null,
        "DV_FACTOR_CONTEXT": null,
        "OLS_MAX_WRITE_LABEL": null,
        "OLS_PARENT_GROUP_NAME": null,
        "EXCLUDED_OBJECT": null,
        "DV_RULE_SET_NAME": null,
        "EXTERNAL_USERID": null,
        "EXECUTION_ID": null,
        "ROLE": null,
        "PROXY_SESSIONID": 0,
        "DP_BOOLEAN_PARAMETERS1": null,
        "OLS_POLICY_NAME": null,
        "OLS_GRANTEE": null,
        "OLS_MIN_WRITE_LABEL": null,
        "APPLICATION_CONTEXTS": null,
        "XS_SCHEMA_NAME": null,
        "DV_GRANTEE": null,
```

```
        "XS_COOKIE": null,
        "DBPROXY_USERNAME": null,
        "DV_ACTION_CODE": null,
        "OLS_PRIVILEGES_USED": null,
        "RMAN_DEVICE_TYPE": null,
        "XS_NS_ATTRIBUTE_OLD_VAL": null,
        "TARGET_USER": null,
        "XS_ENTITY_TYPE": null,
        "ENTRY_ID": 13,
        "XS_PROCEDURE_NAME": null,
        "XS_INACTIVITY_TIMEOUT": null,
        "RMAN_OBJECT_TYPE": null,
        "SYSTEM_PRIVILEGE": null,
        "NEW_SCHEMA": null,
        "SCN": 5136972
    }
}
```

次の例は、SQL Server DB の SELECT イベントを示しています。

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
```

```
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x010500000000000515000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

DatabaseActivityMonitoringRecords JSON オブジェクト

データベースアクティビティイベントレコードは、次の情報を含む JSON オブジェクトにあります。

JSON フィールド	データ型	説明
`type`	文字列	JSON レコードのタイプ。値は `DatabaseActivityMonitoringRecords` です。
`version`	文字列	データベースアクティビティモニタリングレコードのバージョン。Oracle DB ではバージョン 1.3、SQL Server ではバージョン 1.4 を使用します。これらのエンジンバージョンでは `engineNativeAuditFields` JSON オブジェクトが導入されています。
databaseActivityEvents	string	アクティビティイベントを含む JSON オブジェクト。
key	string	databaseActivityEventList JSON 配列 の復号に使用する暗号化キー。

databaseActivityEvents JSON オブジェクト

`databaseActivityEvents` JSON オブジェクトには、次の情報が含まれています。

JSON レコードの最上位フィールド

監査ログの各イベントは、JSON 形式のレコード内にラップされます。このレコードには、次のフィールドが含まれます。

type

このフィールドは常に値 `DatabaseActivityMonitoringRecords` を持ちます。

バージョン

このフィールドは、データベースアクティビティストリーミングデータプロトコルまたはコントラクトのバージョンを表します。これは、使用可能なフィールドを定義します。

databaseActivityEvents

1 つ以上のアクティビティイベントを表す暗号化された文字列。これは、base64 バイト配列として表されます。文字列を復号すると、このセクションの例に示すフィールドを持つ JSON 形式のレコードが生成されます。

key

`databaseActivityEvents` 文字列の暗号化に使用される暗号化されたデータキー。これは、データベースアクティビティストリーミングをスタートしたときに指定した AWS KMS key と同じです。

以下の例は、このレコードの形式を示しています。

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.3",
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

```
          "type":"DatabaseActivityMonitoringRecords",
          "version":"1.4",
          "databaseActivityEvents":"encrypted audit records",
          "key":"encrypted key"
```

`databaseActivityEvents` フィールドの内容を復号化するには、次のステップを実行します。

1. データベースアクティビティストリーミングをスタートするときに指定した KMS キーを使用して、key JSON フィールドの値を復号します。これにより、データ暗号化キーがクリアテキストで返されます。
2. `databaseActivityEvents` JSON フィールドの値を Base64 デコードして、監査ペイロードの暗号化テキストをバイナリ形式で取得します。
3. 初期のステップでデコードしたデータ暗号化キーを使用して、バイナリ暗号文を復号化します。
4. 復号化されたペイロードを解凍します。
   - 暗号化されたペイロードは、`databaseActivityEvents` フィールドにあります。

- databaseActivityEventList フィールドには、監査レコードの配列が含まれます。配列内の type フィールドには、 record または heartbeat を使用できます。

監査ログのアクティビティイベントレコードは、次の情報を含む JSON オブジェクトです。

JSON フィールド	データ型	説明
type	文字列	JSON レコードのタイプ。値は DatabaseActivityMonitoringRecord です。
instanceId	文字列	DB インスタンスのリソース識別子。DB インスタンス属性 DbiResourceId に対応します。
databaseActivityEventList JSON 配列	文字列	アクティビティ監査レコードまたはハートビートメッセージの配列。

データベースアクティビティストリームの databaseActivityEventList JSON 配列

監査ログのペイロードは、暗号化された databaseActivityEventList JSON 配列です。次の表に、監査ログの復号された DatabaseActivityEventList 配列内の各アクティビティイベントのフィールドをアルファベット順に示します。

Oracle Database で統合監査が有効になっている場合、この新しい監査証跡に監査レコードが配置されます。UNIFIED_AUDIT_TRAIL ビューでは、監査証跡から監査レコードを取得することにより、監査レコードを表形式で表示します。データベースアクティビティストリーミングをスタートすると、UNIFIED_AUDIT_TRAIL 内の列が databaseActivityEventList 配列のフィールドにマッピングされます。

> **Important**
>
> イベントの構造は変わる場合があります。Amazon RDS では、将来、アクティビティイベントに新しいフィールドが追加される可能性があります。JSON データを分析するアプリケーションでは、コードが未知のフィールド名に対して無視または適切なアクションを実行できることを確認します。

Amazon RDS for Oracle の databaseActivityEventList フィールド

Amazon RDS for Oracle の `databaseActivityEventList` フィールドは以下のとおりです。

フィールド	データ型	送信元	説明
`class`	文字列	`UNIFIED_AUDIT_TRAIL` の `AUDIT_TYPE` 列	アクティビティイベントのクラス。これは `UNIFIED_AUDIT_TRAIL` ビューの `AUDIT_TYPE` 列に対応します。Amazon RDS for Oracle の有効な値は以下のとおりです。 • `Standard` • `FineGrainedAudit` • `XS` • `Database Vault` • `Label Security` • `RMAN_AUDIT` • `Datapump` • `Direct path API` 詳細については、Oracle ドキュメントの「UNIFIED_AUDIT_TRAIL」を参照してください。
`clientApplication`	文字列	`CLIENT_PROGRAM_NAME` (内)`UNIFIED_AUDIT_TRAIL`	クライアントのレポートどおりにクライアントが接続に使用していたアプリケーション。クライア

フィールド	データ型	送信元	説明
			ントはこの情報を指定する必要はないため、値は null でも問題ありません。サンプル値はJDBC Thin Client。
command	文字列	UNIFIED_AUDIT_TRAIL の ACTION_NAME 列	ユーザーによって実行されるアクションの名前。完全なアクションを把握するには、コマンド名と AUDIT_TYPE 値の両方を読み取ります。サンプル値はALTER DATABASE。
commandText	文字列	UNIFIED_AUDIT_TRAIL の SQL_TEXT 列	イベントに関連付けられる SQL ステートメント。サンプル値はALTER DATABASE BEGIN BACKUP。
databaseName	文字列	V$DATABASE の NAME 列	データベースの名前。
dbid	numb	UNIFIED_AUDIT_TRAIL の DBID 列	データベースの数値識別子。サンプル値は1559204751 。
dbProtocol	文字列	該当なし	データベースプロトコル。このベータ版では、値は oracle です。

フィールド	データ型	送信元	説明
dbUserName	文字列	UNIFIED_AUDIT_TRAIL の DBUSERNAME 列	アクションが監査されたデータベースユーザーの名前。サンプル値はRDSADMIN。
endTime	文字列	該当なし	このフィールドは RDS for Oracle には使用されず、常に null です。

フィールド	データ型	送信元	説明
engineNativeAuditFields	オブジェクト	UNIFIED_AUDIT_TRAIL	デフォルトでは、このオブジェクトは空です。--engine-native-audit-fields-included オプションでアクティビティストリーミングをスタートすると、このオブジェクトには以下の列とその値が含まれます。 ADDITIONAL_INFO APPLICATION _CONTEXTS AUDIT_OPTION AUTHENTICATIO N_TYPE CLIENT_IDENTIFIER CURRENT_USER DBLINK_INFO DBPROXY_USERNAME DIRECT_PATH_NU M_COLUMNS_LOADED DP_BOOLEAN _PARAMETERS1 DP_TEXT_PARAME TERS1 DV_ACTION_CODE DV_ACTION_NAME DV_ACTION_OBJECT_N AME DV_COMMENT DV_EXTENDED_ ACTION_CODE DV_FACTOR_CONTEXT DV_GRANTEE DV_OBJECT_STATUS

フィールド	データ型	送信元	説明
			DV_RETURN_CODE DV_RULE_SET_NAME ENTRY_ID EXCLUDED_OBJECT EXCLUDED_SCHEMA EXCLUDED_USER EXECUTION_ID EXTERNAL_USERID FGA_POLICY_NAME GLOBAL_USERID INSTANCE_ID KSACL_SER VICE_NAME KSACL_SOURCE_LOCA TION KSACL_USER_NAME NEW_NAME NEW_SCHEMA OBJECT_EDITION OBJECT_PRIVILEGES OLS_GRANTEE OLS_LABEL_COM PONENT_NAME OLS_LABEL_COMPO NENT_TYPE OLS_MAX_READ_LABEL OLS_MAX_WRITE_LA BEL OLS_MIN_WRITE_ LABEL OLS_NEW_VALUE OLS_OLD_VALUE OLS_PARENT_ GROUP_NAME OLS_POLICY_NAME OLS_PRIVILEGES_GRA NTED OLS_PRIVILEGE S_USED

フィールド	データ型	送信元	説明
			OLS_PROGRAM_UNIT_NAME OLS_STRING_LABEL OS_USERNAME PROTOCOL_ACTION_NAME PROTOCOL_MESSAGE PROTOCOL_RETURN_CODE PROTOCOL_SESSION_ID PROTOCOL_USERHOST PROXY_SESSIONID RLS_INFO RMAN_DEVICE_TYPE RMAN_OBJECT_TYPE RMAN_OPERATION RMAN_SESSION_RECID RMAN_SESSION_STAMP ROLE SCN SYSTEM_PRIVILEGE SYSTEM_PRIVILEGE_USED TARGET_USER TERMINAL UNIFIED_AUDIT_POLICIES USERHOST XS_CALLBACK_EVENT_TYPE XS_COOKIE XS_DATASEC_POLICY_NAME XS_ENABLED_ROLE XS_ENTITY_TYPE XS_INACTIVITY_TIMEOUT XS_NS_ATTRIBUTE

フィールド	データ型	送信元	説明
			XS_NS_ATTRIBUTE_NEW_VAL XS_NS_ATTRIBUTE_OLD_VAL XS_NS_NAME XS_PACKAGE_NAME XS_PROCEDURE_NAME XS_PROXY_USER_NAME XS_SCHEMA_NAME XS_SESSIONID XS_TARGET_PRINCIPAL_NAME XS_USER_NAME 詳細については、Oracle Database ドキュメントの「UNIFIED_AUDIT_TRAIL」を参照してください。
errorMessage	文字列	該当なし	このフィールドは RDS for Oracle には使用されず、常に null です。
exitCode	numb	UNIFIED_AUDIT_TRAIL の RETURN_CODE 列	アクションによって生成された Oracle データベースのエラーコード。アクションが成功した場合、値は 0 です。
logTime	文字列	UNIFIED_AUDIT_TRAIL の EVENT_TIMESTAMP_UTC 列	監査証跡エントリの作成のタイムスタンプ。サンプル値は2020-11-27 06:56:14.981404 。

フィールド	データ型	送信元	説明
netProtocol	文字列	UNIFIED_AUDIT_TRAIL の AUTHENTICATION_TYPE 列	ネットワーク通信プロトコル。サンプル値はTCP。
objectName	文字列	UNIFIED_AUDIT_TRAIL の OBJECT_NAME 列	アクションによって影響を及ぼされるオブジェクトの名前。サンプル値はemployees 。
objectType	文字列	UNIFIED_AUDIT_TRAIL の OBJECT_SCHEMA 列	アクションによって影響を及ぼされるオブジェクトのスキーマ名。サンプル値はhr。
paramList	リスト	UNIFIED_AUDIT_TRAIL の SQL_BINDS 列	SQL_TEXT に関連付けられているバインド可変のリスト (存在する場合)。サンプル値はparameter_1,parameter_2 。
pid	numb	UNIFIED_AUDIT_TRAIL の OS_PROCESS 列	Oracle データベースプロセスのオペレーティングシステムプロセス識別子。サンプル値は22396。
remoteHost	文字列	UNIFIED_AUDIT_TRAIL の AUTHENTICATION_TYPE 列	セッションを生成したホストのクライアント IP アドレスまたは名前です。サンプル値は123.456.789.123 。

フィールド	データ型	送信元	説明
remotePort	文字列	UNIFIED_AUDIT_TRAIL の AUTHENTICATION_TYPE 列	クライアントのポート番号。Oracle Database 環境における一般的な値は 1521 です。
rowCount	numb	該当なし	このフィールドは RDS for Oracle には使用されず、常に null です。
serverHost	文字列	データーベースホスト:	データベースサーバーのホスト IP アドレス。サンプル値は123.456.789.123 。
serverType	文字列	該当なし	データベースサーバーのタイプ。値は常に ORACLE です。
serverVersion	文字列	データーベースホスト:	Amazon RDS for Oracle のバージョン、リリースアップデート (RU)、およびリリースアップデートリビジョン (RUR)。サンプル値は19.0.0.0.ru-2020-01.rur-2020-01.r1.EE.3 。
serviceName	文字列	データーベースホスト:	サービスの名前。サンプル値はoracle-ee 。

フィールド	データ型	送信元	説明
sessionId	numb	UNIFIED_AUDIT_TRAIL の SESSIONID 列	監査のセッション識別子。例:「1894327130 」。
startTime	文字列	該当なし	このフィールドは RDS for Oracle には使用されず、常に null です。
statementId	numb	UNIFIED_AUDIT_TRAIL の STATEMENT_ID 列	実行する各ステートメントの数値 ID。1 つのステートメントで多くのアクションが起こる可能性があります。サンプル値は142197。
substatementId	該当なし	該当なし	このフィールドは RDS for Oracle には使用されず、常に null です。
transactionId	文字列	UNIFIED_AUDIT_TRAIL の TRANSACTION_ID 列	オブジェクトが変更されるトランザクションの識別子。サンプル値は02000800D5030000 。

Amazon RDS for SQL Server の databaseActivityEventList フィールド

Amazon RDS for SQL Server の databaseActivityEventList フィールドは次のとおりです。

フィールド	データ型	送信元	説明
class	string	sys.fn_get_audit_file.class_type のマッピング先 sys.dm_audit_class_type_map.class_type_desc	アクティビティイベントのクラス。詳細については、Microsoft ドキュメントの「SQL Server 監査 (データベースエンジン)」を参照してください。
clientApplication	文字列	sys.fn_get_audit_file.application_name	クライアントから報告されたとおりにクライアントが接続するアプリケーション (SQL Server バージョン 14 以降)。SQL Server バージョン 13 では、このフィールドは NULL になっています。
command	文字列	sys.fn_get_audit_file.action_id のマッピング先 sys.dm_audit_actions.name	SQL ステートメントの一般的なカテゴリ。このフィールドの値はクラスの値によって異なります。
commandText	文字列	sys.fn_get_audit_file.statement	このフィールドは SQL ステートメントを示します。
databaseName	文字列	sys.fn_get_audit_file.database_name	データベースの名前
dbProtocol	文字列	該当なし	データベースプロトコル。値は SQLSERVER です。
dbUserName	文字列	sys.fn_get_audit_file.server_principal_name	クライアント認証のためのデータベースユーザー。
endTime	文字列	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。

フィールド	データ型	送信元	説明
engineNativeAuditFields	オブジェクト	この列にリストされていない sys.fn_get_audit_file の各フィールド。	デフォルトでは、このオブジェクトは空です。--engine-native-audit-fields-included オプションでアクティビティストリームを開始すると、このオブジェクトには他のネイティブエンジン監査フィールドが含まれ、この JSON マップでは返されません。
errorMessage	文字列	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。
exitCode	integer	sys.fn_get_audit_file.succeeded	イベントを開始したアクションが成功したかどうかを示します。このフィールドを null にすることはできません。ログインイベントを除くすべてのイベントについて、このフィールドはアクセス許可チェックの成否を報告しますが、操作の成否は報告しません。 値には以下のものが含まれます。 • 0 - 失敗 • 1 - 成功
logTime	文字列	sys.fn_get_audit_file.event_time	SQL Server によって記録されるイベントのタイムスタンプ。
netProtocol	文字列	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。

フィールド	データ型	送信元	説明
objectName	文字列	sys.fn_get_audit_file.object_name	SQL ステートメントがオブジェクトに対して使用されている場合のデータベースオブジェクトの名前。
objectType	文字列	sys.fn_get_audit_file.class_type のマッピング先 sys.dm_audit_class_type_map.class_type_desc	SQL ステートメントがオブジェクトタイプに対して使用されている場合のデータベースオブジェクトタイプ。
paramList	文字列	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。
pid	integer	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。
remoteHost	文字列	sys.fn_get_audit_file.client_ip	SQL ステートメントを発行したクライアントの IP アドレスまたはホスト名 (SQL Server バージョン 14 以降)。SQL Server バージョン 13 では、このフィールドは NULL になっています。
remotePort	integer	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。
rowCount	integer	sys.fn_get_audit_file.affected_rows	SQL ステートメントによって影響を受けるテーブルの行数 (SQL Server バージョン 14 以降)。SQL Server バージョン 13 では、このフィールドは null になっています。

フィールド	データ型	送信元	説明
serverHost	文字列	データーベースホスト	ホストデータベースサーバーの IP アドレス。
serverType	文字列	該当なし	データベースサーバーのタイプ。値は SQLSERVER です。
serverVersion	文字列	データーベースホスト	データベースサーバーのバージョン。SQL Server 2017 の場合は 15.00.4073.23.v1.R1 などです。
serviceName	文字列	データーベースホスト	サービスの名前。値の例は sqlserver-ee です。
sessionId	integer	sys.fn_get_audit_file.session_id	セッションの一意識別子。
startTime	文字列	該当なし	このフィールドは Amazon RDS for SQL Server では使用されず、値は null です。
statementId	文字列	sys.fn_get_audit_file.sequence_group_id	クライアントの SQL ステートメントの一意識別子。識別子は、生成されるイベントごとに異なります。サンプル値は0x38eaf4156267184094bb82071aaab644 。
substatementId	integer	sys.fn_get_audit_file.sequence_number	ステートメントのシーケンス番号を決定する識別子。この識別子は、大きなレコードが複数のレコードに分割されている場合に役立ちます。

フィールド	データ型	送信元	説明
transactionId	integer	sys.fn_get_audit_file.transaction_id	トランザクションの識別子。アクティブなトランザクションがない場合、値は 0 です。
type	文字列	データベースアクティビティストリームが生成されました	イベントのタイプ。値は record または heartbeat です。

AWS SDK を使用したデータベースアクティビティストリーミングの処理

アクティビティストリーミングをプログラムで処理するには、AWS SDK を使用します。以下は、インスタンスベースの有効化に Database Activity Streams レコードを使用する、完全に機能する Java および Python の例です。

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
```

```
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {

    private static final String STREAM_NAME = "aws-rds-das-[instance-external-
resource-id]"; // aws-rds-das-db-ABCD123456
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String RESOURCE_ID = "[external-resource-id]"; // db-
ABCD123456
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
```

```
   private static final BasicAWSCredentials CREDENTIALS = new
BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
   private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
AWSStaticCredentialsProvider(CREDENTIALS);

   private static final AwsCrypto CRYPTO = new AwsCrypto();
   private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
           .withRegion(REGION_NAME)
           .withCredentials(CREDENTIALS_PROVIDER).build();

   class Activity {
       String type;
       String version;
       String databaseActivityEvents;
       String key;
   }

   class ActivityEvent {
       @SerializedName("class") String _class;
       String clientApplication;
       String command;
       String commandText;
       String databaseName;
       String dbProtocol;
       String dbUserName;
       String endTime;
       String errorMessage;
       String exitCode;
       String logTime;
       String netProtocol;
       String objectName;
       String objectType;
       List<String> paramList;
       String pid;
       String remoteHost;
       String remotePort;
       String rowCount;
       String serverHost;
       String serverType;
       String serverVersion;
       String serviceName;
       String sessionId;
       String startTime;
       String statementId;
```

```
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId; // note that clusterId will contain an empty string on RDS
Oracle and RDS SQL Server
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {

        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }
```

```
        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }

        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);

                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:db-id", RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));
```

```
                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
```

```
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
CREDENTIALS_PROVIDER, workerId);

kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
```

```
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
 APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # db-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-db-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL
```

```
class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                          wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)


def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],
```

```
 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:db-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

データベースアクティビティストリームの IAM ポリシーの例

データベースアクティビティストリーミングに対する適切な AWS Identity and Access Management (IAM) ロールの権限が付与されているユーザーは、DB インスタンスのアクティビティストリーミング設定の作成、スタート、停止、および変更ができます。これらのアクションはストリーミングの監査ログに含まれています。コンプライアンスのベストプラクティスとして、これらの権限は DBA に付与しないことをお勧めします。

データベースアクティビティストリーミングへのアクセスを設定するには、IAM ポリシーを使用します。Amazon RDS 認証の詳細については、「Amazon RDS での Identity and Access Management」を参照してください。IAM ポリシーの作成の詳細については、「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。

Exampleデータベースアクティビティストリーミングの設定を許可するポリシー

アクティビティストリーミングを変更するためのきめ細かいアクセスをユーザーに付与するには、IAM ポリシーでサービス固有のオペレーションコンテキストキー (`rds:StartActivityStream` および `rds:StopActivityStream`) を使用します。次の IAM ポリシーの例では、ユーザーまたはロールがアクティビティストリーミングを設定することを許可します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureActivityStreams",
            "Effect": "Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource": "*"
        }
    ]
}
```

Exampleデータベースアクティビティストリーミングのスタートを許可するポリシー

次の IAM ポリシーの例では、ユーザーまたはロールがアクティビティストリーミングをスタートすることを許可します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
```

```
        }
    ]
}
```

Exampleデータベースアクティビティストリーミングの停止を許可するポリシー

次の IAM ポリシーの例では、ユーザーまたはロールがアクティビティストリーミングを停止することを許可します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Exampleデータベースアクティビティストリーミングのスタートを拒否するポリシー

次の IAM ポリシーの例では、ユーザーまたはロールによるアクティビティストリーミングのスタートが阻止されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
```

```
    ]
}
```

Exampleデータベースアクティビティストリーミング停止を拒否するポリシー

次の IAM ポリシーの例では、ユーザーまたはロールによるアクティビティストリーミングの停止が阻止されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Amazon GuardDuty RDS Protection による脅威のモニタリング

Amazon GuardDuty は、AWS 環境内のアカウント、コンテナ、ワークロード、データを保護する脅威検知サービスです。GuardDuty は、機械学習 (ML) モデル、異常および脅威検出機能を使用して、さまざまなログソースとランタイムアクティビティを継続的に監視し、環境内の潜在的なセキュリティリスクと悪意のあるアクティビティを特定して優先順位を付けます。

GuardDuty RDS Protection は、Amazon RDS データベースに対する潜在的なアクセス脅威がないか、ログインイベントを分析してプロファイリングします。RDS Protection を有効にすると、GuardDuty は RDS データベースの RDS ログインイベントを消費します。RDS Protection は、これらのイベントを監視し、潜在的な内部脅威や外部アクターがないかプロファイリングします。

GuardDuty RDS Protection の有効化の詳細については、「Amazon GuardDuty ユーザーガイド」の「GuardDuty RDS Protection」を参照してください。

RDS Protection が潜在的な脅威 (一連の成功、失敗、未完了のログイン試行で異常なパターンが発生するなど) を検出すると、GuardDuty は侵害された可能性のあるデータベースに関する詳細を含む新しい検出結果を生成します。Amazon GuardDuty コンソールでは、検出結果の概要セクションで検出結果の詳細を見ることができます。検出結果の詳細は検出結果のタイプによって異なります。リソースタイプとリソースロールという基本的な情報によって、検出結果にどのような情報が表示されるか決まります。検索結果の一般的に入手可能な詳細と検出結果タイプの詳細については、「Amazon GuardDuty ユーザーガイド」の「検出結果の詳細」と「GuardDuty RDS Protection の検索結果タイプ」を参照してください。

RDS Protection 機能は、この機能が使用可能な任意の AWS リージョン の任意の AWS アカウントについて、有効または無効にできます。RDS Protection が有効になっていない場合、GuardDuty は侵害された可能性のある RDS データベースを検出したり、侵害の詳細を提供したりしません。

既存の GuardDuty アカウントでは、RDS Protection を 30 日間お試しいただけます。新しい GuardDuty アカウントでは、RDS Protection が既に有効になっており、30 日間の無料トライアル期間に含まれています。詳細については、「Amazon GuardDuty ユーザーガイド」の「GuardDuty コストの見積もり」を参照してください。

GuardDuty がまだ RDS Protection をサポートしていない AWS リージョンについては、「Amazon GuardDuty ユーザーガイド」の「リージョン固有機能の可用性」を参照してください。

GuardDuty RDS Protection がサポートする RDS データベースのバージョンについては、「Amazon Amazon GuardDutyユーザーガイド」の「Supported Amazon Aurora, Amazon RDS, and Aurora Limitless databases」を参照してください。

Amazon RDS Custom

Amazon RDS Custom でデータベース管理タスクとオペレーションが自動化されます。RDS Custom で、データベース管理者としてデータベース環境とオペレーティングシステムへのアクセスおよびカスタマイズが可能になります。RDS Custom を使用すると、レガシー、カスタム、パッケージのアプリケーション要件を満たすカスタマイズが可能です。

RDS Custom に関する最新のオンラインセミナーやブログについては、「Amazon RDS Custom のリソース」を参照してください。

トピック

- データベースカスタマイズの課題への対応
- Amazon RDS Custom の管理モデルと利点
- Amazon RDS Custom アーキテクチャ
- Amazon RDS Custom のセキュリティ
- RDS Custom for Oracle を使用する
- RDS Custom for SQL Server の使用

データベースカスタマイズの課題への対応

Amazon RDS Custom は、サードパーティーアプリケーションでのカスタマイズが必要なため、フルマネージドサービスに簡単に移行できない市場に Amazon RDS のメリットを提供します。Amazon RDS Custom は、管理時間を節約し、耐久性の高いがあり、ビジネスに合わせて拡張できます。

データベースとオペレーティングシステム全体をAWSでフルマネージする必要がある場合は、Amazon RDS をお勧めします。依存型アプリケーションを利用するために、データベースおよび基盤となるオペレーティングシステムの管理者権限が必要な場合は、Amazon RDS Custom が適しています。完全な管理責任が必要で、マネージドコンピューティングサービスが必要な場合は、Amazon EC2での商用データベースの自己管理が最良の選択です。

マネージドサービスエクスペリエンスを提供するために、Amazon RDS では基盤となるホストにアクセスできません。また、Amazon RDS では、高度な特権を必要とする一部のシステムプロシージャやオブジェクトへのアクセスが制限されます。ただし、一部のアプリケーションでは、オペレーティングシステム (OS) の特権ユーザーとしてオペレーションを実行しなければならないことがあります。

例えば、以下を必要とする場合があります。

- カスタムデータベースと OS のパッチおよびパッケージをインストールします。
- 特定のデータベース設定を構成します。
- ファイルシステムを構成し、そのアプリケーションで直接ファイルを共有できるようにします。

以前は、アプリケーションのカスタマイズが必要な場合は、データベースをオンプレミスまたは Amazon EC2 にデプロイする必要がありました。この場合、次の表のように、データベース管理に関する責任の大部分またはすべてを負うことになります。

特徴	オンプレミスの責任	Amazon EC2 の責任	Amazon RDS の責任
アプリケーションの最適化	カスタマー	カスタマー	カスタマー
スケーリング	カスタマー	カスタマー	AWS
高可用性	カスタマー	カスタマー	AWS
データベースバックアップ	カスタマー	カスタマー	AWS
データベースソフトウェアのパッチ適用	カスタマー	カスタマー	AWS
データベースソフトウェアのインストール	カスタマー	カスタマー	AWS
OS のパッチ適用	カスタマー	カスタマー	AWS
OS インストール	カスタマー	カスタマー	AWS
サーバーのメンテナンス	カスタマー	AWS	AWS
ハードウェアライフサイクル	カスタマー	AWS	AWS

特徴	オンプレミスの責任	Amazon EC2 の責任	Amazon RDS の責任
電力、ネットワーク、冷却	カスタマー	AWS	AWS

データベースソフトウェアを自分で管理する場合、制御はしやすくなりますが、ユーザーエラーが発生しやすくなります。例えば、マニュアルで変更すると、誤ってアプリケーションのダウンタイムを引き起こすこと可能性があります。問題を特定して修正するため、すべての変更をチェックするのに何時間も費やす可能性があります。理想的には、一般的な DBA タスクを自動化し、さらにデータベースおよび基盤となるオペレーティングシステムへの特権アクセスもサポートするマネージドデータベースサービスが必要です。

Amazon RDS Custom の管理モデルと利点

Amazon RDS Custom は、基盤となる OS とデータベース環境へのアクセスを必要とするレガシー、カスタム、およびパッケージアプリケーション向けのマネージドデータベースサービスです。RDS Custom では、データベースのセットアップ、運用、スケーリングを自動化し、AWS クラウドデータベースおよび基盤となるオペレーティングシステムへのアクセスを許可します。このアクセス権により、依存型アプリケーションの要件を満たすために、設定の構成、パッチのインストール、ネイティブ機能の有効化を行うことができます。RDS Custom では、AWS マネジメントコンソールまたはAWS CLIを使用してデータベースワークロードを実行できます。

RDS Custom は Oracle データベースエンジンと Microsoft SQL Server エンジンのみをサポートしています。

トピック

- RDS Custom の責任分担モデル
- RDS Custom で境界と未サポートの構成をサポート
- RDS カスタムの主な利点

RDS Custom の責任分担モデル

RDS Custom では、Amazon RDS の管理機能を使用しますが、Amazon EC2 の場合と同様にホストを管理し、OS をカスタマイズします。Amazon RDS における業務以外のデータベース管理の責任を負うことになります。その結果、Amazon RDS よりもデータベースと DB インスタンスの管理をより細かく制御できると同時に、RDS 自動化のメリットを享受できます。

責任共有とは、以下のことを意味します。

1. RDS Custom 機能を使用するときに、プロセスの一部を所有します。

   例えば RDS Custom for Oracle では、どの Oracle データベースパッチを使用するか、それらを DB インスタンスに適用するタイミングを制御できます。
2. RDS カスタム機能のカスタマイズがすべて正常に機能することを確認するのは、お客様の責任です。

   無効なカスタマイズからの保護を目的に、RDS Custom は DB インスタンス外部で実行される自動化ソフトウェアを備えています。基盤となる Amazon EC2 インスタンスに障害が発生した場合、RDS Custom は、EC2 インスタンスを再起動または置換することによってこれらの問題を解決します。ユーザーに表示される唯一の変更は、新しい IP アドレスです。詳細については、「Amazon RDS Custom ホストの置換」を参照してください。

次の表では、RDS Custom の各種機能の責任共有モデルの詳細を示します。

機能	Amazon EC2 の責任	Amazon RDS の責任	RDS Custom for Oracle の責任	RDS Custom for SQL Server の責任
アプリケーションの最適化	カスタマー	カスタマー	カスタマー	カスタマー
スケーリング	カスタマー	AWS	Shared	Shared
高可用性	カスタマー	AWS	AWS	AWS
データベースバックアップ	カスタマー	AWS	Shared	AWS
データベースソフトウェアのパッチ適用	カスタマー	AWS	Shared	RPEV の場合は AWS、CEV の場合はカスタマー[1]
データベースソフトウェアのインストール	カスタマー	AWS	Shared	RPEV の場合は AWS、CEV の

機能	Amazon EC2 の責任	Amazon RDS の責任	RDS Custom for Oracle の責任	RDS Custom for SQL Server の責任
				場合はカスタマー[1]
OS のパッチ適用	カスタマー	AWS	お客様	RPEV の場合は AWS、CEV の場合はカスタマー[1]
OS インストール	カスタマー	AWS	Shared	AWS
サーバーのメンテナンス	AWS	AWS	AWS	AWS
ハードウェアライフサイクル	AWS	AWS	AWS	AWS
電力、ネットワーク、および冷却	AWS	AWS	AWS	AWS

[1] カスタムエンジンバージョン (CEV) は、データベースバージョンと Amazon マシンイメージ (AMI) のバイナリボリュームスナップショットです。RDS が提供するエンジンバージョン (RPEV) は、デフォルトの Amazon マシンイメージ (AMI) と Microsoft SQL Server のインストールです。

Microsoft SQL Server を使用して RDS Custom DB インスタンスを作成できます。この場合は以下のようになります。

- ライセンス込み (LI) と Bring Your Own Media (BYOM) の 2 つのライセンスモデルから選択できます。
- LI では、SQL Server のライセンスを別途購入する必要はありません。AWS は、SQL Server データベースソフトウェアのライセンスを保持しています。
- BYOM では、独自の Microsoft SQL Server バイナリとライセンスを容易してインストールします。

Oracle データベースを使用して RDS Custom DB インスタンスを作成できます。この場合、次の操作を行います。

- 自分のメディアを管理します。

  RDS Custom を使用するときは、独自のデータベースインストールファイルとパッチをアップロードします。これらのファイルからカスタムエンジンバージョン (CEV) を作成します。次に、この CEV を使用して RDS Custom DB インスタンスを作成できます。
- 独自のライセンスを管理します。

  Oracle データベース独自のライセンスを持ち込み、ライセンス管理を自分で行います。

RDS Custom で境界と未サポートの構成をサポート

RDS Custom は、サポートペリメーターと呼ばれるモニタリング機能を提供しています。この機能により、ホストとデータベース環境が正しく設定されます。DB インスタンスがサポート範囲外になるような変更を加えた場合、設定の問題を手動で修正するまで、RDS Custom はインスタンスのステータスを `unsupported-configuration` に変更します。詳細については、「RDS Custom サポート範囲」を参照してください。

RDS カスタムの主な利点

RDS Customを使用すると、次のことが可能です。

- 以下を含む Amazon RDS と同様の管理タスクの多くを自動化します。
  - データベースのライフサイクル管理
  - 自動バックアップとポイントインタイムリカバリ (PITR) の自動化
  - RDS Custom DB インスタンスの状態をモニタリングし、インフラストラクチャ、オペレーティングシステム、およびデータベースプロセスの変化を観察します。
  - DB インスタンスの障害に対して、問題解決のための通知またはアクションの実行
- サードパーティー製アプリケーションをインストールします。

  ソフトウェアをインストールして、カスタムアプリケーションとエージェントを実行できます。ホストへの特権アクセス権があるため、レガシーアプリケーションをサポートするためにファイルシステムを変更できます。
- カスタムパッチをインストールします。

RDS Custom DB インスタンスで、カスタムデータベースパッチを適用したり OS パッケージを変更したりできます。

- フルマネージドサービスへの移行前に、オンプレミスのデータベースをステージングします。

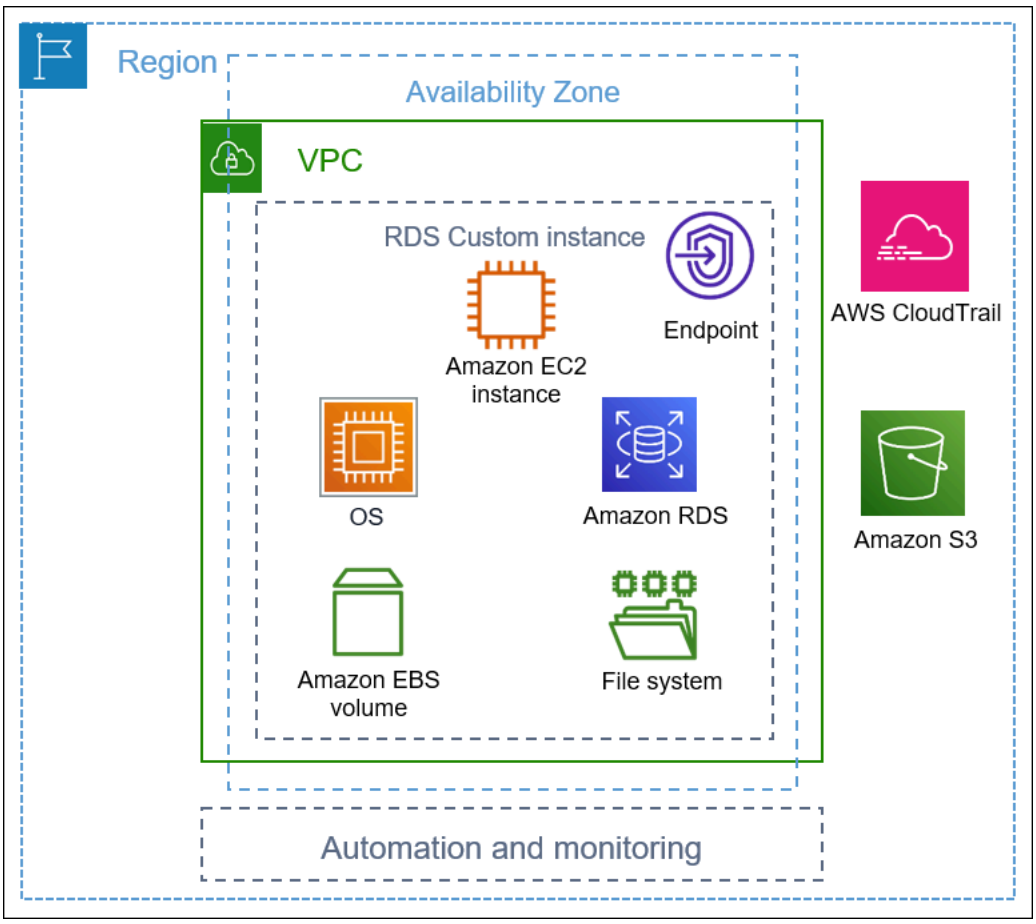
  独自のオンプレミスデータベースを管理している場合、データベースをそのまま RDS Custom にステージングできます。クラウド環境に慣れたら、データベースをフルマネージド Amazon RDS DB インスタンスに移行できます。

- 独自のオートメーションを作成します。

  レポート、管理、または診断ツール用のカスタムオートメーションスクリプトを作成、スケジュール、および実行できます。

Amazon RDS Custom アーキテクチャ

Amazon RDS Custom アーキテクチャは Amazon RDS に基づいていますが、重要な違いがあります。RDS Custom アーキテクチャのキーコンポーネントを次の図表に示します。



トピック

- VPC
- RDS Custom のオートメーションとモニタリング
- Amazon S3
- AWS CloudTrail

VPC

Amazon RDS と同様に、RDS Custom DB インスタンスは 仮想プライベートクラウド (VPC) 内にあります。

RDS Custom DB インスタンスは、次の主要コンポーネントで構成されています。

- Amazon EC2 インスタンス
- インスタンスエンドポイント
- Amazon EC2 インスタンスにインストールされるオペレーティングシステム
- 追加のファイルシステムを含む Amazon EBS ストレージ

RDS Custom のオートメーションとモニタリング

RDS Custom には、DB インスタンスの外部で実行されるオートメーションフトウェアがあります。このソフトウェアは、DB インスタンス上のエージェントと RDS Custom 環境全体内の他のコンポーネントと通信します。

RDS Customのモニタリングおよびリカバリ機能は、Amazon RDS と同様の機能を提供します。デフォルトでは、RDS Custom はフルオートメーションモードになっています。オートメーションソフトウェアには、主に次のロールがあります。

- メトリクスを収集して通知を送信する
- インスタンスの自動復旧を実行する

RDS カスタムオートメーションの重要な責務は、Amazon EC2 インスタンスの問題に対応することです。さまざまな理由で、ホストに障害が発生したり、到達できなくなったりすることがあります。RDS Custom は、Amazon EC2 インスタンスを再起動または置換することによってこれらの問題を解決します。

トピック

- Amazon RDS Custom ホストの置換
- RDS Custom サポート範囲

Amazon RDS Custom ホストの置換

Amazon EC2 ホストに障害が発生すると、RDS Custom は再起動を試みます。この作業に失敗すると、RDS Custom は Amazon EC2 に含まれる同じ停止およびスタート機能を使用します。ホスト交換の際にお客様が見ることのできる変更は、新しいパブリック IP アドレスのみです。

トピック

- ホストの停止とスタート
- ホスト交換の影響
- Amazon EC2 のベストプラクティス

ホストの停止とスタート

RDS Custom は次のステップを自動的に実行し、ユーザーの介入は必要ありません。

1. Amazon EC2 ホストを停止します。

   EC2 インスタンスは正常なシャットダウンを実行し、実行を停止します。Amazon EBS ボリュームはインスタンスに接続されたままとなり、そのデータは保持されます。インスタンスストレージのボリューム (RDS Custom ではサポートされていません) またはホストコンピュータの RAM に保存されているデータはすべて消えます。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスの停止と起動」を参照してください。

2. Amazon EC2 ホストをスタートします。

   EC2 インスタンスは、基盤となる新しいホストハードウェアに移行します。場合によっては、RDS カスタム DB インスタンスは元のホストに残ります。

ホスト交換の影響

RDS Custom では、ルートデバイスボリュームと Amazon EBS ストレージボリュームを完全に制御できます。ルートボリュームは、失いたくない重要なデータや設定を含むことがあります。

RDS Custom for Oracle は、ルートボリュームデータを含めて、すべてのデータベースと顧客データをオペレーション後に保持します。ユーザーによる介入は必要ありません。RDS Custom for SQL Server では、データベースデータは保持されますが、C: ドライブのデータ (オペレーティングシステムや顧客のデータなど) はすべて失われます。

ホストの交換プロセス後、Amazon EC2 ホストには新しいパブリック IP アドレスが割り当てられます。新しいホストは、次の表に示すデータとメタデータを保持します。

メタデータまたはデータ	RDS Custom for Oracle の保持	RDS Custom for SQL Server の保持
EC2のインスタンス ID	はい	いいえ
EC2 インスタンスのメタデータ	可能	はい
データストレージボリュームデータ	可能	はい
ルートボリュームのデータ	はい	いいえ
プライベート IP アドレス	可能	はい
Elastic IP アドレス	可能	はい

Amazon EC2 のベストプラクティス

Amazon EC2 ホスト交換特徴は、Amazon EC2 障害シナリオの大部分をカバーしています。ACM のベストプラクティスに従うことをおすすめします。

- 設定または OS を変更する前に、データをバックアップしてください。ルートボリュームまたはOSが破損した場合、ホストの交換では修復できません。DB スナップショットまたはポイントインタイムリカバリからの復元が唯一のオプションとなります。
- 物理 Amazon EC2 ホストをマニュアルで停止または終了しないでください。どちらのアクションでも、インスタンスを RDS Custom サポートペリメーターの外に配置することになります。
- (RDS Custom for SQL Server) Amazon EC2 ホストに追加のボリュームをアタッチする場合は、再起動時に再マウントするように設定します。ホストに障害が発生すると、RDS Custom はホストを自動的に停止してスタートすることがあります。

RDS Custom サポート範囲

RDS Custom は、周辺サポートと呼ばれる追加のモニタリング機能を提供しています。この追加のモニタリングにより、RDS Custom DB インスタンスがサポート対象の AWS インフラストラクチャ、オペレーティングシステム、およびデータベースを使用していることが保証されます。

サポート境界は、DB インスタンスが RDS Custom for Oracle でサポートされていない構成の修正 と RDS Custom for SQL Server DB でサポートされていない構成の修正 にリストされている要件に準拠していることを確認します。これらの要件のいずれかが満たされていないと、RDS Custom は DB インスタンスがサポートペリメーターの範囲外であるとみなします。

トピック

- RDS Custom でサポートされていない構成
- サポートされていない構成のトラブルシューティング

RDS Custom でサポートされていない構成

DB インスタンスがサポート範囲外にある場合、RDS Custom は DB インスタンスステータスを `unsupported-configuration` に変更し、イベント通知を送信します。設定の問題を修正すると、RDS Custom によって DB インスタンスのステータスが `available` に戻されます。

DB インスタンスが `unsupported-configuration` 状態のときは、次のようになります。

- データベースにアクセスできる。例外は、データベースが予期せずシャットダウンしているため、DB インスタンスが unsupported-configuration である場合です。
- DB インスタンスを変更できません。
- DB スナップショットを作成できません。
- 自動バックアップが作成されない。
- RDS Custom for SQL Server DB インスタンスに限り、RDS Custom は基盤となる Amazon EC2 インスタンスに障害が発生しても置換されません。ホスト置換の詳細については、「Amazon RDS Custom ホストの置換」を参照してください。
- DB インスタンスは削除できますが、他のほとんどの RDS カスタム API オペレーションは使用できません。
- RDS Custom は、REDO ログファイルをアーカイブして Amazon S3 にアップロードすることで、ポイントインタイムリカバリ (PITR) を引き続きサポートします。unsupported-configuration 状態の PITR は、以下の点で異なります。
  - PITR は、新しい RDS Custom DB インスタンスに完全に復元するには時間がかかることがあります。この状況は、インスタンスが unsupported-configuration の状態にあると、自動スナップショットもマニュアルスナップショットも取得できないためです。
  - PITRは、インスタンスがunsupported-configuration状態になる前に取得した最新のスナップショットから始まる、より多くの REDO ログを再生しなければなりません。
  - 場合によっては、アーカイブされた redo ログファイルのアップロード機能に影響を与える変更を行ったために、DB インスタンスが unsupported-configuration 状態になっていることがあります。例としては、EC2 インスタンスの停止、RDS カスタムエージェントの停止、EBS ボリュームのデタッチなどがあります。このような場合、PITR は DB インスタンスを最新の復元可能な時間に復元することができません。

サポートされていない構成のトラブルシューティング

RDS Custom は、unsupported-configuration 状態のトラブルシューティングガイダンスを提供します。ガイダンスの一部は RDS Custom for Oracle と RDS Custom for SQL Server 両方に適用されますが、その他のガイダンスはお使いの DB エンジンに依存します。エンジン別のトラブルシューティング情報については、以下のトピックを参照してください。

- RDS Custom for Oracle でサポートされていない構成の修正
- RDS Custom for SQL Server DB でサポートされていない構成の修正

Amazon S3

RDS Custom for Oracle を使用する場合は、ユーザー作成の Simple Storage Service (Amazon S3) バケットにインストールメディアをアップロードします。RDS Custom for Oracle は、このバケット内のメディアを使用してカスタムエンジンバージョン (CEV) を作成します。CEVは、データベースバージョンと Amazon マシンイメージ (AMI) のバイナリボリュームスナップショットです。CEV から RDS Custom DB インスタンスを作成できます。詳細については、「Amazon RDS Custom for Oracle のカスタムエンジンバージョンでの作業」を参照してください。

RDS Custom for Oracle および RDS Custom for SQL Server の両方で、RDS Custom は、文字列 `do-not-delete-rds-custom-` のプレフィックスが付いた Simple Storage Service (Amazon S3) バケットを自動的に作成します。RDS Custom は、`do-not-delete-rds-custom-` S3 バケットを使用して、次のタイプのファイルを保存します。

- AWS CloudTrailRDS Custom によって作成された証跡の ログ
- 境界アーティファクトのサポート (RDS Custom サポート範囲 を参照)
- データベース REDO ログファイル (RDS Custom for Oracle のみ):
- トランザクションログ (RDS Custom for SQL Server のみ)
- カスタムエンジンバージョンのアーティファクト (RDS Custom for Oracle のみ)

次のリソースのいずれかを作成するとき、RDS Custom は、`do-not-delete-rds-custom-` S3 バケットを作成します。

- RDS Custom for Oracle の最初の CEV
- RDS Custom for SQL Server の最初の DB インスタンス

RDS Custom は、次の組み合わせごとに 1 つのバケットを作成します。

- AWS アカウント ID
- エンジンタイプ (RDS Custom for Oracle または RDS Custom for SQL Server のいずれか)
- AWS リージョン

例えば、RDS Custom for Oracle CEV を 1 つの AWS リージョン に作成すると、1 つの `do-not-delete-rds-custom-` バケットが存在します。複数の RDS Custom for SQL Server インスタンスを作成し、それらが異なる AWS リージョン に存在する場合、1 つの `do-not-delete-rds-`

custom- バケットがそれぞれの AWS リージョン に存在します。1 つの RDS Custom for Oracle インスタンスと 2 つの RDS Custom for SQL Server インスタンスを 1 つの AWS リージョン に作成する場合、2 つのdo-not-delete-rds-custom- バケットが存在します。

AWS CloudTrail

RDS Custom は、名前が do-not-delete-rds-custom- で始まる AWS CloudTrail 追跡を自動的に作成します。RDS Custom サポートの周辺構成は CloudTrail からのイベントに依存して、アクションが RDS Custom オートメーションに影響を与えるかどうかを判断します。詳細については、「サポートされていない構成のトラブルシューティング」を参照してください。

RDS Custom は、最初の DB インスタンスを作成するときに追跡を作成します。RDS Custom は、次の組み合わせごとに 1 つの追跡を作成します。

- AWS アカウント ID
- エンジンタイプ (RDS Custom for Oracle または RDS Custom for SQL Server のいずれか)
- AWS リージョン

RDS カスタム DB インスタンスを削除しても、このインスタンスの CloudTrail は自動的に削除されません。この場合、AWS アカウント は、削除されていない CloudTrail の料金を引き続き請求されます。RDS Custom は、このリソースの削除について責任を負いません。CloudTrail を手動で削除する方法については、「AWS CloudTrailユーザーガイド」の「トレイルを削除する」を参照してください。

Amazon RDS Custom のセキュリティ

RDS Custom のセキュリティ上の考慮事項を、よく理解しておきます。

RDS Custom のセキュリティの詳細については、以下のトピックを参照してください。

- Amazon S3 バケットを、「混乱した代理」問題から保護する
- コンプライアンスプログラムのための RDS Custom for Oracle の認証情報のローテーション

RDS Custom がユーザーに代わってタスクを安全に管理する方法

RDS Custom は、次のツールとテクニックを使用して、ユーザーに代わって安全に操作を実行します。

AWSServiceRoleForRDSCustom サービスリンクロール

サービスリンクロールはサービスによって事前に定義されており、サービスがユーザーに代わって他の AWS のサービス サービスを呼び出すために必要なアクセス許可がすべて含まれています。RDS Custom の場合、`AWSServiceRoleForRDSCustom` は最小特権の原則に従って定義されるサービスリンクロールです。RDS Custom は `AmazonRDSCustomServiceRolePolicy` で特権を使用します。これは、このロールにアタッチされているポリシーであり、ほとんどのプロビジョニングタスクとすべてのオフホスト管理タスクを実行します。詳細については、「AmazonRDSCustomServiceRolePolicy」を参照してください。

ホスト上でタスクを実行する場合、RDS Custom オートメーションはサービスリンクロールの認証情報を使用し、AWS Systems Manager を使用してコマンドを実行します。Systems Manager のコマンド履歴と AWS CloudTrail を使用してコマンド履歴を監査できます。Systems Manager は、ネットワーク設定を使用して RDS Custom DB インスタンスに接続します。詳細については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。

一時 IAM 認証情報

リソースをプロビジョニングまたは削除する際、RDS Custom は呼び出し元の IAM プリンシパルの認証情報から取得した一時的な認証情報を使用することがあります。これらの IAM 認証情報は、そのプリンシパルにアタッチされている IAM ポリシーによって制限され、そのオペレーションが完了すると失効します。RDS Custom を使用する IAM プリンシパルに必要なアクセス許可については、「ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する」を参照してください。

Amazon EC2 インスタンスプロファイル

EC2 インスタンスプロファイルは IAM ロールのコンテナであり、EC2 インスタンスにロール情報を渡すために使用できます。EC2 インスタンスは RDS Custom DB インスタンスの基礎となります。RDS Custom DB インスタンスを作成するときに、インスタンスプロファイルを提供します。RDS Custom は、バックアップなどのホストベースの管理タスクを実行する際に EC2 インスタンスプロファイルの認証情報を使用します。詳細については、「IAM ロールおよびインスタンスプロファイルをマニュアルで作成する」を参照してください。

SSH キーペア

RDS Custom が DB インスタンスの基盤となる EC2 インスタンスを作成すると、ユーザーに代わって SSH キーペアが作成されます。キーは命名プレフィックス `do-not-delete-rds-custom-ssh-privatekey-db-` または `rds-custom!oracle-do-not-delete-`*`db_resource_id-uuid`*`-ssh-privatekey` を使用します。AWS Secrets Manager は

この SSH プライベートキーをシークレットとして AWS アカウントに保存します。Amazon RDS はこれらの認証情報を保存、アクセス、または使用しません。詳細については、「Amazon EC2 キーペアおよび Linux インスタンス」を参照してください。

SSL 証明書

RDS Custom DB インスタンスは、マネージド SSL 証明書をサポートしていません。SSL をデプロイする場合は、SSL 証明書を自分のウォレットでセルフマネージし、SSL リスナーを作成してクライアントデータベース間の接続を保護したり、データベースのレプリケーションを行ったりできます。詳細については、Oracle Database ドキュメントの「Transport Layer Security 認証の構成」を参照してください。

Amazon S3 バケットを、「混乱した代理」問題から保護する

Amazon RDS Custom for Oracle カスタムエンジンバージョン (CEV) または RDS Custom for SQL Server DB インスタンスを作成すると、RDS Custom は Amazon S3 バケットを作成します。S3 バケットには、CEV アーティファクト、REDO (トランザクション) ログ、サポートペリメーターの設定項目、AWS CloudTrail ログなどのファイルが保存されます。

これらの S3 バケットをより安全にするために、グローバル条件コンテキストキーを使用することで、「混乱した代理」問題を防止することができます。(詳しくは、「サービス間での混乱した代理問題の防止」を参照してください。)

次の RDS Custom for Oracle の例では、S3 バケットポリシーで `aws:SourceArn` と`aws:SourceAccount` のグローバル条件コンテキストキーを使用することを示します。RDS Custom for Oracle の場合は、CEV と DB インスタンスの Amazon リソースネーム (ARN) を必ず含めてください。RDS Custom for SQL Server の場合は、DB インスタンスの ARN を必ず含めてください。

```
...
{
  "Sid": "AWSRDSCustomForOracleInstancesObjectLevelAccess",
  "Effect": "Allow",
  "Principal": {
     "Service": "custom.rds.amazonaws.com"
  },
  "Action": [
     "s3:GetObject",
     "s3:GetObjectVersion",
```

```
      "s3:DeleteObject",
      "s3:DeleteObjectVersion",
      "s3:GetObjectRetention",
      "s3:BypassGovernanceRetention"
   ],
   "Resource": "arn:aws:s3:::do-not-delete-rds-custom-123456789012-us-east-2-c8a6f7/
RDSCustomForOracle/Instances/*",
   "Condition": {
      "ArnLike": {
         "aws:SourceArn": [
             "arn:aws:rds:us-east-2:123456789012:db:*",
             "arn:aws:rds:us-east-2:123456789012:cev:*/*"
         ]
      },
      "StringEquals": {
         "aws:SourceAccount": "123456789012"
     }
   }
},
...
```

コンプライアンスプログラムのための RDS Custom for Oracle の認証情報のローテーション

一部のコンプライアンスプログラムでは、データベースユーザーの認証情報を定期的に (例えば、90 日ごとに) 変更する必要があります。RDS Custom for Oracle は、一部の定義済みデータベースユーザーの認証情報を自動的にローテーションします。

トピック

- 事前定義済みユーザーの認証情報の自動ローテーション
- ユーザー認証情報のローテーションに関するガイドライン
- ユーザー認証情報の手動ローテーション

事前定義済みユーザーの認証情報の自動ローテーション

RDS Custom for Oracle DB インスタンスが Amazon RDS でホストされている場合、以下の事前定義された Oracle ユーザーの認証情報は 30 日ごとに自動的にローテーションされます。前述のユーザーの認証情報は AWS Secrets Manager にあります。

データベースユーザー	Created by (作成者)	サポート対象エンジンバージョン	注意事項
SYS	Oracle	custom-oracle-ee custom-oracle-ee-cdb custom-oracle-se2 custom-oracle-se2-cdb	
SYSTEM	Oracle	custom-oracle-ee custom-oracle-ee-cdb custom-oracle-se2 custom-oracle-se2-cdb	
RDSADMIN	RDS	custom-oracle-ee custom-oracle-se2	
C##RDSADMIN	RDS	custom-oracle-ee-cdb custom-oracle-se2-cdb	C## プレフィックスが付いたユーザー名は CDB にのみ存在します。CDB の詳細については、「Amazon RDS Custom for Oracle アーキテクチャの概要」を参照してください。
RDS_DATAGUARD	RDS	custom-oracle-ee	このユーザーは、リードレプリカ、リードレプリカのソースデータベース、および Oracle Data Guard を使用して RDS Custom に物理的に移行したデータベースにのみ存在します。

データベースユーザー	Created by (作成者)	サポート対象エンジンバージョン	注意事項
C##RDS_DATAGUARD	RDS	custom-oracle-ee-cdb	このユーザーは、リードレプリカ、リードレプリカのソースデータベース、および Oracle Data Guard を使用して RDS Custom に物理的に移行したデータベースにのみ存在します。C## プレフィックスが付いたユーザー名は CDB にのみ存在します。CDB の詳細については、「Amazon RDS Custom for Oracle アーキテクチャの概要」を参照してください。

自動認証情報のローテーションの例外は、スタンバイデータベースとして手動で設定した RDS Custom for Oracle DB インスタンスです。RDS は、`create-db-instance-read-replica` CLI コマンドまたは `CreateDBInstanceReadReplica` API を使用して作成したリードレプリカの認証情報のみを更新します。

ユーザー認証情報のローテーションに関するガイドライン

コンプライアンスプログラムに従って資格情報が更新されるように、次のガイドラインに注意してください。

- DB インスタンスが認証情報を自動的にローテーションする場合、「事前定義された Oracle ユーザー」にリストされているユーザーのシークレット、パスワードファイル、またはパスワードを手動で変更または削除しないでください。そうしないと、RDS Custom が DB インスタンスをサポート範囲外に配置し、自動ローテーションが停止する可能性があります。
- RDS マスターユーザーは事前に定義されていないため、パスワードを手動で変更するか、Secrets Manager で自動ローテーションを設定する必要があります。詳細については、「AWS Secrets Manager シークレットのローテーション」を参照してください。

ユーザー認証情報の手動ローテーション

次のカテゴリのデータベースでは、RDS は「事前定義済み Oracle ユーザー」にリストされているユーザーの認証情報を自動的に更新しません。

- スタンバイデータベースとして機能するように手動で設定したデータベース。
- オンプレミスのデータベース
- サポート範囲外にある DB インスタンス、または RDS Custom オートメーションを実行できない状態にある DB インスタンス。この場合、RDS Custom はキーをローテーションしません。

データベースが上記のカテゴリのいずれかに該当する場合は、ユーザー認証情報を手動でローテーションする必要があります。

DB インスタンスのユーザー認証情報を手動でローテーションするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. データベースで、RDS が現在 DB インスタンスをバックアップしていないか、高可用性の設定などの操作を実行していないことを確認します。
3. データベースの詳細ページで、[設定] を選択し、DB インスタンスのリソース ID を書き留めます。または、AWS CLI コマンド `describe-db-instances` を使用できます。
4. https://console.aws.amazon.com/secretsmanager/ から Secrets Manager コンソールを開きます。
5. 検索ボックスにデータベースのリソース ID を入力し、次のいずれかの命名規則を使用してシークレットを検索します。

   ```
   do-not-delete-rds-custom-resource_id-uuid
   rds-custom!oracle-do-not-delete-resource_id-uuid
   ```

   このシークレットには、RDSADMIN、SYS、および SYSTEM のパスワードが保存されます。次のサンプルキーは、DB リソース ID db-ABCDEFG12HIJKLNMNOPQRS3TUVWX および UUID 123456 の DB インスタンス用です。

   ```
   do-not-delete-rds-custom-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-123456
   rds-custom!oracle-do-not-delete-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-123456
   ```

> **Important**
>
> DB インスタンスがリードレプリカで custom-oracle-ee-cdb エンジンを使用する場合、サフィックス *db-resource_id-uuid* が付いた 2 つのシークレットが存在します。1 つはマスターユーザー用で、もう 1 つは RDSADMIN、SYS および SYSTEM です。正しいシークレットを見つけるには、次のコマンドをホストで実行します。
>
> ```
> cat /opt/aws/rdscustomagent/config/database_metadata.json | python3 -c
>  "import sys,json; print(json.load(sys.stdin)['dbMonitoringUserPassword'])"
> ```
>
> この dbMonitoringUserPassword 属性は、RDSADMIN、SYS、および SYSTEM のシークレットを示します。

6. DB インスタンスが Oracle Data Guard 設定に存在する場合は、次のいずれかの命名規則を使用してシークレットを検索します。

   ```
   do-not-delete-rds-custom-resource_id-uuid-dg
   rds-custom!oracle-do-not-delete-resource_id-uuid-dg
   ```

   このシークレットには、RDS_DATAGUARD のパスワードが保存されます。次のサンプルキーは、DB リソース ID db-ABCDEFG12HIJKLNMNOPQRS3TUVWX および UUID 789012 の DB インスタンス用です。

   ```
   do-not-delete-rds-custom-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-789012-dg
   rds-custom!oracle-do-not-delete-db-ABCDEFG12HIJKLNMNOPQRS3TUVWX-789012-dg
   ```

7. 「定義済みの Oracle ユーザー」に記載されているすべてのデータベースユーザーについて、「AWS Secrets Manager シークレットの変更」の指示に従ってパスワードを更新します。

8. データベースがスタンドアロンデータベースまたは Oracle Data Guard 設定のソースデータベースの場合:

   a. Oracle SQL クライアントを起動し、SYS としてログインします。

   b. 「事前定義済み Oracle ユーザー」にリストされているデータベースユーザーごとに、次の形式の SQL ステートメントを実行します。

      ```
      ALTER USER user-name IDENTIFIED BY pwd-from-secrets-manager ACCOUNT UNLOCK;
      ```

例えば、Secrets Manager に保存されている RDSADMIN の新しいパスワードが pwd-123 の場合、次のステートメントを実行します。

```
ALTER USER RDSADMIN IDENTIFIED BY pwd-123 ACCOUNT UNLOCK;
```

9. DB インスタンスが Oracle Database 12c リリース 1 (12.1) を実行していて、Oracle Data Guard によって管理されている場合は、プライマリ DB インスタンスから各スタンバイ DB インスタンスにパスワードファイル (orapw) を手動でコピーします。

   DB インスタンスが Amazon RDS でホストされている場合、パスワードファイルの場所は /rdsdbdata/config/orapw です。Amazon RDS でホストされていないデータベースの場合、デフォルトの場所は、Linux と UNIX では $ORACLE_HOME/dbs/orapw$ORACLE_SID、Windows では %ORACLE_HOME%\database\PWD%ORACLE_SID%.ora です。

RDS Custom for Oracle を使用する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle DB インスタンスの作成、管理、保守の手順を以下に示します。

トピック

- RDS Custom for Oracle のサポート終了
- RDS Custom for Oracle ワークフロー
- Amazon RDS Custom for Oracle のデータベースアーキテクチャ
- RDS Custom for Oracle の機能の可用性とサポート
- RDS Custom for Oracle の要件と制限
- Amazon RDS Custom for Oracle の環境設定
- Amazon RDS Custom for Oracle のカスタムエンジンバージョンでの作業
- Amazon RDS Custom for Oracle DB インスタンスの設定

- Amazon RDS Custom for Oracle DB インスタンスの管理
- RDS Custom for Oracle のマルチ AZ 配置の管理
- RDS Custom for Oracle の Oracle レプリカの使用
- Amazon RDS Custom for Oracle DB インスタンスのバックアップと復元
- RDS Custom for Oracle のオプショングループ使用する
- オンプレミスデータベースの RDS Custom for Oracle への移行
- Amazon RDS Custom for Oracle DB インスタンスのアップグレード
- Amazon RDS Custom for Oracle の DB に関する問題のトラブルシューティング
- Amazon RDS Custom for Oracle の既知の問題

RDS Custom for Oracle のサポート終了

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

概要

慎重に検討した後、AWS は Amazon RDS Custom for Oracle サービスを中止することを決定しました。このサービスは、2027 年 3 月 31 日をもって廃止されます。この規範的ガイダンスでは、RDS Custom for Oracle から Amazon Elastic Compute Cloud (Amazon EC2) 上のセルフマネージド Oracle データベースへの移行に役立つ詳細な移行戦略を提供します。

主要なタイムライン

- 2026 年 3 月 31 日から 2027 年 3 月 31 日: RDS Custom for Oracle から EC2 での Oracle 実行に移行することをお勧めします。この期間中は、既存の機能とサポートを利用して RDS Custom for Oracle を引き続き使用できます。
- 2027 年 3 月 31 日以降: RDS Custom for Oracle サービスを使用できなくなります。

ターゲットオーディエンス

このガイダンスは、以下の対象者向けです。

- Oracle データベースの移行を担当するデータベース管理者
- 移行戦略を計画するクラウドアーキテクト
- データベースインフラストラクチャを管理する DevOps エンジニア
- 移行プロセスを監督する IT マネージャー

前提条件

開始する前に、以下があることを確認してください。

- Oracle 19c Enterprise Edition を実行するアクティブな Amazon RDS Custom for Oracle インスタンス
- EC2 インスタンスを作成および管理するための適切な AWS Identity and Access Management (IAM) アクセス許可
- データベースアーキテクチャ (非 CDB または PDB を含むマルチテナント CDB) に関する理解
- ソースインスタンスとターゲットインスタンス間のネットワーク接続計画
- 移行のバックアップと復旧戦略

移行オプション

移行プロセスの一環として、ビジネス要件とユースケースに基づいて、次の 2 つの移行オプションから 1 つを選択できます。

オプション 1: RMAN アクティブ複製 (オンライン/オフライン移行)

以下に最適:

- 最終的なカットオーバー中に計画的なダウンタイムを許容できるワークロード
- 可動部分が少なく移行要件が簡素
- 簡単な 1 回限りの移行が必要なデータベース
- カットオーバー前に継続的な同期を必要としないシナリオ

主な特徴

- ダウンタイム: 最終カットオーバー中のダウンタイムを最小限に抑えます (データベースは複製中はオンラインのままで、最終カットオーバー時のダウンタイムは短い)
- 複雑さ: データベースを直接複製することで複雑さを軽減します
- 所要時間: 移行時間はデータベースのサイズとネットワーク帯域幅によって異なります
- フォールバック: 検証が完了するまでソースデータベースを維持する必要があります
- オンライン機能: ソースデータベースは、複製処理中もオンラインのままでアクセス可能です

移行アプローチ: RMAN アクティブ複製では、稼働中のソースデータベースからネットワーク経由でデータベースファイルをコピーすることで、ターゲット上にソースデータベースの正確なコピーを作成します。ソースデータベースはオンラインのままで、複製処理中もアプリケーションからアクセスできます。マルチテナントデータベースの場合、RMAN は、CDB$ROOT、PDB$SEED、およびすべてのプラガブルデータベースを含む CDB 全体を単一のオペレーションで自動的に複製します。アプリケーションを新しい EC2 インスタンスにリダイレクトするために必要なのは、ごく短いカットオーバーウィンドウのみです。

オプション 2: Oracle Data Guard (オンライン移行)

以下に最適:

- 最小限のダウンタイムを必要とする本稼働ワークロード
- 可用性を維持する必要があるミッションクリティカルなデータベース
- カットオーバー前に継続的な同期が必要なシナリオ
- 組み込みフォールバック機能を必要とする移行

主な特徴

- ダウンタイム: ダウンタイムはほぼゼロ (スイッチオーバーは数秒から数分)
- 複雑さ: Data Guard 設定による複雑性の増加
- 期間: 初期セットアップ時間とスイッチオーバーまでの継続的な同期時間
- フォールバック: ソースをスタンバイとして保持することでフォールバック機能を組み込み

移行アプローチ: Oracle Data Guard は、プライマリデータベースから REDO ログを継続的に配布して適用することで、同期されたスタンバイデータベースを維持します。移行を完了する準備ができた

ら、最小限のダウンタイムで EC2 スタンバイをプライマリに昇格させるスイッチオーバーを実行します。マルチテナントデータベースの場合、Data Guard はすべての PDB を含む CDB 全体を自動的に保護します。

ディシジョンマトリックス

次のマトリックスを使用して、適切な移行オプションを選択します。

側面	RMAN アクティブ複製	Oracle Data Guard:
ソースデータベースの可用性	複製中はオンライン	プロセス全体を通してオンライン
許容できるダウンタイム	数分から数時間 (最終カットオーバー)	数秒から数分 (スイッチオーバー)
移行の複雑さ	Lower	より高い
継続的な同期	いいえ	はい
フォールバック機能	手動 (ソースを保持)	組み込み (自動)
カットオーバー前のテスト	制限あり	完全なテストが可能
ネットワーク帯域幅要件	複製中は高	中程度 (連続)
次の用途に適しています	ほとんどの移行、開発/テスト、計画されたカットオーバー	本番環境、ミッションクリティカル、ほぼゼロのダウンタイム

アーキテクチャに関する考慮事項

どちらの移行オプションも、2 つの Oracle データベースアーキテクチャをサポートしています。

非 CDB:

マルチテナントアーキテクチャを持たない従来の単一インスタンス Oracle データベース。これらのデータベースは次の特徴を持ちます。

- 単一のデータベースインスタンスを持つ

- プラガブルデータベース (PDB) を使用しない
- 管理と移行が簡単
- RDS Custom for Oracle では通常 ORCL という名前

マルチテナント (PDB を含む CDB)

Oracle 12c で導入された複数のプラガブルデータベース (PDB) をホストするコンテナデータベース (CDB)。これらのデータベースは次の特徴を持ちます。

- CDB$ROOT および PDB$SEED でコンテナデータベース (CDB) を使用する
- 1 つ以上のプラガブルデータベース (PDB) ホストする
- データベース統合とリソース分離を提供する
- RDS Custom for Oracle では通常 RDSCDB という名前
- PDB データファイルには、GUID ベースのサブディレクトリを持つ Oracle Managed Files (OMF) を使用する

マルチテナント移行に関する重要な注意点: 移行プロセス中にターゲットデータベースの名前が ORCL に変更されます (ソース: RDSCDB → ターゲット: ORCL)。これにより、ターゲット設定が簡素化され、標準的な命名規則に準拠します。

移行アプローチの主な違い

側面™	非 CDB:	マルチテナント (PDB を含む CDB)
移行範囲	単一のデータベース	すべての PDB を含む CDB 全体
移行後の状態	データベースは READ WRITE でオープン	CDB は READ WRITE でオープン、PDB は MOUNTED 状態
PDB 管理	該当なし	PDB を開き、自動オープンを設定する必要があります
クリーンアップ	単一のデータベース	CDB$ROOT (PDB にカスケード)、C## 共通ユーザーを扱う

側面™	非 CDB:	マルチテナント (PDB を含む CDB)
アプリケーション接続	データベースサービス	PDB サービス (CDB ではない)
パラメータファイル	標準パラメータ	enable_pluggable_database=TRUE が必要
複雑さ	Lower	複数のコンテナが原因で高

両方の移行オプションに共通する前提条件

いずれの移行オプションを開始する前に、以下の前提条件のステップを完了してください。

1. EC2 インスタンスを起動して設定する

   以下の点を考慮して、EC2 インスタンスを起動します。

   - インスタンスタイプ: ワークロードのリソース要件を満たす EC2 インスタンスタイプを選択します。開始ポイントとして、RDS Custom インスタンスと同じインスタンスクラスを使用することをお勧めします。メモリ、CPU、ネットワーク帯域幅の要件を検討してください。
   - オペレーティングシステム: Oracle Linux または Red Hat Enterprise Linux (RDS Custom OS バージョンと一致するか、互換性のあるもの)
   - Oracle ソフトウェア: Oracle Database ソフトウェア (同じメジャーバージョン、マイナーバージョン、リリース更新、理想的には RDS Custom と同じワンオフパッチ) をインストールします。Oracle ソフトウェアが /u01/app/oracle/ または任意の場所にインストールされていることを確認します。
   - ストレージ: ワークロードの要件を満たすために、適切なサイズと IOPS で Amazon EBS ボリュームを設定します。コスト効率の高いパフォーマンスには gp3 ボリュームを使用し、高性能ワークロードには io2 Block Express を使用することを検討してください。

2. ストレージアーキテクチャを設定する

   a. ファイルシステムストレージ (ほとんどのシナリオで推奨)

      - Oracle データファイルに標準ファイルシステムディレクトリを使用します
      - 管理が簡単で、ほとんどのワークロードに適しています
      - このガイダンスでは、ファイルシステムストレージの例を使用します

   b. Oracle Automatic Storage Management (ASM)

- ワークロードに ASM が必要な場合は、EC2 インスタンスにスタンドアロン ASM をインストールして設定します
- ASM ディスクグループ (+DATA、+FRA など) を使用するように init ファイル内のすべてのパスパラメータを調整します
- 移行プロセスは ASM でも同様ですが、パスを調整します

3. ファイル転送メカニズムを設定する

   RDS Custom インスタンスと EC2 インスタンス間でファイルを転送するメカニズムを作成します。これには複数のオプションがあります。

   a. オプション A: Amazon S3 (ほとんどのシナリオで推奨)
      - Amazon S3 バケットを作成するか、既存のバケットを使用します
      - 両方のインスタンスに AWS CLI をインストールして設定します
      - 手順については、「AWS CLI の開始方法」を参照してください。

   b. オプション B: SCP/SFTP による直接転送
      - インスタンス間で SSH ポートが開いている場合は、ファイルを直接転送できます
      - パラメータファイルやパスワードファイルなどの小さなファイルに適しています

   c. オプション C: Amazon EFS
      - 両方のインスタンスに Amazon EFS が既にマウントされている場合は、共有ファイルシステムとして使用できます
      - 既存の EFS インフラストラクチャを持つ環境に適しています

   このガイダンスでは、Amazon S3 を例として使用していますが、選択したメソッドに合わせてコマンドを適応させることができます。

4. ネットワーク接続の設定

   RDS Custom インスタンスと EC2 インスタンス間のネットワーク接続を確認します。

   - 同じ VPC: セキュリティグループは Oracle リスナーポート (デフォルトは 1521、またはカスタムポート) で双方向トラフィックを許可する必要があります
   - 異なる VPC (同じアカウント): VPC ピアリングを設定し、ルートテーブルとセキュリティグループを設定します
   - 異なるアカウント: アカウント間で VPC ピアリングを設定するか、AWS Transit Gateway を使用します
   - 接続の検証: ping と telnet を使用してデータベースポートの接続をテストします

5. EC2 でディレクトリ構造を作成する

ディレクトリ構造は、データベースアーキテクチャによって異なります。

Example非 CDB の場合。

```
# Non-CDB directories
mkdir -p /u01/app/oracle/oradata/ORCL/controlfile/
mkdir -p /u01/app/oracle/oradata/ORCL/datafile
mkdir -p /u01/app/oracle/oradata/ORCL/onlinelog
mkdir -p /u01/app/oracle/oradata/ORCL/arch
mkdir -p /u01/app/oracle/admin/ORCL/adump
mkdir -p /u01/app/oracle/backup
# Set ownership
chown -R oracle:oinstall /u01/app/oracle/oradata/ORCL
chown -R oracle:oinstall /u01/app/oracle/admin/ORCL
chown -R oracle:oinstall /u01/app/oracle/backup
```

Exampleマルチテナント (PDB を含む CDB) の場合。

```
# CDB directories
mkdir -p /u01/app/oracle/oradata/ORCL/controlfile/
mkdir -p /u01/app/oracle/oradata/ORCL/cdb/datafile
mkdir -p /u01/app/oracle/oradata/ORCL/pdbseed/datafile
mkdir -p /u01/app/oracle/oradata/ORCL/onlinelog
mkdir -p /u01/app/oracle/oradata/ORCL/arch
mkdir -p /u01/app/oracle/admin/ORCL/adump
mkdir -p /u01/app/oracle/backup
# PDB directories (RDS Custom uses OMF with GUID-based paths)
# Create a generic pdb directory - migration will create subdirectories as needed
mkdir -p /u01/app/oracle/oradata/ORCL/pdb/datafile
# Set ownership
chown -R oracle:oinstall /u01/app/oracle/oradata/ORCL
chown -R oracle:oinstall /u01/app/oracle/admin/ORCL
chown -R oracle:oinstall /u01/app/oracle/backup
```

> **Note**
>
> RDS Custom for Oracle は、GUID ベースのサブディレクトリ (例: `/rdsdbdata/db/pdb/RDSCDB_A/{GUID}/datafile/`) を持つ PDB データファイルに Oracle Managed

Files (OMF) を使用します。移行プロセスでは、ターゲットに必要なサブディレクトリ構造が自動的に作成されます。親ディレクトリを作成するだけで済みます。

ストレージ戦略: /u01/app/oracle/backup に別の EBS ボリュームを使用することを検討してください。これにより、移行が完了したら簡単にデタッチおよび削除し、ストレージコストを削減できます。

6. ソースデータベースの設定を確認します

移行を開始する前に、ソースデータベースの設定を確認します。

1. RDS Custom データベースホストに rdsdb ユーザーとしてログインし、環境を設定します。

   Example

```
# For non-CDB
export ORACLE_HOME=/rdsdbbin/oracle.19.custom.r1.EE.1
export ORACLE_SID=ORCL
export PATH=$ORACLE_HOME/bin:$PATH

# For multitenant CDB
export ORACLE_HOME=/rdsdbbin/oracle.19.custom.r1.EE-CDB.1
export ORACLE_SID=RDSCDB
export PATH=$ORACLE_HOME/bin:$PATH
```

2. データベースに接続し、アーキテクチャを確認します。

   Example

```
sqlplus / as sysdba
SQL> SELECT name, cdb, open_mode, log_mode FROM v$database;
```

   Example非 CDB の場合、予想される出力

```
NAME CDB OPEN_MODE                LOG_MODE
--------- --- -------------------- ------------
ORCL NO  READ  WRITE               ARCHIVELOG
```

Exampleマルチテナント (CDB) の場合、予想される出力。

```
NAME    CDB  OPEN_MODE             LOG_MODE
--------- --- -------------------- ------------
RDSCDB    YES READ WRITE           ARCHIVELOG
```

3. マルチテナント CDB がある場合は、すべての PDB とそのステータスを一覧表示します。

   Example

   ```
   SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
   ```

   予想される出力 (ORCLDB という名前の PDB が 1 つある場合の例)。

   Example

   ```
   CON_ID     NAME                           OPEN_MODE  RES
   ---------- ------------------------------ ---------- ---
   2          PDB$SEED                       READ ONLY  NO
   3          ORCLDB                         READ WRITE NO
   ```

4. データベースの合計サイズを確認します。

   Example非 CDB の場合:

   ```
   SQL> SELECT SUM(bytes)/1024/1024/1024 AS total_size_gb FROM dba_data_files;
   ```

   Exampleマルチテナントの場合。

   ```
   -- Total CDB size (all containers)
   SQL> SELECT SUM(bytes)/1024/1024/1024 AS total_size_gb FROM cdb_data_files;
   -- Size per PDB
   SQL> SELECT p.name AS pdb_name,
         ROUND(SUM(d.bytes)/1024/1024/1024, 2) AS size_gb
   FROM v$pdbs p
   JOIN cdb_data_files d ON p.con_id = d.con_id
   GROUP BY p.name, p.con_id
   ORDER BY p.con_id;
   ```

オプション 1: RMAN アクティブ複製を使用した物理移行

トピック

- RMAN アクティブ複製を使用するタイミング
- RMAN アクティブ複製の概要
- RMAN アクティブ複製の移行ワークフロー
- ステップ 1: Amazon RDS Custom オートメーションを一時停止する
- ステップ 2: パスワードとパラメータファイルを作成する
- ステップ 3: EC2 を使用してファイルを転送する
- ステップ 4: EC2 でパラメータファイルを編集する
- ステップ 5: TNS とリスナーを設定する
- ステップ 6: EC2 でデータベースを NOMOUNT で起動する
- ステップ 7: RMAN アクティブ複製を実行する
- ステップ 8: PDB を開く (マルチテナントのみ)
- ステップ 9: RDS Custom オブジェクトを削除する
- ステップ 10: 自動起動を設定する
- ステップ 11: 最終検証

このセクションでは、RMAN アクティブ複製を使用して Oracle データベースを RDS Custom for Oracle から EC2 に移行する詳細な手順について説明します。このメソッドでは、稼働中のデータベースから複製され、移行プロセス中にソースをオンラインに保ち、アクセスできるようにします。

RMAN アクティブ複製を使用するタイミング

次の場合に RMAN アクティブ複製を選択します。

- 移行中にソースデータベースをオンラインにしてアクセス可能にしたい
- 最終的なアプリケーションリダイレクトのために、ごく短いカットオーバーウィンドウを利用できる
- 可動部分が少ない簡単な移行プロセスが必要
- データベースサイズとネットワーク帯域幅により、妥当な複製時間を確保できる
- カットオーバー前に継続的な同期は必要ない
- 本番稼働用データベース、開発用データベース、またはテスト用データベースを移行している

RMAN アクティブ複製の概要

RMAN アクティブ複製では、ソースデータベースのバックアップや、ソースデータベースをオフラインにする必要はありません。ソースがオンラインのままでアプリケーションにアクセス可能な状態を維持しながら、ネットワーク経由でデータベースファイルをコピーすることで、稼働中のソースデータベースを移行先に複製します。

主な利点。

- ソースはオンラインのまま: アプリケーションは複製中にソースデータベースに引き続きアクセスできます
- バックアップ不要: 中間バックアップストレージの必要性を排除します
- 直接ネットワーク転送: データベースファイルはソースからターゲットに直接コピーされます
- 自動整合性: RMAN は、複製するデータベースの整合性が保たれるようにします
- 単一オペレーション: マルチテナントの場合、すべての PDB を含む CDB 全体を単一のオペレーションで複製します

複製プロセスでは、すべてのデータファイル、コントロールファイル、REDO ログなど、ソースデータベースの正確なコピーがターゲットに作成されます。ソースデータベースは、複製プロセス全体を通してアプリケーションのリクエストを処理し続けます。最後に、アプリケーションをソースからターゲットにリダイレクトするため必要なのは、ごく短いカットオーバーウィンドウのみです。

一般的なタイムライン

1. 複製フェーズ (ソースはオンラインのまま): データベースのサイズに応じて 30 分から数時間
2. 検証フェーズ (ソースはオンラインのまま): 必要に応じて数時間から数日
3. カットオーバーフェーズ (短時間のダウンタイム): アプリケーションを EC2 にリダイレクトするのに数分

RMAN アクティブ複製の移行ワークフロー

RDS Custom DB インスタンス (ソース) のステップ。

1. Amazon RDS Custom オートメーションを一時停止する
2. データベースアーキテクチャを検証する (非 CDB または PDB を含む CDB)
3. ソースデータベースからパスワードファイルとパラメータファイルを作成する
4. パスワードファイルとパラメータファイルをターゲット EC2 インスタンスにコピーする

5. ソースデータベースがアーカイブログモードで実行されていることを確認する
6. RDS Custom DB サーバーで tnsnames.ora を設定する

EC2 DB インスタンス (ターゲット) のステップ。

1. EC2 のパラメータファイルを編集する (アーキテクチャ固有)
2. EC2 で tnsnames.ora を設定する
3. EC2 インスタンスの環境を設定する
4. EC2 で Oracle Listener を設定する
5. EC2 でデータベースを NOMOUNT 状態で起動する

RMAN 複製のステップ。

1. RMAN アクティブ複製を実行する
2. データベースを (マルチテナントの場合は PDB も) 開く
3. PDB の自動オープンを設定する (マルチテナントのみ)
4. RDS Custom 固有のユーザーとオブジェクトを削除する
5. SPFILE を作成し、自動起動を設定する
6. ソースで Amazon RDS Custom オートメーションを再開する (アクティブな状態を維持する場合)

ステップ 1: Amazon RDS Custom オートメーションを一時停止する

移行手順を進める前に、RDS Custom インスタンスでオートメーションモードを一時停止して、オートメーションが RMAN アクティビティに干渉しないようにします。--resume-full-automation-mode-minute パラメータ (この例では 240 分 = 4 時間) は、データベースのサイズと予想される複製時間に基づいて調整する必要があります。

重要: オートメーションを一時停止しても、データベースの可用性には影響しません。データベースはオンラインのままで、複製処理中もアプリケーションからアクセスできます。

Example

```
aws rds modify-db-instance \
--db-instance-identifier my-custom-instance \
--automation-mode all-paused \
--resume-full-automation-mode-minute 240 \
```

```
--region us-east-1
--query 'DBInstances[0].AutomationMode'
```

オートメーションステータスを検証します。

Example

```
aws ds describe-db-instances \
--db-instance-identifier my-custom-instance \
--region us-east-1 \
--query 'DBInstances[0].AutomationMode'
```

正常な出力:「all-paused」

ステップ 2: パスワードとパラメータファイルを作成する

orapwd を使用してパスワードファイルを作成します。AWS Secrets Manager から SYS パスワードを取得します (RDS Custom インスタンスの作成時に保存されたもの)。

Example

```
# Non-CDB
$ORACLE_HOME/bin/orapwd file=/rdsdbdata/config/orapwORCL password=<SYS_PASSWORD>
 entries=10
# Multitenant
$ORACLE_HOME/bin/orapwd file=/rdsdbdata/config/orapwRDSCDB password=<SYS_PASSWORD>
 entries=10
```

ソースデータベースからパラメータファイルを作成します。

Example

```
sqlplus / as sysdba
SQL> CREATE PFILE='/tmp/initORCL.ora' FROM SPFILE; -- Non-CDB
SQL> CREATE PFILE='/tmp/initRDSCDB.ora' FROM SPFILE; -- Multitenant
```

生成されたパラメータファイルには、RDS Custom 固有のパスとパラメータが含まれます。マルチテナントの場合、キーパラメータには以下が含まれます。

- enable_pluggable_database=TRUE (マルチテナントにとって重要)
- noncdb_compatible=TRUE (下位互換性のため)

- CDB$ROOT、PDB$SEED およびすべての PDB のデータファイルパス
- OMF 用の db_create_file_dest および db_create_online_log_dest_1

ステップ 3: EC2 を使用してファイルを転送する

任意のファイル転送方法を選択します。このガイダンスでは、Amazon S3 を例として使用します。

Amazon S3 の使用。

Amazon S3 の使用。

Example非 CDB の場合:

```
# Copy files to Amazon S3 from the RDS Custom instance
aws s3 cp /tmp/initORCL.ora s3://<YOUR_BUCKET>/
aws s3 cp /rdsdbdata/config/orapwORCL s3://<YOUR_BUCKET>/
# On EC2, download files
aws s3 cp s3://<YOUR_BUCKET>/initORCL.ora $ORACLE_HOME/dbs/
aws s3 cp s3://<YOUR_BUCKET>/orapwORCL $ORACLE_HOME/dbs/
```

Exampleマルチテナントの場合

```
# Copy files to Amazon S3 from the RDS Custom instance
aws s3 cp /tmp/initRDSCDB.ora s3://<YOUR_BUCKET>/
aws s3 cp /rdsdbdata/config/orapwRDSCDB s3://<YOUR_BUCKET>/
# On EC2, download and rename files to use ORCL naming
aws s3 cp s3://<YOUR_BUCKET>/initRDSCDB.ora $ORACLE_HOME/dbs/initORCL.ora
aws s3 cp s3://<YOUR_BUCKET>/orapwRDSCDB $ORACLE_HOME/dbs/orapwORCL
```

EC2 のファイルを検証します。

Example

```
ls -l $ORACLE_HOME/dbs/initORCL.ora
ls -l $ORACLE_HOME/dbs/orapwORCL
```

ステップ 4: EC2 でパラメータファイルを編集する

移行を成功させるためには、パラメータファイルを慎重に編集する必要があります。これは、移行プロセスにおける最も重要なステップの 1 つです。

EC2 インスタンスで $ORACLE_HOME/dbs/initORCL.ora を編集し、以下の重要な変更を加えます。

1. データベース名の更新: マルチテナントの場合は、RDSCDB のすべての出現箇所を ORCL に変更する
2. RDS Custom パスを EC2 パスに変換する: /rdsdbdata/ を /u01/app/oracle/ に置き換える
3. RDS Custom 固有のパラメータを削除する
   - dg_broker_config_file1 および dg_broker_config_file2 (存在する場合 - レプリカが存在することを示します)
   - standby_file_management (存在する場合)
   - spfile (後で新しい SPFILE を作成します)
   - スタンバイ送信先を指す log_archive_dest パラメータ (ローカルアーカイブの場合は log_archive_dest_1 のみを保持)
4. ファイル名変換パラメータを追加する (以下を参照)
5. EC2 インスタンスのサイズに基づいてメモリパラメータを調整する (オプションですが推奨)

ファイル名変換パラメータについて。

DB_FILE_NAME_CONVERT および LOG_FILE_NAME_CONVERT パラメータは、RMAN 複製に不可欠です。これらは、複製プロセス中にソースファイルパスをターゲットファイルパスにマッピングする方法を RMAN に伝えます。これらのパラメータがないと、RMAN はソースと同じパスにファイルを作成しようとしますが、EC2 では失敗します。

主な考慮事項:

- 各パラメータは文字列のペアを受け入れます。ソースパスの後にターゲットパスが続きます
- 1 つのパラメータで複数のペアを指定できます
- マルチテナントの場合、CDB$ROOT、PDB$SEED およびすべての PDB のパスをマッピングする必要があります
- ペアの順序は重要です - RMAN は指定された順序でペアを処理します
- パスでは大文字と小文字が区別され、正確に一致する必要があります

パスマッピング:

非 CDB:

RDS Custom パス	EC2 パス	説明
/rdsdbbin	/u01/app/oracle	Oracle ベース
/rdsdbdata/db/ORCL_A/datafile/	/u01/app/oracle/oradata/ORCL/datafile/	データファイル
/rdsdbdata/db/ORCL_A/controlfile/	/u01/app/oracle/oradata/ORCL/controlfile/	コントロールファイル
/rdsdbdata/db/ORCL_A/onlinelog/	/u01/app/oracle/oradata/ORCL/onlinelog/	オンライン REDO ログ
/rdsdbdata/db/ORCL_A/arch/	/u01/app/oracle/oradata/ORCL/arch/	ログのアーカイブ
/rdsdbdata/admin/ORCL/adump	/u01/app/oracle/admin/ORCL/adump	監査ダンプ
/rdsdbdata/log	/u01/app/oracle	診断の送信先

マルチテナント

RDS Custom パス	EC2 パス	説明
/rdsdbbin	/u01/app/oracle	Oracle ベース
/rdsdbdata/db/cdb/RDSCDB/datafile/	/u01/app/oracle/oradata/ORCL/cdb/datafile/	CDB ルートデータファイル
/rdsdbdata/db/cdb/pdbseed/	/u01/app/oracle/oradata/ORCL/pdbseed/datafile/	PDB$SEED データファイル

RDS Custom パス	EC2 パス	説明
/rdsdbdata/db/pdb/RDSCDB_A/	/u01/app/oracle/oradata/ORCL/pdb/datafile/	PDB データファイル (GUID 付き OMF)
/rdsdbdata/db/cdb/RDSCDB_A/controlfile/	/u01/app/oracle/oradata/ORCL/controlfile/	コントロールファイル
/rdsdbdata/db/cdb/RDSCDB_A/onlinelog/	/u01/app/oracle/oradata/ORCL/onlinelog/	オンライン REDO ログ
/rdsdbdata/db/cdb/RDSCDB_A/arch/redolog	/u01/app/oracle/oradata/ORCL/arch/	ログのアーカイブ
/rdsdbdata/admin/RDSCDB/adump	/u01/app/oracle/admin/ORCL/adump	監査ダンプ
/rdsdbdata/log	/u01/app/oracle	診断の送信先

変換パラメータを追加する。

Example非 CDB:

```
*.DB_FILE_NAME_CONVERT='/rdsdbdata/db/ORCL_A/datafile/','/u01/app/oracle/oradata/ORCL/datafile/'
*.LOG_FILE_NAME_CONVERT='/rdsdbdata/db/ORCL_A/onlinelog/','/u01/app/oracle/oradata/ORCL/onlinelog/'
```

Exampleマルチテナント (CDB ルート、PDB$SEED、およびすべての PDB パスのマッピングを含める必要があります)。

```
*.DB_FILE_NAME_CONVERT='/rdsdbdata/db/cdb/RDSCDB/datafile/','/u01/app/oracle/oradata/ORCL/cdb/datafile/','/rdsdbdata/db/cdb/pdbseed/','/u01/app/oracle/oradata/ORCL/pdbseed/datafile/','/rdsdbdata/db/pdb/RDSCDB_A/','/u01/app/oracle/oradata/ORCL/pdb/datafile/'
```

```
 *.LOG_FILE_NAME_CONVERT='/rdsdbdata/db/cdb/RDSCDB_A/onlinelog/','/u01/app/oracle/
oradata/ORCL/onlinelog/'
```

重要: マルチテナントの場合は、パラメータファイルに enable_pluggable_database=TRUE が存在することを確認します。RDS Custom は、GUID ベースのサブディレクトリを持つ PDB データファイルに OMF を使用します。これはパスマッピングによって自動的に処理されます。

完全なパラメータファイルの例。

Example非 CDB パラメータファイルの例 (initORCL.ora)。

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=6039797760
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=0
ORCL.__large_pool_size=33554432
ORCL.__oracle_base='/u01/app/oracle'
ORCL.__pga_aggregate_target=4966055936
ORCL.__sga_target=7449083904
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1207959552
ORCL.__streams_pool_size=0
ORCL.__unified_pga_pool_size=0
*.archive_lag_target=300
*.audit_file_dest='/u01/app/oracle/admin/ORCL/adump'
*.compatible='19.0.0'
*.control_files='/u01/app/oracle/oradata/ORCL/controlfile/control-01.ctl'
*.db_block_checking='MEDIUM'
*.db_create_file_dest='/u01/app/oracle/oradata/ORCL'
*.db_name='ORCL'
*.db_recovery_file_dest_size=1073741824
*.db_unique_name='ORCL'
*.dbfips_140=FALSE
*.diagnostic_dest='/u01/app/oracle'
*.filesystemio_options='setall'
*.heat_map='OFF'
*.job_queue_processes=50
*.local_listener='(address=(protocol=tcp)(host=)(port=1521))'
*.log_archive_dest_1='location="/u01/app/oracle/oradata/ORCL/arch",
 valid_for=(ALL_LOGFILES,ALL_ROLES)'
*.log_archive_format='-%s-%t-%r.arc'
*.max_string_size='STANDARD'
```

```
*.memory_max_target=12385852416
*.memory_target=12385852416
*.open_cursors=300
*.pga_aggregate_target=0
*.processes=1673
*.recyclebin='OFF'
*.sga_target=0
*.undo_tablespace='UNDO_T1'
*.use_large_pages='FALSE'
*.DB_FILE_NAME_CONVERT='/rdsdbdata/db/ORCL_A/datafile/','/u01/app/oracle/oradata/ORCL/
datafile/'
*.LOG_FILE_NAME_CONVERT='/rdsdbdata/db/ORCL_A/onlinelog/','/u01/app/oracle/oradata/
ORCL/onlinelog/'
```

Exampleマルチテナントパラメータファイルの例 (initORCL.ora)。

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=6006243328
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=0
ORCL.__large_pool_size=33554432
ORCL.__oracle_base='/u01/app/oracle'
ORCL.__pga_aggregate_target=4966055936
ORCL.__sga_target=7415529472
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1207959552
ORCL.__streams_pool_size=0
ORCL.__unified_pga_pool_size=0
*.archive_lag_target=300
*.audit_file_dest='/u01/app/oracle/admin/ORCL/adump'
*.compatible='19.0.0'
*.control_files='/u01/app/oracle/oradata/ORCL/controlfile/control-01.ctl'
*.db_block_checking='MEDIUM'
*.db_create_file_dest='/u01/app/oracle/oradata/ORCL/pdb/datafile'
*.db_create_online_log_dest_1='/u01/app/oracle/oradata/ORCL'
*.db_name='ORCL'
*.db_recovery_file_dest_size=1073741824
*.db_unique_name='ORCL'
*.dbfips_140=FALSE
*.diagnostic_dest='/u01/app/oracle'
*.enable_pluggable_database=TRUE
*.filesystemio_options='setall'
```

```
*.heat_map='OFF'
*.job_queue_processes=50
*.local_listener='(address=(protocol=tcp)(host=)(port=1521))'
*.log_archive_dest_1='location="/u01/app/oracle/oradata/ORCL/arch",
 valid_for=(ALL_LOGFILES,ALL_ROLES)'
*.log_archive_format='-%s-%t-%r.arc'
*.max_string_size='STANDARD'
*.memory_max_target=12361688064
*.memory_target=12361688064
*.noncdb_compatible=TRUE
*.open_cursors=300
*.pga_aggregate_target=0
*.processes=1670
*.recyclebin='OFF'
*.sga_target=0
*.undo_tablespace='UNDO_T1'
*.use_large_pages='FALSE'
*.DB_FILE_NAME_CONVERT='/rdsdbdata/db/cdb/RDSCDB/datafile/','/u01/app/oracle/oradata/
ORCL/cdb/datafile/','/rdsdbdata/db/cdb/pdbseed/','/u01/app/oracle/oradata/ORCL/pdbseed/
datafile/','/rdsdbdata/db/pdb/RDSCDB_A/','/u01/app/oracle/oradata/ORCL/pdb/datafile/'
*.LOG_FILE_NAME_CONVERT='/rdsdbdata/db/cdb/RDSCDB_A/onlinelog/','/u01/app/oracle/
oradata/ORCL/onlinelog/'
```

キーパラメータの説明。

- enable_pluggable_database=TRUE: マルチテナント CDB に必要です。このパラメータは、プラガブルデータベース機能を有効にします。
- noncdb_compatible=TRUE: 下位互換性のために RDS Custom で設定します。要件に基づいて保持または削除できます。
- db_create_file_dest: Oracle Managed Files (OMF) のデフォルトの場所を指定します。マルチテナントの場合、これは PDB データファイルディレクトリを指します。
- db_create_online_log_dest_1: OMF を使用するときのオンライン REDO ログの場所を指定します。
- DB_FILE_NAME_CONVERT: RMAN 複製に不可欠です。ソースデータファイルパスをターゲットパスにマッピングします。
- LOG_FILE_NAME_CONVERT: ソース REDO ログパスをターゲットパスにマッピングします。
- memory_target および memory_max_target: EC2 インスタンスのメモリに基づいてこれらを調整します。表示されている値は例です。

- `processes`: Oracle に接続できるオペレーティングシステムプロセスの最大数。ワークロードに基づいて調整します。

メモリサイズ設定のガイドライン。

EC2 インスタンスのメモリパラメータを調整する場合。

EC2 インスタンスメモリ	推奨 memory_target	推奨 memory_max_target
16 GB	12 GB (12884901888 バイト)	14 GB (15032385536 バイト)
32 GB	24 GB (25769803776 バイト)	28 GB (30064771072 バイト)
64 GB	48 GB (51539607552 バイト)	56 GB (60129542144 バイト)
128 GB	96 GB (103079215104 バイト)	112 GB (120259084288 バイト)

システムメモリの約 20～25% をオペレーティングシステムやその他のプロセス用に残します。

ステップ 5: TNS とリスナーを設定する

両方のインスタンスで、`tnsnames.ora` に TNS エントリを に追加します。

Example非 CDB:

```
DB_SOURCE = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <RDS_CUSTOM_IP>)(PORT =
 1521)) (CONNECT_DATA = (SID = ORCL)))
DB_TARGET = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <EC2_IP>)(PORT = 1521))
 (CONNECT_DATA = (SID = ORCL)))
```

Exampleマルチテナント。

```
DB_SOURCE = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <RDS_CUSTOM_IP>)(PORT =
 1521)) (CONNECT_DATA = (SID = RDSCDB)))
DB_TARGET = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <EC2_IP>)(PORT = 1521))
 (CONNECT_DATA = (SID = ORCL)))
```

Example EC2 でリスナーを設定します (**$ORACLE_HOME/network/admin/listener.ora**)。

```
LISTENER = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <EC2_IP>)(PORT = 1521)))
SID_LIST_LISTENER = (SID_LIST = (SID_DESC = (GLOBAL_DBNAME = ORCL) (ORACLE_HOME = /u01/
app/oracle/product/19.0.0/dbhome_1) (SID_NAME = ORCL)))
```

Exampleリスナーを起動します。

```
lsnrctl start
```

> **Note**
>
> RMAN アクティブ複製の場合、TNS エントリは SID (SERVICE_NAME ではない) を使用してデータベースインスタンスに接続します。マルチテナントの場合、これは CDB に接続され、RMAN はすべての PDB を自動的に複製します。

ステップ 6: EC2 でデータベースを **NOMOUNT** で起動する

Example

```
export ORACLE_SID=ORCL
export ORACLE_HOME=/u01/app/oracle/product/19.0.0/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH

sqlplus / as sysdba

SQL> STARTUP NOMOUNT PFILE='$ORACLE_HOME/dbs/initORCL.ora';
```

Exampleパラメータを確認する。

```
SQL> SHOW PARAMETER db_name
SQL> SHOW PARAMETER control_files
SQL> SHOW PARAMETER enable_pluggable_database -- For multitenant
```

ステップ 7: RMAN アクティブ複製を実行する

RMAN アクティブ複製は、稼働中のソースからターゲットにデータベースをコピーします。ソースデータベースはオンラインのままで、このプロセス全体を通してアプリケーションからアクセスできます。

RMAN をソースインスタンスとターゲットインスタンスの両方に接続します。

Example

```
rman target sys/<password>@DB_SOURCE auxiliary sys/<password>@DB_TARGET
```

Example非 CDB の場合の予想される出力。

```
Recovery Manager: Release 19.0.0.0.0 - Production
connected to target database: ORCL (DBID=4089929259)
connected to auxiliary database: ORCL (not mounted)
```

Exampleマルチテナントの予想される出力。

```
Recovery Manager: Release 19.0.0.0.0 - Production
connected to target database: RDSCDB (DBID=4089929259)
connected to auxiliary database: ORCL (not mounted)
```

Exampleアクティブ複製コマンドを実行します。

```
RMAN> DUPLICATE DATABASE TO 'ORCL' FROM ACTIVE DATABASE NOFILENAMECHECK;
```

Note

- ソースはオンラインのまま: ソースデータベースは、複製している間もアプリケーションリクエストを処理し続けます
- 非 CDB の場合: オンラインのままデータベース全体を複製します
- マルチテナントの場合: ソースがオンラインのまま、CDB$ROOT、PDB$SEED、およびすべての PDB を含む CDB 全体が 1 回のオペレーションで複製されます。
- NOFILENAMECHECK: ファイル名がソースとターゲットで異なっていても RMAN が続行できるようにします
- 期間: データベースのサイズとネットワーク帯域幅によって異なります。100GB のデータベースの場合、30～60 分かかります
- ネットワークへの影響: 複製中はネットワーク帯域幅の使用量が多くなりますが、ソースデータベースには引き続きアクセスできます

RMAN アクティブ複製プロセスには、いくつかのフェーズが含まれます。

1. ソースおよびターゲットデータベースに接続する
2. ターゲットのメモリから SPFILE を作成する
3. コントロールファイルをターゲットに復元する
4. ターゲットデータベースをマウントする
5. ネットワーク経由でソースからターゲットにすべてのデータファイルをコピーする (ソースはオンラインのまま)
6. ターゲットデータベースを復旧する
7. RESETLOGS を使用してターゲットデータベースを開く

複製中、ソースデータベースは次の操作を行います。

- READ WRITE モードのまま維持される
- 接続を引き続き受け入れる
- トランザクションを通常どおり処理する
- 通常どおり REDO ログを生成する
- アプリケーションから完全にアクセスできる

Example別のセッションで進行状況をモニタリングします。

```
SQL> SELECT sid, serial#, sofar, totalwork, ROUND(sofar/totalwork*100,2) pct_complete
FROM v$session_longops WHERE totalwork > 0 AND sofar <> totalwork;
```

RMAN 複製時の詳細なモニタリングとトラブルシューティング:

RMAN 複製の実行中に、いくつかの方法を使用して進行状況をモニタリングできます。

1. RMAN 出力をモニタリングします。

   RMAN セッションには、次のような詳細な進行状況情報が表示されます。

   - チャネル割り当て
   - データファイルのコピーの進行状況
   - 推定完了時間
   - 処理されたバイト数

```
channel ORA_AUX_DISK_1: starting datafile copy
input datafile file number=00001 name=/rdsdbdata/db/ORCL_A/datafile/system01.dbf
output file name=/u01/app/oracle/oradata/ORCL/datafile/system01.dbf
 tag=TAG20260303T120000
channel ORA_AUX_DISK_1: datafile copy complete, elapsed time: 00:05:23
channel ORA_AUX_DISK_1: starting datafile copy
input datafile file number=00003 name=/rdsdbdata/db/ORCL_A/datafile/sysaux01.dbf
output file name=/u01/app/oracle/oradata/ORCL/datafile/sysaux01.dbf
 tag=TAG20260303T120000
```

Example出力例:

```
channel ORA_AUX_DISK_1: starting datafile copy
input datafile file number=00001 name=/rdsdbdata/db/ORCL_A/datafile/system01.dbf
output file name=/u01/app/oracle/oradata/ORCL/datafile/system01.dbf
 tag=TAG20260303T120000
channel ORA_AUX_DISK_1: datafile copy complete, elapsed time: 00:05:23
channel ORA_AUX_DISK_1: starting datafile copy
input datafile file number=00003 name=/rdsdbdata/db/ORCL_A/datafile/sysaux01.dbf
output file name=/u01/app/oracle/oradata/ORCL/datafile/sysaux01.dbf
 tag=TAG20260303T120000
```

2. 長時間実行されるオペレーションをモニタリングします。

Exampleターゲット EC2 インスタンスの別の SQL*Plus セッションでの実行:

```
SQL> SELECT sid, serial#, opname, sofar, totalwork,
       ROUND(sofar/totalwork*100,2) pct_complete,
       time_remaining, elapsed_seconds
FROM v$session_longops
WHERE totalwork > 0 AND sofar <> totalwork
ORDER BY start_time;
```

このクエリは以下を表示します。

- opname: オペレーション名 (「RMAN: full datafile restore」など)
- sofar: これまでに処理されたブロック
- totalwork: 処理する合計ブロック数
- pct_complete: 完了率
- time_remaining: 推定残り秒数

- elapsed_seconds: これまでに経過した時間

3. RMAN チャネルをモニタリングします。

   Example

```
SQL> SELECT sid, serial#, context, sofar, totalwork,
       ROUND(sofar/totalwork*100,2) pct_complete
       FROM v$session_longops
WHERE opname LIKE 'RMAN%'
       AND totalwork > 0
       AND sofar <> totalwork;
```

4. アラートログを確認します。

   Exampleターゲット EC2 インスタンスで、エラーや警告がないかアラートログをモニタリングします。

```
tail -f $ORACLE_BASE/diag/rdbms/orcl/ORCL/trace/alert_ORCL.log
```

   RMAN 複製時の一般的な問題。

   - ネットワークタイムアウト

```
RMAN-03009: failure of duplicate command on ORA_AUX_DISK_1 channel
ORA-03135: connection lost contact
```

     解決策: 両方のインスタンスの sqlnet.ora でネットワークタイムアウト値を増やします。

```
SQLNET.RECV_TIMEOUT=600
SQLNET.SEND_TIMEOUT=600
```

   - ターゲットのスペースが不十分

```
RMAN-03009: failure of duplicate command
ORA-19504: failed to create file "/u01/app/oracle/oradata/ORCL/datafile/
users01.dbf"
ORA-27040: file create error, unable to create file
Linux-x86_64 Error: 28: No space left on device
```

     解決策: 使用可能なスペースを確認し、EBS ボリューム容量を追加します。

```
df -h /u01/app/oracle/oradata
```

- パラメータファイルのエラー

```
RMAN-05021: this configuration cannot be used
RMAN-06004: ORACLE error from auxiliary database: ORA-01261: Parameter
 db_create_file_dest destination string cannot be translated
```

解決策: パラメータファイル内のすべてのディレクトリが存在し、適切なアクセス許可を持っていることを確認します。

- パスワードファイルの不一致

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of duplicate command at 03/03/2026 12:00:00
RMAN-06136: ORACLE error from auxiliary database: ORA-01017: invalid username/
password; logon denied
```

解決策: ターゲットのパスワードファイルがソースと一致し、正しい名前 (orapwORCL) であることを確認します。

ステップ 8: PDB を開く (マルチテナントのみ)

RMAN 複製が完了すると、EC2 の CDB は READ WRITE モードで開きますが、すべての PDB は MOUNTED 状態になります。これは想定される動作です。手動で開く必要があります。

現在の PDB ステータスを確認します。

```
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
```

予想される出力 (ORCLDB という名前の PDB が 1 つある場合の例)。

```
CON_ID     NAME                           OPEN_MODE
---------- ------------------------------ ----------
2          PDB$SEED                       READ ONLY
3          ORCLDB                         MOUNTED
```

すべての PDB を開きます。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
Pluggable database altered.
```

すべての PDB が READ WRITE モードでオープンされていることを確認します。

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
```

正常な出力:

```
CON_ID     NAME                           OPEN_MODE  RES
---------- ------------------------------ ---------- ---
2          PDB$SEED                       READ ONLY  NO
3          ORCLDB                         READ WRITE NO
```

状態保存メソッド (Oracle 19c に推奨) を使用して、起動時に自動オープンを設定します。

```
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
Pluggable database altered.
```

これは、Oracle に対し、PDB の現在のオープン状態を記憶し、CDB の起動時にその状態を復元するように指示します。

保存された状態を確認します。

```
SQL> SELECT con_name, instance_name, state FROM dba_pdb_saved_states;
```

PDB サービスがリスナーに登録されていることを確認します。

```
lsnrctl services
```

予想される出力には、CDB と各 PDB のサービスが表示されるはずです。サービスが表示されない場合:

```
SQL> ALTER SYSTEM REGISTER;
```

次に、`lsnrctl services` でもう一度確認します。

ステップ 9: RDS Custom オブジェクトを削除する

これは EC2 上のセルフマネージドデータベースになったため、RDS Custom 固有のユーザーとオブジェクトを削除する必要があります。クリーンアッププロセスは、非 CDB アーキテクチャとマルチテナントアーキテクチャで異なります。

> **Important**
>
> RDS 固有のユーザーとテーブルスペースを削除する前に、これらのスキーマの下にアプリケーションオブジェクトが存在しないことを確認します。

```
SQL> SELECT owner, object_type, COUNT(*)
FROM dba_objects
WHERE owner IN ('RDSADMIN', 'RDS_DATAGUARD')
  AND object_name NOT LIKE 'RDS%'
  AND object_name NOT LIKE 'SYS_%'
GROUP BY owner, object_type;
```

アプリケーションオブジェクトが見つかった場合は、先に進む前に適切なアプリケーションスキーマに移行します。

Example非 CDB:

```
SQL> DROP USER RDSADMIN CASCADE;
SQL> DROP USER RDS_DATAGUARD CASCADE;
SQL> DROP DIRECTORY OPATCH_INST_DIR;
SQL> DROP DIRECTORY OPATCH_LOG_DIR;
SQL> DROP DIRECTORY OPATCH_SCRIPT_DIR;
SQL> DROP TABLESPACE RDSADMIN INCLUDING CONTENTS AND DATAFILES;

-- Verify removal
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';
```

正常な出力: `no rows selected`

マルチテナント:

マルチテナント環境では、RDS Custom はすべての PDB から参照可能な共通ユーザーを CDB$ROOT に作成します。CDB$ROOT からクリーンアップする必要があります。

```
-- Connect to CDB$ROOT
SQL> SHOW CON_NAME;
-- Check for RDS-specific common users (including C## prefixed users)
SQL> SELECT username, common, con_id FROM cdb_users
WHERE username LIKE '%RDS%' OR username LIKE 'C##RDS%'
ORDER BY username;

-- Drop non-common users
SQL> DROP USER RDSADMIN CASCADE;
SQL> DROP USER RDS_DATAGUARD CASCADE;

-- If any C## common users exist
-- SQL> DROP USER C##RDSADMIN CASCADE
;
-- Drop directories and tablespace
SQL> DROP DIRECTORY OPATCH_INST_DIR;
SQL> DROP DIRECTORY OPATCH_LOG_DIR;
SQL> DROP DIRECTORY OPATCH_SCRIPT_DIR;
SQL> DROP TABLESPACE RDSADMIN INCLUDING CONTENTS AND DATAFILES;

-- Verify removal from CDB$ROOT
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';

-- Verify removal from each PDB
SQL> ALTER SESSION SET CONTAINER = <PDB_NAME>;
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';
```

すべてのクエリで予想される出力: `no rows selected`

ステップ 10: 自動起動を設定する

SPFILE を作成します。

```
SQL> CREATE SPFILE FROM PFILE='$ORACLE_HOME/dbs/initORCL.ora';
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

マルチテナントの場合、PDB が開いていることを確認します。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
```

/etc/oratab を編集します。

```
ORCL:/u01/app/oracle/product/19.0.0/dbhome_1:Y
```

ステップ 11: 最終検証

Example非 CDB:

```
SQL> SELECT name, open_mode, log_mode FROM v$database;
SQL> SELECT SUM(bytes)/1024/1024/1024 AS size_gb FROM dba_data_files;
SQL> SELECT COUNT(*) FROM dba_objects WHERE status = 'INVALID';
```

Exampleマルチテナント:

```
SQL> SELECT name, open_mode, log_mode, cdb FROM v$database;
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
SQL> SELECT SUM(bytes)/1024/1024/1024 AS total_size_gb FROM cdb_data_files;
SQL> SELECT con_id, COUNT(*) FROM cdb_objects WHERE status = 'INVALID' GROUP BY con_id;
```

```
Test application connectivity:
# Non-CDB
sqlplus <app_user>/<password>@<EC2_IP>:1521/ORCL

# Multitenant (connect to PDB)
sqlplus <app_user>/<password>@<EC2_IP>:1521/<PDB_NAME>
```

ステップ 12: RDS Custom オートメーションを再開する

```
aws rds modify-db-instance \
--db-instance-identifier my-custom-instance \
--automation-mode full \
--region us-east-1
```

オプション 2: Oracle Data Guard を使用した物理移行

このセクションでは、Oracle Data Guard を使用して Oracle データベースを RDS Custom for Oracle から EC2 に移行する詳細な手順について説明します。この方法は、最小限のダウンタイムを必要とする移行に適しています。

Oracle Data Guard を使用するタイミング

次の場合に Oracle Data Guard を選択します。

- 最小限のダウンタイム (スイッチオーバーに数秒から数分) が必要
- 本番稼働用データベースまたはミッションクリティカルなデータベースを移行する場合
- カットオーバーの前に継続的な同期が必要
- 組み込みのフォールバック機能が必要
- 移行にコミットする前に、ターゲットデータベースをテストする必要がある

Oracle Data Guard の概要

Oracle Data Guard は、プライマリデータベースから REDO ログを継続的に配布して適用することで、同期されたスタンバイデータベースを維持します。移行を完了する準備ができたら、最小限のダウンタイム (数秒から数分) で EC2 スタンバイをプライマリに昇格させるスイッチオーバーを実行します。マルチテナントデータベースの場合、Data Guard はすべての PDB を含む CDB 全体を自動的に保護します。任意の PDB によって生成された REDO はスタンバイに配布され、スタンバイの対応する PDB に適用されます。

Oracle Data Guard の移行ワークフロー

RDS Custom DB インスタンス (プライマリ) のステップ。

1. Amazon RDS Custom オートメーションを一時停止する
2. データベースアーキテクチャを検証する (非 CDB または PDB を含む CDB)
3. プライマリデータベースがアーカイブログモードで実行され、`FORCE_LOGGING` が有効になっていることを確認する
4. パスワードファイルを作成する
5. プライマリデータベースの RMAN オンラインバックアップを実行する (またはアクティブ複製を使用する)
6. ソースデータベースからパラメータファイルを作成する
7. バックアップセット、パラメータファイル、パスワードファイルをターゲット EC2 インスタンスにコピーする

EC2 DB インスタンス (スタンバイ) のステップ:

1. ソースから EC2 インスタンスにすべてのファイルをコピーする

2. パスワードファイルを EC2 インスタンスにコピーする
3. EC2 のパラメータファイルを編集する (アーキテクチャ固有)
4. パラメータファイルからサーバーパラメータファイルを作成する
5. スタンバイコントロールファイルとデータベースを復元する

Data Guard 設定のステップ:

1. 両方のインスタンスで Oracle リスナーを設定する
2. 両方のインスタンスで tnsnames.ora を設定する
3. 両方のインスタンスで Oracle Data Guard ブローカーを起動する (オプションですが推奨)
4. Oracle Data Guard 設定を有効にする
5. EC2 スタンバイインスタンスで fal_server と fal_client を設定する
6. 両方のインスタンスでスタンバイ REDO ログファイルを設定する
7. スタンバイインスタンスを復旧する
8. 手動スイッチオーバーを実行する
9. データベースを (マルチテナントの場合は PDB も) 開く
10. PDB の自動オープンを設定する (マルチテナントのみ)
11. RDS Custom 固有のユーザーとオブジェクトを削除する
12. SPFILE を作成し、自動起動を設定する

ステップ 1: Amazon RDS Custom オートメーションを一時停止する

RDS Custom インスタンスのオートメーションモードを一時停止します。`--resume-full-automation-mode-minute` パラメータは、データベースのサイズと Data Guard の予想セットアップ時間に基づいて調整する必要があります。

```
aws rds modify-db-instance \
  --db-instance-identifier my-custom-instance \
  --automation-mode all-paused \
  --resume-full-automation-mode-minute 480 \
  --region us-east-1
```

オートメーションステータスを検証します。

```
aws rds describe-db-instances \
```

```
  --db-instance-identifier my-custom-instance \
  --region us-east-1 \
  --query 'DBInstances[0].AutomationMode'
```

正常な出力:「all-paused」

ステップ 2: アーカイブログモードと **FORCE_LOGGING**

Oracle Data Guard では、プライマリデータベースがアーカイブログモードで、強制ログが有効になっている必要があります。

```
sqlplus / as sysdba
SQL> SELECT log_mode, force_logging FROM v$database;
```

正常な出力:

```
LOG_MODE     FORCE_LOGGING
------------ -------------
ARCHIVELOG   YES
```

強制ログ記録が有効になっていない場合は、以下を実行します。

```
SQL> ALTER DATABASE FORCE LOGGING;
```

ステップ 3: パスワードとパラメータファイルを作成する

orapwd を使用してパスワードファイルを作成します。AWS Secrets Manager から SYS パスワードを取得します。

```
# Non-CDB
$ORACLE_HOME/bin/orapwd file=/rdsdbdata/config/orapwORCL password=<SYS_PASSWORD>
 entries=10

# Multitenant
$ORACLE_HOME/bin/orapwd file=/rdsdbdata/config/orapwRDSCDB password=<SYS_PASSWORD>
 entries=10
```

ソースデータベースからパラメータファイルを作成します。

```
sqlplus / as sysdba
```

```
SQL> CREATE PFILE='/tmp/initORCL.ora' FROM SPFILE; -- Non-CDB
SQL> CREATE PFILE='/tmp/initRDSCDB.ora' FROM SPFILE; -- Multitenant
```

ステップ 4: RMAN バックアップを実行するか、アクティブ複製を使用する

スタンバイデータベースを作成するには、次の 2 つのオプションがあります。

オプション A: RMAN オンラインバックアップ (ほとんどのシナリオで推奨)

バックアップディレクトリを作成し、データベースをバックアップします。

```
mkdir -p /rdsdbdata/backup
rman target /
RMAN> run {
  backup as compressed backupset
  filesperset 2
  format '/rdsdbdata/backup/db_%U'
  database;
  sql 'alter system archive log current';
  backup as compressed backupset
  filesperset 50
  format '/rdsdbdata/backup/arch_%U'
  archivelog all;
}

RMAN> backup current controlfile for standby format '/rdsdbdata/backup/standby.ctl';
```

> **Note**
>
> マルチテナントの場合、データベースキーワードは、すべての PDB を含む CDB 全体を自動的にバックアップします。

オプション B: アクティブ複製 (代替方法)

バックアップステップをスキップし、RMAN アクティブ複製を使用してスタンバイをネットワーク経由で直接構築します。これにより、バックアップストレージやファイル転送が不要になります。TNS とリスナーを設定した後 (以下を参照)、以下を実行します。

```
RMAN> DUPLICATE TARGET DATABASE FOR STANDBY FROM ACTIVE DATABASE DORECOVER;
```

このガイダンスでは、オプション A (バックアップベース) に焦点を当てていますが、オプション B も有効な代替手段です。

ステップ 5: EC2 を使用してファイルを転送する

任意のファイル転送方法を選択します。このガイダンスでは、Amazon S3 を例として使用します。

Amazon S3 の使用。

Example非 CDB の場合:

```
# Copy files to Amazon S3 from the RDS Custom instance
aws s3 cp /rdsdbdata/backup s3://<YOUR_BUCKET>/ --recursive
aws s3 cp /tmp/initORCL.ora s3://<YOUR_BUCKET>/
aws s3 cp /rdsdbdata/config/orapwORCL s3://<YOUR_BUCKET>/

# On EC2, download files
aws s3 cp s3://<YOUR_BUCKET>/backup /u01/app/oracle/backup/ --recursive
aws s3 cp s3://<YOUR_BUCKET>/initORCL.ora $ORACLE_HOME/dbs/
aws s3 cp s3://<YOUR_BUCKET>/orapwORCL $ORACLE_HOME/dbs/
```

Exampleマルチテナントの場合:

```
# Copy files to Amazon S3 from the RDS Custom instance
aws s3 cp /rdsdbdata/backup s3://<YOUR_BUCKET>/ --recursive
aws s3 cp /tmp/initRDSCDB.ora s3://<YOUR_BUCKET>/
aws s3 cp /rdsdbdata/config/orapwRDSCDB s3://<YOUR_BUCKET>/

# On EC2, download and rename files to use ORCL naming
aws s3 cp s3://<YOUR_BUCKET>/backup /u01/app/oracle/backup/ --recursive
aws s3 cp s3://<YOUR_BUCKET>/initRDSCDB.ora $ORACLE_HOME/dbs/initORCL.ora
aws s3 cp s3://<YOUR_BUCKET>/orapwRDSCDB $ORACLE_HOME/dbs/orapwORCL
```

ステップ 6: EC2 でパラメータファイルを編集する

EC2 インスタンスで `$ORACLE_HOME/dbs/initORCL.ora` を編集し、以下の重要な変更を加えます。

1. データベース名の更新: マルチテナントの場合は、RDSCDB のすべての出現箇所を ORCL に変更する
2. db_unique_name の変更: ORCL_A (または RDSCDB_A) から ORCL_B に変更
3. RDS Custom パスを EC2 パスに変換する: `/rdsdbdata/` を `/u01/app/oracle/` に置き換える

4. RDS Custom 固有のパラメータを削除する
   - dg_broker_config_file1 および dg_broker_config_file2 (存在する場合)
   - standby_file_management (存在する場合)
   - spfile (後で新しい SPFILE を作成します)
   - スタンバイ送信先を指す log_archive_dest パラメータ
5. EC2 インスタンスのサイズに基づいてメモリパラメータを調整する (オプションですが推奨)

パスマッピング:

非 CDB:

- /rdsdbdata/db/ORCL_A/datafile/ → /u01/app/oracle/oradata/ORCL/datafile/
- /rdsdbdata/db/ORCL_A/controlfile/ → /u01/app/oracle/oradata/ORCL/controlfile/
- /rdsdbdata/db/ORCL_A/onlinelog/ → /u01/app/oracle/oradata/ORCL/onlinelog/
- /rdsdbdata/admin/ORCL/adump → /u01/app/oracle/admin/ORCL/adump

マルチテナント:

- /rdsdbdata/db/cdb/RDSCDB/datafile/ → /u01/app/oracle/oradata/ORCL/cdb/datafile/
- /rdsdbdata/db/cdb/pdbseed/ → /u01/app/oracle/oradata/ORCL/pdbseed/datafile/
- /rdsdbdata/db/pdb/RDSCDB_A/ → /u01/app/oracle/oradata/ORCL/pdb/datafile/
- /rdsdbdata/db/cdb/RDSCDB_A/controlfile/ → /u01/app/oracle/oradata/ORCL/controlfile/
- /rdsdbdata/admin/RDSCDB/adump → /u01/app/oracle/admin/ORCL/adump

重要: マルチテナントの場合は、パラメータファイルに enable_pluggable_database=TRUE が存在することを確認します。

ステップ 7: **SPFILE** を作成して、スタンバイデータベースを復元する

インスタンスを起動して、SPFILE を作成します。

```
sqlplus / as sysdba
SQL> STARTUP NOMOUNT PFILE='$ORACLE_HOME/dbs/initORCL.ora';
SQL> CREATE SPFILE='/u01/app/oracle/admin/ORCL/pfile/spfileORCL.ora' FROM
 PFILE='$ORACLE_HOME/dbs/initORCL.ora';
SQL> SHUTDOWN IMMEDIATE;
```

シンボリックリンクを作成します。

```
ln -sfn /u01/app/oracle/admin/ORCL/pfile/spfileORCL.ora $ORACLE_HOME/dbs/spfileORCL.ora
```

インスタンスを起動して復元します。

```
SQL> STARTUP NOMOUNT;
rman target /
```

バックアップファイルがソースとは異なるパスにある場合は、まずそれらをカタログ化します。

```
RMAN> catalog start with '/u01/app/oracle/backup/';
```

スタンバイコントロールファイルを復元してマウントします。

```
RMAN> restore standby controlfile from '/u01/app/oracle/backup/standby.ctl';
RMAN> alter database mount;
```

データファイルのパスが異なる場合 (ASM の使用など) は、SET NEWNAME を使用します。

```
RMAN> run {
set newname for database to '+DATA/%b';
restore database;
switch datafile all;
}
```

それ以外の場合は、単純に復元します。

```
RMAN> restore database;
```

データベースを使用可能な最後のシーケンスに復旧します。

```
RMAN> list backup of archivelog all;
```

```
RMAN> recover database until sequence <LAST_SEQ + 1>;
```

> **Note**
>
> マルチテナントの場合、RMAN はすべての PDB を自動的に復元および復旧します。各 PDB を個別に復元する必要はありません。

ステップ 8: TNS とリスナーを設定する

両方のインスタンスで、`tnsnames.ora` に TNS エントリを に追加します。

Example非 CDB:

```
ORCL_A = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <RDS_CUSTOM_IP>)(PORT =
 1521)) (CONNECT_DATA = (SID = ORCL)))
ORCL_B = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <EC2_IP>)(PORT = 1521))
 (CONNECT_DATA = (SID = ORCL)))
```

Exampleマルチテナント:

```
RDSCDB_A = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <RDS_CUSTOM_IP>)(PORT =
 1521)) (CONNECT_DATA = (SID = RDSCDB)))
ORCL_B = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <EC2_IP>)(PORT = 1521))
 (CONNECT_DATA = (SID = ORCL)))
```

両方のインスタンスでリスナーを設定します。RDS Custom では、`listener.ora` に追加します。

Example非 CDB の場合:

```
SID_LIST_L_ORCL_DG=(SID_LIST = (SID_DESC = (SID_NAME = ORCL)(GLOBAL_DBNAME = ORCL)
 (ORACLE_HOME = /rdsdbbin/oracle.19.custom.r1.EE.1)))
L_ORCL_DG=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = 1521)(HOST
 = <RDS_CUSTOM_IP>)))
```

Exampleマルチテナントの場合:

```
SID_LIST_L_RDSCDB_DG=(SID_LIST = (SID_DESC = (SID_NAME = RDSCDB)(GLOBAL_DBNAME =
 RDSCDB) (ORACLE_HOME = /rdsdbbin/oracle.19.custom.r1.EE-CDB.1)))
L_RDSCDB_DG=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = 1521)(HOST
 = <RDS_CUSTOM_IP>)))
```

リスナーを起動します。

```
$ORACLE_HOME/bin/lsnrctl start L_ORCL_DG # or L_RDSCDB_DG for multitenant
```

EC2 で、$ORACLE_HOME/network/admin/listener.ora を作成します。

```
SID_LIST_L_ORCL_DG=(SID_LIST = (SID_DESC = (SID_NAME = ORCL)(GLOBAL_DBNAME = ORCL)
 (ORACLE_HOME = /u01/app/oracle/product/19.0.0/dbhome_1)))
L_ORCL_DG=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = 1521)(HOST = <EC2_IP>)))
```

リスナーを起動します。

```
$ORACLE_HOME/bin/lsnrctl start L_ORCL_DG
```

> **Note**
>
> 必要に応じて RDS Custom で既存のリスナーを使用できますが、別の Data Guard リスナーを作成すると、より優れた分離性が得られます。

> **Important**
>
> tnsping または listener 接続が失敗する場合、EC2 の iptables ルールを確認します。多くの EC2 Linux インスタンスには、ポート 1521 をブロックするデフォルトの iptables ルールがあります。ルールを追加します。sudo iptables -I INPUT 5 -p tcp --dport 1521 -j ACCEPT

ステップ 9: Data Guard ブローカーと設定を有効にする

両方のインスタンスで、Data Guard ブローカーを有効にします。

```
sqlplus / as sysdba
SQL> ALTER SYSTEM SET dg_broker_start=true;
```

RDS Custom プライマリで、Data Guard ブローカーに接続し、設定を作成します。

```
dgmgrl /
```

Example非 CDB の場合:

```
DGMGRL> CREATE CONFIGURATION my_dg_config AS PRIMARY DATABASE IS ORCL_A CONNECT
 IDENTIFIER IS ORCL_A;
DGMGRL> ADD DATABASE ORCL_B AS CONNECT IDENTIFIER IS ORCL_B MAINTAINED AS PHYSICAL;
```

Exampleマルチテナントの場合:

```
DGMGRL> CREATE CONFIGURATION my_dg_config AS PRIMARY DATABASE IS RDSCDB_A CONNECT
 IDENTIFIER IS RDSCDB_A;
DGMGRL> ADD DATABASE ORCL_B AS CONNECT IDENTIFIER IS ORCL_B MAINTAINED AS PHYSICAL;
```

静的接続識別子を設定し、以下を有効にします。

Example非 CDB の場合:

```
DGMGRL> edit database orcl_a set property
 StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PORT=1521)
(HOST=<RDS_CUSTOM_IP>))(CONNECT_DATA=(SID=ORCL)(SERVER=DEDICATED)))';
DGMGRL> edit database orcl_b set property
 StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PORT=1521)
(HOST=<EC2_IP>))(CONNECT_DATA=(SID=ORCL)(SERVER=DEDICATED)))';
DGMGRL> ENABLE CONFIGURATION;
```

Exampleマルチテナントの場合:

```
DGMGRL> edit database rdscdb_a set property
 StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PORT=1521)
(HOST=<RDS_CUSTOM_IP>))(CONNECT_DATA=(SID=RDSCDB)(SERVER=DEDICATED)))';
DGMGRL> edit database orcl_b set property
 StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PORT=1521)
(HOST=<EC2_IP>))(CONNECT_DATA=(SID=ORCL)(SERVER=DEDICATED)))';
DGMGRL> ENABLE CONFIGURATION;
```

> **Note**
>
> Data Guard ブローカーはオプションですが、管理を容易にするために推奨されます。簡単な移行では、ブローカーなしで Data Guard を手動で設定できます。

> **Note**
>
> CDB に対して Data Guard を有効にすると、すべての PDB が自動的に保護されます。任意の PDB によって生成された REDO はスタンバイに配布され、スタンバイの対応する PDB に適用されます。

ステップ 10: スタンバイ REDO ログを設定し、復旧を開始する

EC2 スタンバイインスタンスで、スタンバイ REDO ログファイル (n+1、n はオンライン REDO ロググループの数) を追加します。

```
ALTER DATABASE ADD STANDBY LOGFILE ('slog1.rdo') SIZE 128M;
ALTER DATABASE ADD STANDBY LOGFILE ('slog2.rdo') SIZE 128M;
ALTER DATABASE ADD STANDBY LOGFILE ('slog3.rdo') SIZE 128M;
ALTER DATABASE ADD STANDBY LOGFILE ('slog4.rdo') SIZE 128M;
ALTER DATABASE ADD STANDBY LOGFILE ('slog5.rdo') SIZE 128M;
```

> **Note**
>
> マルチテナントの場合、スタンバイ REDO ログは CDB レベルで作成され、すべての PDB で共有されます。

スタンバイで FAL パラメータを設定します。

Example非 CDB の場合:

```
SQL> alter system set fal_server = 'ORCL_A';
SQL> alter system set fal_client = 'ORCL_B';
```

Exampleマルチテナントの場合:

```
SQL> alter system set fal_server = 'RDSCDB_A';
SQL> alter system set fal_client = 'ORCL_B';
```

マネージドリカバリを開始します。

```
SQL> recover managed standby database disconnect from session;
```

適用遅延をモニタリングします。

```
SQL> SELECT name, value FROM v$dataguard_stats WHERE name = 'apply lag';
```

Data Guard 同期の詳細なモニタリングと管理:

Data Guard のモニタリングは、移行を成功させるために不可欠です。包括的なモニタリング手法は次のとおりです。

1. Data Guard 統計をモニタリングします。

   ```
   -- Comprehensive Data Guard statistics
   SQL> SELECT name, value, unit, time_computed, datum_time
   FROM v$dataguard_stats
   ORDER BY name;
   ```

   モニタリングすべき主要なメトリクス:

   - トランスポート遅延: REDO がプライマリで生成されてスタンバイで受信されるまでの時間差
   - 適用遅延: REDO が生成されてスタンバイ時に適用されるまでの時間差
   - 適用レート: REDO が適用されているレート (MB/秒)
   - 受信済み REDO: スタンバイによって受信した REDO の合計
   - 適用済み REDO: スタンバイによって適用された REDO の合計
2. アーカイブログの配布をモニタリングします。

   プライマリ (RDS Custom) の場合:

   ```
   -- Check archive log generation rate
   SQL> SELECT TO_CHAR(first_time, 'YYYY-MM-DD HH24') AS hour,
          COUNT(*) AS log_count,
          ROUND(SUM(blocks * block_size)/1024/1024/1024, 2) AS size_gb
   FROM v$archived_log
   WHERE first_time > SYSDATE - 1
   GROUP BY TO_CHAR(first_time, 'YYYY-MM-DD HH24')
   ORDER BY hour;

   -- Check archive log destination status
   ```

```
SQL> SELECT dest_id, status, error, destination
FROM v$archive_dest
WHERE status != 'INACTIVE';
```

スタンバイ (EC2) の場合:

```
-- Check archive log apply status
SQL> SELECT sequence#, first_time, next_time, applied
FROM v$archived_log
WHERE applied = 'NO'
ORDER BY sequence#;

-- Check archive log gap
SQL> SELECT thread#, low_sequence#, high_sequence#
FROM v$archive_gap;
```

3. マネージドリカバリプロセスをモニタリングします。

```
-- Check if managed recovery is running
SQL> SELECT process, status, thread#, sequence#, block#, blocks
FROM v$managed_standby
WHERE process LIKE 'MRP%' OR process LIKE 'RFS%';

-- Check recovery progress
SQL> SELECT process, status, sequence#,
       TO_CHAR(timestamp, 'YYYY-MM-DD HH24:MI:SS') AS timestamp
FROM v$managed_standby
ORDER BY process;
```

4. マルチテナントの REDO 適用レートをモニタリングします。

   マルチテナントデータベースの場合、PDB あたりの適用レートをモニタリングします。

```
-- Check redo apply rate per container
SQL> SELECT con_id, name,
       ROUND(SUM(value)/1024/1024, 2) AS redo_applied_mb
FROM v$con_sysstat cs, v$containers c
WHERE cs.con_id = c.con_id
  AND cs.name = 'redo size'
GROUP BY con_id, name
ORDER BY con_id;
```

5. スタンバイ REDO ログをモニタリングします。

```
-- Check standby redo log status
SQL> SELECT group#, thread#, sequence#, bytes/1024/1024 AS size_mb, status
FROM v$standby_log
ORDER BY group#;

-- Check if standby redo logs are being used
SQL> SELECT group#, thread#, sequence#, status, archived
FROM v$standby_log
WHERE status = 'ACTIVE';
```

6. 同期の完了を見積もりします。

   適用レートに基づいて残り時間を計算します。

```
-- Calculate estimated time to catch up
SQL> SELECT
       ROUND((SELECT value FROM v$dataguard_stats WHERE name = 'apply lag')/60, 2) AS
 lag_minutes,
       ROUND((SELECT value FROM v$dataguard_stats WHERE name = 'apply rate')/1024, 2)
 AS apply_rate_mbps,
       ROUND(
         (SELECT value FROM v$dataguard_stats WHERE name = 'apply lag') /
         NULLIF((SELECT value FROM v$dataguard_stats WHERE name = 'apply rate'), 0) /
 60,
         2
       ) AS estimated_catchup_minutes
FROM dual;
```

Data Guard の同期に関する一般的な問題:

問題 1: 適用遅延が大きい

症状:

```
SQL> SELECT name, value FROM v$dataguard_stats WHERE name = 'apply lag';
NAME                           VALUE
------------------------------ -----
apply lag                      +00 01:30:00
```

原因と解決策:

- スタンバイ時の CPU/IO が不十分: EC2 インスタンスタイプをアップグレードするか、EBS IOPS を増やします
- ネットワーク帯域幅の制限: 拡張ネットワーキングまたはより高い帯域幅のインスタンスタイプを使用します
- トランザクションレートが高い複数の PDB: REDO 適用並列処理を増やすことを検討します (Active Data Guard ライセンスが必要)

```
-- Increase apply parallelism (requires Active Data Guard)
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE PARALLEL 4
 DISCONNECT FROM SESSION;
```

問題 2: アーカイブログのギャップ

症状:

```
SQL> SELECT * FROM v$archive_gap;
THREAD# LOW_SEQUENCE# HIGH_SEQUENCE#
------- ------------- --------------
      1          1234           1240
```

解決策:

```
-- FAL (Fetch Archive Log) will automatically fetch missing logs
-- Verify FAL parameters are set correctly
SQL> SHOW PARAMETER fal_server
SQL> SHOW PARAMETER fal_client

-- Manually register missing archive logs if needed
-- On primary, check if logs still exist
SQL> SELECT name FROM v$archived_log WHERE sequence# BETWEEN 1234 AND 1240;

-- If logs are missing on primary, you may need to rebuild the standby
```

問題 3: REDO トランスポートの失敗

症状:

```
SQL> SELECT dest_id, status, error FROM v$archive_dest WHERE dest_id = 2;
```

```
DEST_ID STATUS    ERROR
------- --------- -----------------------------------------
2       ERROR     ORA-16191: Primary log shipping client not logged on standby
```

解決策:

```
-- Check network connectivity
-- Verify TNS configuration
-- Check listener status on standby
-- Restart log transport

SQL> ALTER SYSTEM SET log_archive_dest_state_2 = 'DEFER';
SQL> ALTER SYSTEM SET log_archive_dest_state_2 = 'ENABLE';
```

問題 4: マネージドリカバリが REDO を適用しない

症状:

```
SQL> SELECT process, status FROM v$managed_standby WHERE process = 'MRP0';
PROCESS   STATUS
--------- ------------
MRP0      WAIT_FOR_LOG
```

解決策:

```
# Check if archive logs are arriving
ls -ltr /u01/app/oracle/oradata/ORCL/arch/

# Check alert log for errors
tail -100 $ORACLE_BASE/diag/rdbms/orcl_b/ORCL/trace/alert_ORCL.log

# Restart managed recovery
sqlplus / as sysdba
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

マルチテナントの場合、スタンバイの各 PDB のステータスを確認することもできます。

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
```

予想される出力 (スタンバイで MOUNTED 状態の PDB):

```
CON_ID     NAME                           OPEN_MODE  RES
---------- ------------------------------ ---------- ---
2          PDB$SEED                       MOUNTED
3          ORCLDB                         MOUNTED
```

> **Note**
>
> 物理スタンバイでは、PDB はマネージドリカバリ中も MOUNTED 状態のままです。

ステップ 11: スイッチオーバーを実行する

スタンバイが完全に同期され、準備が整ったことを確認したら、スイッチオーバーを実行します。マルチテナントの場合、CDB 全体 (すべての PDB) が RDS Custom プライマリから EC2 スタンバイにスイッチオーバーします。

RDS Custom プライマリインスタンスで、Data Guard ブローカーに接続し、両方のデータベースのスイッチオーバーの準備が整っていることを確認します。

Example非 CDB の場合:

```
DGMGRL> VALIDATE DATABASE ORCL_A;
DGMGRL> VALIDATE DATABASE ORCL_B;
```

Exampleマルチテナントの場合:

```
DGMGRL> VALIDATE DATABASE RDSCDB_A;
DGMGRL> VALIDATE DATABASE ORCL_B;
```

どちらも `Ready for Switchover: Yes` と表示されるはずです。

RDS Custom プライマリから EC2 スタンバイに切り替えます。

```
DGMGRL> SWITCHOVER TO ORCL_B;
```

スイッチオーバーが成功したことを確認します。

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
```

EC2 インスタンス (ORCL_B) がプライマリデータベースになり、RDS Custom インスタンスが物理スタンバイになりました。

ステップ 12: PDB を開く (マルチテナントのみ)

スイッチオーバー後、EC2 の CDB は READ WRITE モードでオープンになりますが、すべての PDB は MOUNTED 状態です。手動で開く必要があります。

EC2 の新しいプライマリに接続します。

```
sqlplus / as sysdba
SQL> SELECT name, open_mode, database_role, cdb FROM v$database;
```

正常な出力:

```
NAME      OPEN_MODE            DATABASE_ROLE    CDB
--------- -------------------- ---------------- ---
ORCL      READ WRITE           PRIMARY          YES
```

現在の PDB ステータスを確認します。

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
```

予想される出力 (MOUNTED 状態の PBD - ORCLDB という名前の PDB が 1 つある例):

```
CON_ID     NAME                           OPEN_MODE  RES
---------- ------------------------------ ---------- ---
2          PDB$SEED                       READ ONLY  NO
3          ORCLDB                         MOUNTED
```

すべての PDB を開きます。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
```

プラガブルデータベースが変更されました。

すべての PDB が READ WRITE モードでオープンされていることを確認します。

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
```

正常な出力:

```
CON_ID     NAME                           OPEN_MODE  RES
---------- ------------------------------ ---------- ---
2          PDB$SEED                       READ ONLY  NO
3          ORCLDB                         READ WRITE NO
```

ステップ 13: 起動時に PDB を自動オープンするように設定する (マルチテナントのみ)

CDB の起動時に PDB の自動オープンを設定するには、状態保存メソッド (Oracle 19c に推奨) を使用します。

```
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
Pluggable database altered.
```

保存された状態を確認します。

```
SQL> SELECT con_name, instance_name, state FROM dba_pdb_saved_states;
```

PDB サービスがリスナーに登録されていることを確認します。

```
lsnrctl services
```

予想される出力には、CDB と各 PDB のサービスが表示されるはずです。サービスが表示されない場合:

```
SQL> ALTER SYSTEM REGISTER;
```

次に、`lsnrctl services` でもう一度確認します。

ステップ 14: RDS Custom オブジェクトを削除する

これは EC2 上のセルフマネージドデータベースになったため、RDS Custom 固有のユーザーとオブジェクトを削除する必要があります。クリーンアッププロセスは、非 CDB アーキテクチャとマルチテナントアーキテクチャでわずかに異なります。

> **Important**
>
> RDS 固有のユーザーとテーブルスペースを削除する前に、これらのスキーマの下にアプリケーションオブジェクトが存在しないことを確認します。

```
SQL> SELECT owner, object_type, COUNT(*)
FROM dba_objects
WHERE owner IN ('RDSADMIN', 'RDS_DATAGUARD')
  AND object_name NOT LIKE 'RDS%'
  AND object_name NOT LIKE 'SYS_%'
GROUP BY owner, object_type;
```

アプリケーションオブジェクトが見つかった場合は、先に進む前に適切なアプリケーションスキーマに移行します。

非 CDB クリーンアップ:

```
sqlplus / as sysdba

-- Drop RDS-specific users
SQL> DROP USER RDSADMIN CASCADE;
SQL> DROP USER RDS_DATAGUARD CASCADE;

-- Drop RDS-specific directories
SQL> DROP DIRECTORY OPATCH_INST_DIR;
SQL> DROP DIRECTORY OPATCH_LOG_DIR;
SQL> DROP DIRECTORY OPATCH_SCRIPT_DIR;

-- Drop the RDSADMIN tablespace
SQL> DROP TABLESPACE RDSADMIN INCLUDING CONTENTS AND DATAFILES;

-- Verify removal
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';
```

正常な出力: `no rows selected`

マルチテナントクリーンアップ:

マルチテナント環境では、RDS Custom はすべての PDB から参照可能な共通ユーザーを CDB$ROOT に作成します。CDB$ROOT からクリーンアップする必要があります。

```
sqlplus / as sysdba

-- Verify you are in CDB$ROOT
SQL> SHOW CON_NAME;

-- Check for RDS-specific common users (including C## prefixed users)
```

```
SQL> SELECT username, common, con_id FROM cdb_users
WHERE username LIKE 'RDS%' OR username LIKE 'C##RDS%'
ORDER BY username;

-- Drop non-common users
SQL> DROP USER RDSADMIN CASCADE;
SQL> DROP USER RDS_DATAGUARD CASCADE;

-- If any C## common users exist
-- Example (adjust based on your findings):
-- SQL> DROP USER C##RDS_DATAGUARD CASCADE;
-- Drop RDS-specific directories
SQL> DROP DIRECTORY OPATCH_INST_DIR;
SQL> DROP DIRECTORY OPATCH_LOG_DIR;
SQL> DROP DIRECTORY OPATCH_SCRIPT_DIR;

-- Drop the RDSADMIN tablespace
SQL> DROP TABLESPACE RDSADMIN INCLUDING CONTENTS AND DATAFILES;

-- Verify removal from CDB$ROOT
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';

-- Verify removal from each PDB (example with one PDB named ORCLDB)
SQL> ALTER SESSION SET CONTAINER = ORCLDB;
SQL> SELECT username FROM dba_users WHERE username LIKE '%RDS%';
```

すべてのクエリで予想される出力: no rows selected

ステップ 15: 自動起動を設定する

SPFILE が使用されていることを確認します。

```
sqlplus / as sysdba
SQL> SHOW PARAMETER spfile;
```

spfile パスが正しい場合、アクションは必要ありません。正しくない場合は、作成します。

```
SQL> CREATE SPFILE FROM MEMORY;
```

データベースを再起動します。

```
SQL> SHUTDOWN IMMEDIATE;
```

```
SQL> STARTUP;
```

マルチテナントの場合は、すべての PDB を開きます (以前に状態を保存した場合は自動オープンするはずです)。

```
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
```

PDB が開いていない場合は、手動で開きます。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
```

/etc/oratab を編集します。

```
vi /etc/oratab
```

ORCL の行を N から Y に変更します。

```
ORCL:/u01/app/oracle/product/19.0.0/dbhome_1:Y
```

ステップ 16: 最終検証

移行されたデータベースに対して包括的な検証を実行します。

Example非 CDB の場合:

```
sqlplus / as sysdba

-- Verify database role and status
SQL> SELECT name, open_mode, log_mode, database_role FROM v$database;

-- Check database size
SQL> SELECT SUM(bytes)/1024/1024/1024 AS size_gb FROM dba_data_files;

-- Verify all objects are valid
SQL> SELECT owner, object_type, COUNT(*)
     FROM dba_objects
     WHERE status = 'INVALID'
     GROUP BY owner, object_type;
```

```
-- Verify data files
SQL> SELECT name, status FROM v$datafile;

-- Test application connectivity
SQL> SELECT username, machine, program FROM v$session WHERE username IS NOT NULL;
```

Exampleマルチテナントの場合:

```
sqlplus / as sysdba

-- Verify CDB status
SQL> SELECT name, open_mode, log_mode, cdb, database_role FROM v$database;

-- Verify all PDBs are open
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;

-- Check total CDB size
SQL> SELECT SUM(bytes)/1024/1024/1024 AS total_size_gb FROM cdb_data_files;

-- Check size per PDB
SQL> SELECT p.name AS pdb_name,
       ROUND(SUM(d.bytes)/1024/1024/1024, 2) AS size_gb
FROM v$pdbs p
JOIN cdb_data_files d ON p.con_id = d.con_id
GROUP BY p.name,p.con_id
ORDER BY p.con_id;

-- Verify all objects are valid across all PDBs
SQL> SELECT con_id, owner, object_type, COUNT(*)
     FROM cdb_objects
     WHERE status = 'INVALID'
     GROUP BY con_id, owner, object_type;

-- Verify PDB services are registered
SQL> SELECT name FROM v$services ORDER BY name;

Test application connectivity:

# Non-CDB
sqlplus <app_user>/<password>@<EC2_IP>:1521/ORCL

# Multitenant (connect to PDB)
sqlplus <app_user>/<password>@<EC2_IP>:1521/<PDB_NAME>
```

アプリケーションの接続性をテストします。

```
# Non-CDB
sqlplus <app_user>/<password>@<EC2_IP>:1521/ORCL

# Multitenant (connect to PDB)
sqlplus <app_user>/<password>@<EC2_IP>:1521/<PDB_NAME>
```

ステップ 17: バックアップファイルをクリーンアップする

検証に成功したら、バックアップファイルを削除し、別の EBS ボリュームを使用している場合はバックアップボリュームをデタッチします。

```
rm -rf /u01/app/oracle/backup/*
```

バックアップに別の EBS ボリュームを使用する場合:

```
# Unmount the volume
sudo umount /u01/app/oracle/backup

# Detach and delete the EBS volume from AWS Console or CLI
aws ec2 detach-volume --volume-id <volume-id>
aws ec2 delete-volume --volume-id <volume-id>
```

ステップ 18: RDS Custom オートメーションを再開する

検証期間中に RDS Custom インスタンスをフォールバックとして実行し続ける場合は、オートメーションを再開します。

```
aws rds modify-db-instance \
  --db-instance-identifier my-custom-instance \
  --automation-mode full \
  --region us-east-1
```

一般的な問題のトラブルシューティング

このセクションでは、RMAN 複製アプローチと Oracle Data Guard アプローチの両方で、移行中に発生する可能性がある一般的な問題について説明します。対象となるのは、非 CDB アーキテクチャとマルチテナントアーキテクチャの両方です。

ORA-09925: 監査証跡ファイルを作成できません

原因: audit_file_dest パラメータで指定された監査ディレクトリがターゲット EC2 インスタンスに存在しません。

解決策:

監査ディレクトリが存在し、適切なアクセス許可があることを確認します。

```
mkdir -p /u01/app/oracle/admin/ORCL/adump
chown -R oracle:oinstall /u01/app/oracle/admin/ORCL
chmod -R 755 /u01/app/oracle/admin/ORCL
```

ORA-01261: パラメータ **db_create_file_dest** の送信先文字列を変換できません

原因: db_create_file_dest パラメータで指定されたディレクトリがターゲット EC2 インスタンスに存在しません。

解決策:

非 CDB の場合:

```
mkdir -p /u01/app/oracle/oradata/ORCL
chown -R oracle:oinstall /u01/app/oracle/oradata/ORCL
chmod -R 755 /u01/app/oracle/oradata/ORCL
```

マルチテナントの場合:

```
mkdir -p /u01/app/oracle/oradata/ORCL/pdb/datafile
chown -R oracle:oinstall /u01/app/oracle/oradata/ORCL
chmod -R 755 /u01/app/oracle/oradata/ORCL
```

ORA-01804: タイムゾーン情報の初期化に失敗しました

RDS ソースのタイムゾーンバージョンが EC2 $ORACLE_HOME にインストールされているタイムゾーンバージョンよりも高い場合、RDSADMIN ユーザーを削除するときにこのエラーが発生する可能性があります。

解決策:

1. タイムゾーンのバージョンを確認します。

```
SELECT * FROM v$timezone_file;
SELECT PROPERTY_NAME, PROPERTY_VALUE
FROM database_properties
WHERE PROPERTY_NAME LIKE '%DST%';
```

2. 回避策として、$ORACLE_HOME で使用できるものと一致するようにタイムゾーンファイル環境変数を設定します。

```
ls $ORACLE_HOME/oracore/zoneinfo/timezlrg_*.dat
export ORA_TZFILE=$ORACLE_HOME/oracore/zoneinfo/timezone_40.dat
```

   インストールで使用可能なバージョンと一致するように数値を調整します。

3. 削除を再試行します。

```
sqlplus / as sysdba
SQL> DROP USER RDSADMIN CASCADE;
```

RU 間の移行の問題 (異なるリリースアップデート)

原因: ターゲット EC2 インスタンスの Oracle Release Update (RU) またはワンオフパッチがソース RDS Custom インスタンスとは異なるため、移行中または移行後に互換性エラーが発生します。

一般的なエラー。

- REDO 適用中の ORA-00600 内部エラー (Data Guard)
- ORA-39700 データベースは UPGRADE オプション付きで開く必要があります
- 移行後のディクショナリの不整合
- DBA_REGISTRY または DBA_OBJECTS の無効なオブジェクト

解決策:

ベストプラクティス - RU バージョンとワンオフパッチを正確に一致させます。

1. ソースとターゲットの両方で正確な RU バージョンを確認します。

```
-- On both source and target
SQL> SELECT * FROM v$version;
```

```
SQL> SELECT patch_id, patch_uid, version, action, status, description
FROM dba_registry_sqlpatch
ORDER BY action_time DESC;
```

2. $ORACLE_HOME パッチレベルを確認します。

```
# On both instances
$ORACLE_HOME/OPatch/opatch lsinventory
```

3. バージョンが一致しない場合は、必要に応じてパッチを適用またはロールバックして移行前に調整します。
4. RU が一致しない状態で進める必要がある場合は、移行後に datapatch を実行します。

```
cd $ORACLE_HOME/OPatch
./datapatch -verbose
```

5. 無効なオブジェクトを確認し、再コンパイルします。

```
SQL> @?/rdbms/admin/utlrp.sql

SQL> SELECT owner, object_type, COUNT(*)
FROM dba_objects
WHERE status = 'INVALID'
GROUP BY owner, object_type;
```

ネットワーク接続の問題

原因: セキュリティグループ、ネットワーク ACL、または `iptables` が Oracle リスナーポートをブロックしている。

解決策:

1. セキュリティグループがポートを双方向に許可することを確認します
2. EC2 の iptables を確認します。

```
sudo iptables -L INPUT -n -v
```

3. 必要に応じてルールを追加します。

```
# Insert rule before the REJECT rule (typically position 5)
sudo iptables -I INPUT 5 -p tcp --dport 1521 -j ACCEPT

# For enhanced security, allow only from specific source IPs
sudo iptables -I INPUT 5 -p tcp -s <RDS_Custom_IP> --dport 1521 -j ACCEPT

# Save rules permanently
sudo service iptables save
```

4. 接続をテストします。

```
telnet <EC2_instance_IP> 1521
tnsping DB_SOURCE
tnsping DB_TARGET
```

移行後に PDB が開きません (マルチテナントのみ)

原因: これは想定される動作です。RMAN 複製または Data Guard スイッチオーバー後、CDB はオープンですが、PDB は MOUNTED 状態です。

解決策:

手動で開きます。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
```

特定の PDB が開いていない場合は、アラートログにエラーがないか確認します。

```
tail -100 $ORACLE_BASE/diag/rdbms/orcl/ORCL/trace/alert_ORCL.log
```

一般的な原因には、データファイルの欠落やパスマッピングの問題などがあります。

PDB データファイルが見つからないか、パスが一致しません (マルチテナントのみ)

原因: 移行において、特に OMF ベースの PDB データファイルについて、すべてのソースパスが正しくマッピングされませんでした。

解決策:

1. 欠落しているデータファイルを確認します。

```
SQL> SELECT name, status FROM v$datafile WHERE status != 'ONLINE';
```

2. ファイルが間違ったディレクトリに配置された場合は、コントロールファイル内で名前を変更します。

```
SQL> ALTER DATABASE RENAME FILE '/wrong/path/datafile.dbf' TO '/correct/path/
datafile.dbf';
```

3. これを防ぐには、パラメータファイルを設定する前に、必ず SELECT con_id, name FROM v$datafile ORDER BY con_id; を使用してソースデータファイルのパスを確認してください。

PDB サービスがリスナーに登録されていない (マルチテナントのみ)

原因: PDB を開いた後、リスナーが PDB サービスを認識しません。

解決策:

1. サービス登録を強制します。

```
SQL> ALTER SYSTEM REGISTER;
```

2. サービスがまだ表示されない場合は、local_listener パラメータを確認します。

```
SQL> SHOW PARAMETER local_listener;
```

   正しいリスナーアドレスを指していることを確認します。必要に応じて更新します。

```
SQL> ALTER SYSTEM SET local_listener='(ADDRESS=(PROTOCOL=TCP)(HOST=<EC2_instance_IP>)
(PORT=1521))';
SQL> ALTER SYSTEM REGISTER;
```

3. 以下を使用して検証します。

```
lsnrctl services
```

CDB 再起動後に PDB 自動で開かない (マルチテナントのみ)

原因: PDB 状態保存が設定されていません。

解決策:

PDB 状態保存が設定されていることを確認します。

```
SQL> SELECT con_name, instance_name, state FROM dba_pdb_saved_states;
```

行が返されない場合は、状態を保存します。

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
```

Data Guard REDO トランスポートが機能しない

原因: ネットワーク接続の問題、誤った TNS 設定、または FAL パラメータが設定されていません。

解決策:

1. スタンバイが MOUNT モードであることを確認します。

   ```
   SQL> SELECT status FROM v$instance;
   ```
2. スタンバイで fal_server と fal_client が正しく設定されていることを確認します。

   ```
   SQL> SHOW PARAMETER fal_server
   SQL> SHOW PARAMETER fal_client
   ```
3. ネットワーク接続を検証します。

   ```
   tnsping ORCL_A # or RDSCDB_A for multitenant
   ```
4. プライマリの log_archive_dest_2 パラメータがスタンバイを指していることを確認します (ブローカーを使用せずに手動で設定した場合)。

Data Guard 適用遅延が複数の PDB で増加します (マルチテナントのみ)

原因: 複数の PDB を含む CDB の場合、すべてのコンテナにわたる変更量が多いため、REDO の適用が遅くなる可能性があります。

解決策:

1. 適用レートを確認します。

```
SQL> SELECT name, value, unit FROM v$dataguard_stats WHERE name IN ('apply rate',
 'apply lag');
```

2. REDO 適用の並列処理を増やすことを検討してください (Active Data Guard ライセンスが必要です)。

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE PARALLEL 4
 DISCONNECT FROM SESSION;
```

3. スタンバイインスタンスにリソース制約 (CPU、I/O) がないことを確認します。

RMAN アーカイブログのバックアップが ORA-19625 で失敗しました

原因: RDS Custom オートメーションは古いアーカイブログをディスクから消去しましたが、RMAN のコントロールファイルにはまだそれらのレコードがあります。

解決策:

1. 古いアーカイブログエントリをクロスチェックしてクリーンアップします。

```
RMAN> CROSSCHECK ARCHIVELOG ALL;
RMAN> DELETE NOPROMPT EXPIRED ARCHIVELOG ALL;
```

2. アーカイブログバックアップのみを再実行します。

```
RMAN> RUN {
SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
BACKUP AS COMPRESSED BACKUPSET
FILESPERSET 50
FORMAT '/rdsdbdata/backup/arch_%U'
ARCHIVELOG ALL;
}
```

マルチテナントで共通ユーザーの削除が失敗する (マルチテナントのみ)

原因: 共通ユーザー (C## のプレフィックス) は CONTAINER=ALL 句を使用して削除する必要があります。

解決策:

```
-- For common users
SQL> DROP USER C##RDS_DATAGUARD CASCADE CONTAINER=ALL;

-- For non-common users in CDB$ROOT
SQL> DROP USER RDSADMIN CASCADE;
```

CDB$ROOT に接続していることを確認します。

```
SQL> SHOW CON_NAME;
```

移行後のタスク

移行が成功したら、これらの追加タスクを完了して、EC2 上のセルフマネージド Oracle データベースの本番稼働準備が整っていることを確認します。

アプリケーション接続文字列を更新する

非 CDB の場合:

- アプリケーションを新しい EC2 インスタンスエンドポイントにポイントする
- EC2 インスタンスの IP またはホスト名を使用するように接続文字列を更新する
- アプリケーションのすべての機能を徹底的にテストする

マルチテナントの場合:

- アプリケーションを新しい EC2 インスタンス PDB サービス名 (ORCLDB や特定の PDB 名など) にポイントする
- アプリケーションが CDB ではなく正しい PDB に接続されていることを確認する
- PDB サービス名を使用するように接続文字列を更新する
- 各 PDB のすべてのアプリケーション機能をテストする

接続文字列の例:

```
# Non-CDB
jdbc:oracle:thin:@<EC2_IP>:1521/ORCL

# Multitenant (connect to PDB)
```

```
jdbc:oracle:thin:@<EC2_IP>:1521/ORCLDB
```

バックアップ戦略を設定する

セルフマネージドデータベースの包括的なバックアップ戦略を設定します。

RMAN バックアップ:

- 完全、増分、アーカイブログのバックアップ用に自動 RMAN バックアップスクリプトを設定する
- 目標復旧時点 (RPO) に基づいてバックアップ保持ポリシーを設定する
- 耐久性とコスト効率のために Amazon S3 にバックアップを保存する
- バックアップ復元手順を定期的にテストする

AWS Backup:

- EBS ボリュームスナップショットに AWS Backup を使用する
- バックアップスケジュールと保持ポリシーを設定する
- ディザスタリカバリ用のクロスリージョンバックアップコピーを有効にする

マルチテナントの場合:

- CDB の RMAN バックアップには、すべての PDB が自動的に含まれる
- 必要に応じて個々の PDB をバックアップすることもできる
- ビジネス要件に基づいて PDB 固有のバックアップスケジュールを検討する

RMAN バックアップスクリプトの例:

```
#!/bin/bash
export ORACLE_HOME=/u01/app/oracle/product/19.0.0/dbhome_1
export ORACLE_SID=ORCL
export PATH=$ORACLE_HOME/bin:$PATH
rman target / << EOF
run {
  backup as compressed backupset database plus archivelog;
  delete noprompt obsolete;
}
exit;
EOF
```

モニターリングの設定

EC2 でホストされている Oracle データベースの包括的なモニタリングを実装します。

Amazon CloudWatch

- EC2 インスタンスのヘルス、ディスク使用量、カスタム Oracle メトリクスの CloudWatch メトリクスを設定する
- 重要なしきい値の CloudWatch アラームを作成する
- データベースアラートログのモニタリングに CloudWatch Logs を使用する

Oracle Enterprise Manager (OEM):

- 可能な場合は、データベースモニタリング用に OEM を設定する
- パフォーマンスのモニタリングと診断を設定する
- 重要なイベントの自動アラートを設定する

サードパーティー製ツール:

- データベースモニタリングに Datadog、New Relic、Prometheus などのツールを検討する
- 既存のモニタリングインフラストラクチャと統合する

モニタリングすべき主要なメトリクス:

- テーブルスペースの使用状況
- アーカイブログスペース
- 無効なオブジェクト
- セッション数
- 待機イベント
- CPU とメモリの使用率
- I/O パフォーマンス

マルチテナントの場合:

- CDB レベルと PDB レベルの両方のメトリクスをモニタリングする

- PDB リソースの使用状況とクォータのアラートを設定する
- PDB 固有のパフォーマンスメトリクスを追跡する

セキュリティグループとネットワーク ACL を設定する

EC2 インスタンスのセキュリティを確認して強化します。

セキュリティグループ:

- データベースポートへのアクセスを、承認されたアプリケーションサーバーと踏み台ホストのみに制限する
- 移行中に作成された過度に許可されたルールを削除する
- セキュリティグループのルールとその目的を文書化する

ネットワーク ACL:

- 追加のセキュリティレイヤー用に VPC ネットワーク ACL を設定する
- 多層防御セキュリティ戦略を実装する

SSH アクセス:

- SSH アクセスを特定の IP 範囲に制限するか、AWS Systems Manager Session Manager を使用する
- パスワード認証を無効にし、キーベースの認証のみを使用する
- 特権アクセスに多要素認証 (MFA) を導入する

暗号化:

- EBS ボリュームの保管時の暗号化を有効にする
- Oracle Native Network Encryption または TLS を使用して、データベース接続の転送中の暗号化を有効にする
- 暗号化キーを定期的にローテーションする

高可用性を実装する

ワークロードに高可用性が必要な場合は、次の実装を検討してください。

Oracle Data Guard:

- ディザスタリカバリ用に別の EC2 インスタンスに新しいスタンバイデータベースを設定する
- マルチテナントの場合、Data Guard はすべての PDB を含む CDB 全体を保護する
- スタンバイは別のアベイラビリティーゾーンまたはリージョンに配置できる
- スクリプトまたはサードパーティーツールを使用して、自動フェイルオーバーメカニズムを実装する

AWS マルチ AZ 配置:

- スタンバイインスタンスを異なるアベイラビリティーゾーンにデプロイする
- DNS フェイルオーバーに Amazon Route 53 を使用する
- フェイルオーバーをサポートするアプリケーションレベルの接続プーリングを実装する

バックアップとリカバリ:

- テスト済みの復元手順を備えた定期的なバックアップを維持する
- 目標復旧時間 (RTO) と目標復旧時点 (RPO) を文書化する
- ディザスタリカバリドリルを定期的に実行する

徹底的なアプリケーションテストを実行する

RDS Custom インスタンスを廃止する前:

機能テスト:

- すべてのアプリケーション機能が正しく動作することを確認する
- データベース依存機能をすべてテストする
- データの整合性と一貫性を検証する

パフォーマンステスト:

- パフォーマンスメトリクスを RDS Custom ベースラインと比較する
- パフォーマンスのリグレッションを特定して対処する
- 必要に応じてクエリとインデックスを最適化する

負荷テスト:

- 予想されるピーク負荷でデータベースをテストする
- リソース使用率が許容範囲内に収まっていることを確認する
- ボトルネックを特定して対処する

フェイルオーバーテスト (HA が設定されている場合):

- フェイルオーバーシナリオをテストする
- アプリケーション再接続ロジックを検証する
- 実際の RTO と RPO を測定する

バックアップと復元のテスト:

- バックアップと復元の手順が正しく機能することを確認する
- ポイントインタイムリカバリをテストする
- バックアップの整合性を検証する

マルチテナントの場合:

- 各 PDB を個別にテストする
- PDB の分離とリソース割り当てを検証する
- PDB 固有のオペレーションをテストする (クローン、アンプラグ/プラグなど)

RDS Custom インスタンスの廃止

検証期間 (通常は 1～2 週間) が成功した後:

1. 最終バックアップ: アーカイブ目的で RDS Custom インスタンスの最終バックアップを取得する

```
# Create final snapshot
aws rds create-db-snapshot \
  --db-instance-identifier my-custom-instance \
  --db-snapshot-identifier my-custom-instance-final-snapshot \
  --region us-east-1
```

2. 移行を文書化する: 新しい EC2 設定でランブックとドキュメントを更新する

3. RDS Custom インスタンスを削除する: AWS コンソールまたは CLI を使用してインスタンスを削除する

```
# Delete RDS Custom instance without final snapshot (if already created above)
aws rds delete-db-instance \
  --db-instance-identifier my-custom-instance \
  --skip-final-snapshot \
  --region us-east-1

# Or create a final snapshot before deletion
aws rds delete-db-instance \
  --db-instance-identifier my-custom-instance \
  --final-db-snapshot-identifier my-custom-instance-final-snapshot \
  --region us-east-1
```

4. リソースのクリーンアップ: 不要になった関連スナップショット、パラメータグループ、オプショングループを削除する
5. ドキュメントの更新: すべての運用ドキュメントに新しい EC2 ベースのアーキテクチャが反映されていることを確認する

比較: RMAN アクティブ複製と Oracle Data Guard

次の表は、2 つの移行オプションの主な違いをまとめたものです。

側面	RMAN アクティブ複製	Oracle Data Guard:
ソースデータベースの可用性	複製全体のオンライン	プロセス全体を通してオンライン
ダウンタイム	数分 (最終カットオーバーのみ)	数秒から数分 (スイッチオーバー)
複雑さ	Lower	より高い
移行期間	単一複製オペレーション	初期設定 + 連続同期
継続的な同期	いいえ	はい
フォールバック機能	手動 (ソースの実行を維持)	組み込み (自動)

側面	RMAN アクティブ複製	Oracle Data Guard:
カットオーバー前のテスト	限定 (複製後のテスト)	完全なテストが可能 (スタンバイのテストも可能)
ネットワーク帯域幅	複製中は高	中程度 (連続)
ソースデータベースへの影響	最小 (読み取りオペレーション)	最小 (REDO 配布)
次の用途に適しています	ほとんどの移行、簡単なカットオーバー	ミッションクリティカルで、ほぼゼロのダウンタイムが必要
非 CDB のサポート	可能	はい
マルチテナントのサポート	はい (CDB 全体)	はい (CDB 全体)
移行後の PDB の状態	CDB はオープン、PDB は MOUNTED	CDB はオープン、PDB は MOUNTED
RMAN が必要	はい	はい (バックアップベースのアプローチでの初期バックアップの場合)
Data Guard が必要	いいえ	はい
必要なスキルレベル	中級	アドバンスト
カットオーバープロセス	アプリケーションを EC2 にリダイレクトする	スイッチオーバーコマンド + アプリケーションのリダイレクト

比較: 非 CDB とマルチテナントの移行

次の表は、非 CDB データベースとマルチテナントデータベースの移行の主な違いをまとめたものです。

側面	非 CDB の移行	マルチテナント (PDB を含む CDB) の移行
データベースタイプ	単一インスタンスの非 CDB (例: ORCL)	CDB (ソース: RDSCDB、ターゲット: ORCL) に CDB$ROOT + PDB$SEED + 1 つ以上の PDB が含まれる
移行範囲	単一のデータベース	CDB 全体 (すべての PDB が自動的に含まれます)
RMAN 複製スコープ	単一のデータベースを複製	CDB 全体 (すべてのコンテナ) を複製
Data Guard スコープ	単一のデータベースを保護	CDB 全体を保護 (すべての PDB が自動的に含まれます)
パラメータファイル	標準 init パラメータ	enable_pluggable_database =TRUE を含める必要があります
移行後のデータベースの状態	データベースが READ WRITE モードで開きます	CDB が READ WRITE モードで開きます。PDB は MOUNTED 状態のままです
PDB を開く	該当なし	ALTER PLUGGABLE DATABASE ALL OPEN を使用して、すべての PDB を手動で開く必要があります
起動時の PDB 自動オープン	該当なし	PDB の状態保存または起動トリガーを設定する必要があります
検証:	単一のデータベースチェック	CDB と各 PDB を個別に検証する必要があります

側面	非 CDB の移行	マルチテナント (PDB を含む CDB) の移行
RDS 固有のクリーンアップ	単一のデータベースからユーザー/オブジェクトを削除する	CDB$ROOT から共通ユーザーを削除 (PDB にカスケード)、C## ユーザーを扱う
TNS/リスナー設定	データベースの単一サービス	CDB サービス + 個々の PDB サービスが動的に登録されます
アプリケーション接続文字列	単一のデータベースへの接続	個々の PDB サービス (CDB ではない) に接続
バックアップ戦略	単一データベースのバックアップ	CDB 全体 (すべての PDB を含む) または個々の PDB のバックアップ
リソース管理	データベースレベルのリソース管理	Resource Manager による CDB レベルと PDB レベルのリソース管理
複雑さ	複雑性が低い	複数のコンテナと OMF パスにより複雑性が高い

ベストプラクティスとレコメンデーション

このセクションでは、RDS Custom for Oracle から EC2 への移行を成功させるための包括的なベストプラクティスについて説明します。

移行前の計画

1. 徹底的な評価を実施する。

   移行を開始する前に、環境の包括的な評価を実行します。

   - データベースインベントリ: すべてのデータベース、そのサイズ、アーキテクチャ (非 CDB とマルチテナント)、依存関係を文書化する

- アプリケーションの依存関係: データベースに接続するすべてのアプリケーションとその接続方法を特定する
- パフォーマンスベースライン: 移行後の比較のためにパフォーマンスメトリクス (CPU、メモリ、I/O、ネットワーク) をキャプチャする
- バックアップと復旧の要件: RPO (目標復旧時点) と RTO (目標復旧時間) を文書化する
- コンプライアンス要件: 移行に影響する可能性のある規制またはコンプライアンス要件を特定する

2. 適切な EC2 インスタンスタイプを選択する。

   ワークロードの特性に基づいて EC2 インスタンスタイプを選択します。

ワークロードタイプ	推奨インスタンスファミリー	主な特徴
汎用 OLTP	M6i、M6a、M7i	コンピューティング、メモリ、ネットワークのバランスが取れている
メモリ集約型	R6i、R6a、R7i、X2idn	メモリ対 CPU 比が高い
コンピューティング集約型	C6i、C6a、C7i	高い CPU パフォーマンス
I/O 集約型	I4i、Im4gn	高性能ローカル NVMe SSD ストレージ
混合ワークロード	M5、M5a、M5n	費用対効果の高いバランスのとれたパフォーマンス

   インスタンスのサイズ設定ガイドライン:

   - RDS Custom インスタンスと同じインスタンスクラスから開始する
   - テスト移行中のリソース使用率をモニタリングする
   - AWS Compute Optimizer のレコメンデーションの活用を検討する
   - 成長とピーク負荷に備えて 20～30% のヘッドルームを計画する

3. ストレージアーキテクチャを設計する。

   EBS ボリュームの種類:

ボリュームタイプ	ユースケース	パフォーマンス	Cost
gp3	汎用、ほとんどのワークロード	最大 16,000 IOPS、1,000 MB/秒	低
io2 Block Express	ミッションクリティカル、高性能	最大 256,000 IOPS、4,000 MB/秒	高
Io1	高性能データベース	最大 64,000 IOPS、1,000 MB/秒	やや高い
gp2	レガシーの汎用	最大 16,000 IOPS	低

ストレージレイアウトの推奨事項:

```
# Recommended layout for production databases
      /u01/app/oracle          # Oracle software (50-100 GB, gp3)
      /u01/app/oracle/oradata  # Data files (sized for database, gp3 or io2)
      /u01/app/oracle/arch     # Archive logs (separate volume, gp3)
      /u01/app/oracle/backup   # Backups (separate volume, gp3, can be detached post-
migration)
```

個別ボリュームの利点:

- 独立した IOPS 割り当て
- 容易な容量管理
- バックアップとスナップショットの戦略の簡素化
- パフォーマンスの分離性の向上

4. ロールバックプランを確立する。

   常にロールバック戦略を用意しておきます。

   - 検証期間中 (1～2 週間を推奨) に RDS Custom インスタンスを実行し続ける
   - ソースとターゲットの両方の定期的なバックアップを維持する
   - 接続文字列の変更を含むロールバック手順を文書化する
   - 非本番環境でロールバックプロセスをテストする

- ロールバック基準を定義する (パフォーマンスの低下、データの不整合、アプリケーションエラー)

移行実行のベストプラクティス

1. 移行のタイミング:

   最適な時間枠を選択する。

   - トラフィックが少ない時間帯: 週末、祝日、またはオフピーク時間
   - メンテナンスウィンドウ: 可能であれば、スケジュールされたメンテナンスに合わせる
   - 月末/四半期末を避ける: これらの期間は通常、トランザクション量が多いため
   - タイムゾーンを検討する: グローバルアプリケーションの場合は、リージョン間の影響を最小限に抑える時間を選択する
2. コミュニケーション計画:

   明確なコミュニケーションを確立する。

   - ステークホルダーへの通知: 少なくとも 2 週間前にすべてのステークホルダーに通知する
   - アプリケーションチーム: 接続文字列の更新についてアプリケーションチームと調整する
   - ステータスの更新: 移行中は定期的に最新情報を提供する
   - エスカレーションパス: 問題の明確なエスカレーション手順を定義する
   - 移行後のコミュニケーション: 正常に完了したことと、フォローアップアクションを確認する
3. 検証チェックポイント:

   各段階で検証を実装します。

   移行前の検証:

```
-- Capture object counts
SQL> SELECT object_type, COUNT(*) FROM dba_objects GROUP BY object_type ORDER BY
 object_type;

-- Capture tablespace usage
SQL> SELECT tablespace_name, ROUND(SUM(bytes)/1024/1024/1024, 2) AS size_gb
FROM dba_data_files GROUP BY tablespace_name;

-- Capture user counts
SQL> SELECT COUNT(*) FROM dba_users;
```

```
-- For multitenant, capture PDB information
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
```

移行後の検証:

```
-- Verify object counts match
SQL> SELECT object_type, COUNT(*) FROM dba_objects GROUP BY object_type ORDER BY
 object_type;

-- Verify no invalid objects
SQL> SELECT owner, object_type, object_name FROM dba_objects WHERE status =
 'INVALID';

-- Verify tablespace usage
SQL> SELECT tablespace_name, ROUND(SUM(bytes)/1024/1024/1024, 2) AS size_gb
FROM dba_data_files GROUP BY tablespace_name;

-- Verify database size matches
SQL> SELECT SUM(bytes)/1024/1024/1024 AS total_size_gb FROM dba_data_files;

-- For multitenant, verify all PDBs are open
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
```

4. パフォーマンスの検証:

   移行前と移行後のパフォーマンスメトリクスを比較する。

```
-- Capture AWR snapshots before migration (on RDS Custom)
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;

-- After migration (on EC2), compare metrics
SQL> SELECT snap_id, begin_interval_time, end_interval_time
FROM dba_hist_snapshot
ORDER BY snap_id DESC
FETCH FIRST 10 ROWS ONLY;

-- Generate AWR report for comparison
SQL> @?/rdbms/admin/awrrpt.sql
```

   比較する主要なメトリクス:

   - 平均アクティブセッション

- トランザクションあたりの DB 時間
- 1 秒あたりの物理的な読み取り数
- 1 秒あたりの論理読み取り数
- 1 秒あたりの REDO サイズ
- 1 秒あたりのユーザー呼び出し数
- 実行あたりの解析時間

移行後の最適化

1. 移行後、データベースのパフォーマンスを最適化します。
   a. データベースのパフォーマンスチューニング:

      統計を収集します。

```
-- Gather dictionary statistics
SQL> EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;

-- Gather fixed object statistics
SQL> EXEC DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;

-- Gather schema statistics
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SCHEMA_NAME', cascade=>TRUE);

-- For multitenant, gather statistics for each PDB
SQL> ALTER SESSION SET CONTAINER = PDB_NAME;
SQL> EXEC DBMS_STATS.GATHER_DATABASE_STATS(cascade=>TRUE);
```

      メモリパラメータを最適化します。

```
-- Enable Automatic Memory Management (if not already enabled)
SQL> ALTER SYSTEM SET MEMORY_TARGET = 24G SCOPE=SPFILE;
SQL> ALTER SYSTEM SET MEMORY_MAX_TARGET = 28G SCOPE=SPFILE;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;

-- Or use Automatic Shared Memory Management
SQL> ALTER SYSTEM SET SGA_TARGET = 16G SCOPE=SPFILE;
SQL> ALTER SYSTEM SET PGA_AGGREGATE_TARGET = 8G SCOPE=SPFILE;
```

結果キャッシュを設定します。

```
-- Enable result cache for frequently accessed queries
SQL> ALTER SYSTEM SET RESULT_CACHE_MAX_SIZE = 1G;
SQL> ALTER SYSTEM SET RESULT_CACHE_MODE = MANUAL;
```

b. ストレージの最適化:

圧縮を有効にします。

```
-- For tables with infrequent updates
ALTER TABLE large_table MOVE COMPRESS FOR OLTP;

-- For archive/historical data
ALTER TABLE archive_table MOVE COMPRESS FOR ARCHIVE HIGH;

-- Rebuild indexes after compression
ALTER INDEX index_name REBUILD ONLINE;
```

パーティショニングを実装します。

```
-- For large tables, consider partitioning
-- Example: Range partitioning by date
CREATE TABLE sales_partitioned (
    sale_id NUMBER,
    sale_date DATE,
    amount NUMBER
)
PARTITION BY RANGE (sale_date) (
    PARTITION sales_2024 VALUES LESS THAN (TO_DATE('2025-01-01', 'YYYY-MM-DD')),
    PARTITION sales_2025 VALUES LESS THAN (TO_DATE('2026-01-01', 'YYYY-MM-DD')),
    PARTITION sales_2026 VALUES LESS THAN (MAXVALUE)
);
```

c. モニタリングとアラートを実装します。

CloudWatch のカスタムメトリクス:

Oracle メトリクスを CloudWatch に発行するスクリプトを作成します。

```
#!/bin/bash
```

```
# publish_oracle_metrics.sh

INSTANCE_ID=$(ec2-metadata --instance-id | cut -d " " -f 2)
REGION=$(ec2-metadata --availability-zone | cut -d " " -f 2 | sed 's/[a-z]$//')

# Get tablespace usage
TABLESPACE_USAGE=$(sqlplus -s / as sysdba << EOF
SET PAGESIZE 0 FEEDBACK OFF VERIFY OFF HEADING OFF ECHO OFF
SELECT ROUND(MAX(percent_used), 2)
FROM (
     SELECT tablespace_name,
            ROUND((used_space/tablespace_size)*100, 2) AS percent_used
     FROM dba_tablespace_usage_metrics
);
EXIT;
EOF
)

# Publish to CloudWatch
aws cloudwatch put-metric-data \
  --region $REGION \
  --namespace "Oracle/Database" \
  --metric-name "TablespaceUsage" \
  --value $TABLESPACE_USAGE \
  --unit Percent \
  --dimensions InstanceId=$INSTANCE_ID,Database=ORCL
# Add more metrics as needed (sessions, wait events, etc.)
```

CloudWatch アラームを設定します。

```
# Create alarm for high tablespace usage
aws cloudwatch put-metric-alarm \
  --alarm-name oracle-high-tablespace-usage \
  --alarm-description "Alert when tablespace usage exceeds 85%" \
  --metric-name TablespaceUsage \
  --namespace Oracle/Database \
  --statistic Maximum \
  --period 300 \
  --evaluation-periods 2 \
  --threshold 85 \
  --comparison-operator GreaterThanThreshold \
  --alarm-actions arn:aws:sns:region:account-id:topic-name
```

d. セキュリティ強化:

データベースセキュリティ

```
-- Enforce password complexity
ALTER PROFILE DEFAULT LIMIT
    PASSWORD_LIFE_TIME 90
    PASSWORD_GRACE_TIME 7
    PASSWORD_REUSE_TIME 365
    PASSWORD_REUSE_MAX 5
    FAILED_LOGIN_ATTEMPTS 5
    PASSWORD_LOCK_TIME 1;

-- Enable audit
ALTER SYSTEM SET AUDIT_TRAIL = DB, EXTENDED SCOPE=SPFILE;
SHUTDOWN IMMEDIATE;
STARTUP;

-- Audit critical operations
AUDIT ALL ON SYS.AUD$ BY ACCESS;
AUDIT CREATE USER BY ACCESS;
AUDIT DROP USER BY ACCESS;
AUDIT ALTER USER BY ACCESS;
AUDIT CREATE SESSION BY ACCESS WHENEVER NOT SUCCESSFUL;
```

ネットワークセキュリティ

```
# Restrict SSH access
sudo vi /etc/ssh/sshd_config
# Set: PermitRootLogin no
# Set: PasswordAuthentication no

# Configure firewall
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
 address="10.0.0.0/8" port port="1521" protocol="tcp" accept'
sudo firewall-cmd --reload

# Enable Oracle Native Network Encryption
# Edit sqlnet.ora
SQLNET.ENCRYPTION_SERVER = REQUIRED
SQLNET.ENCRYPTION_TYPES_SERVER = (AES256, AES192, AES128)
SQLNET.CRYPTO_CHECKSUM_SERVER = REQUIRED
```

```
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (SHA256, SHA384, SHA512)
```

1. バックアップと復旧戦略:

   包括的なバックアップ戦略を実装します。

```
#!/bin/bash
# rman_backup.sh - Daily incremental backup script

export ORACLE_HOME=/u01/app/oracle/product/19.0.0/dbhome_1
export ORACLE_SID=ORCL
export PATH=$ORACLE_HOME/bin:$PATH

# Backup to local disk
rman target / << EOF
RUN {
    ALLOCATE CHANNEL ch1 DEVICE TYPE DISK FORMAT '/u01/app/oracle/backup/inc_%U';
    BACKUP INCREMENTAL LEVEL 1 DATABASE PLUS ARCHIVELOG;
    DELETE NOPROMPT OBSOLETE;
    CROSSCHECK BACKUP;
    DELETE NOPROMPT EXPIRED BACKUP;
}
EXIT;
EOF

# Copy backups to S3
aws s3 sync /u01/app/oracle/backup/ s3://my-oracle-backups/$(date +%Y%m%d)/ \
    --storage-class STANDARD_IA \
    --exclude "*" \
    --include "inc_*" \
    --include "arch_*"

# Clean up local backups older than 7 days
find /u01/app/oracle/backup/ -name "inc_*" -mtime +7 -delete
find /u01/app/oracle/backup/ -name "arch_*" -mtime +7 -delete
```

   cron を使用してバックアップをスケジュールします。

```
# Edit crontab
crontab -e

# Add backup schedule
```

```
# Full backup on Sunday at 2 AM
0 2 * * 0 /home/oracle/scripts/rman_full_backup.sh >> /home/oracle/logs/
backup_full.log 2>&1

# Incremental backup Monday-Saturday at 2 AM
0 2 * * 1-6 /home/oracle/scripts/rman_incremental_backup.sh >> /home/oracle/logs/
backup_inc.log 2>&1

# Archive log backup every 4 hours
0 */4 * * * /home/oracle/scripts/rman_archivelog_backup.sh >> /home/oracle/logs/
backup_arch.log 2>&1
```

コスト最適化

1. 適切なサイジング:

移行後、コストをモニタリングして最適化します。

- AWS Cost Explorer を使用して EC2 と EBS のコストを分析する
- インスタンスタイプのレコメンデーションについて、AWS Compute Optimizer を有効にする
- CloudWatch メトリクスを確認して、十分に活用されていないリソースを特定する
- 予測可能なワークロードに対してリザーブドインスタンスまたは Savings Plans を検討する

2. ストレージの最適化:

- S3 バックアップのライフサイクルポリシーを実装する (30 日後に Glacier に移行する)
- 定期的に未使用の EBS スナップショットを削除する
- gp2 の代わりに gp3 を使用して、同じパフォーマンスでコストを削減する
- 移行の完了後にバックアップボリュームをデタッチおよび削除する

3. Automation

- 営業時間外の非本番稼働用データベースの開始/停止を自動化する
- パッチ管理に AWS Systems Manager Patch Manager を使用する
- Data Guard を使用している場合、リードレプリカの自動スケーリングを実装する

結論

この規範的ガイダンスでは、Oracle データベースを Amazon RDS Custom for Oracle から Amazon EC2 のセルフマネージド Oracle データベースに移行するための詳細な移行戦略について説明しました。RDS Custom for Oracle サービス廃止が、2027 年 3 月 31 日に有効となるため、移行を事前に計画して実行することが重要です。

重要なポイント

移行オプション

- RMAN アクティブ複製: ほとんどの移行に最適で、複製時にソースデータベースをオンラインに保ち、アプリケーションのリダイレクトに必要なのはごく短いカットオーバーウィンドウのみです。
- Oracle Data Guard: ほぼゼロのダウンタイムを必要とするミッションクリティカルなワークロードに最適で、継続的な同期と組み込みのスイッチオーバー機能を提供します

アーキテクチャのサポート:

- どちらの移行オプションも、非 CDB (従来型の単一インスタンス) アーキテクチャとマルチテナント (PDB を含む CDB) アーキテクチャをサポートしています
- マルチテナントの場合、どちらのメソッドもすべての PDB を含む CDB 全体を 1 回のオペレーションで自動的に処理します
- PDB は、移行後に手動でのオープンと自動オープン設定が必要です

重要な成功要因:

- ソースとターゲット間の適切なネットワーク設定と接続
- 正確なバージョンの互換性 (メジャーバージョン、マイナーバージョン、リリース更新、およびワンオフパッチ)
- データ転送 (RMAN) または REDO 配布 (Data Guard) に適切なネットワーク帯域幅
- RMAN アクティブ複製がソースをオンラインに保つため、必要なのはごく短いカットオーバーのみであることを理解する
- ソースを廃止する前に徹底的なテストと検証を行う
- バックアップ、モニタリング、セキュリティ設定を含む包括的な移行後タスク

次のステップ:

1. データベースアーキテクチャを評価する (非 CDB またはマルチテナント)
2. ダウンタイムの許容度と複雑さの要件に基づいて、適切な移行オプションを選択する
3. EC2 インスタンスのセットアップやネットワーク設定など、すべての前提条件となるステップを完了する
4. 選択したオプションの詳細な移行ステップに従う
5. 徹底的な検証とテストを実行する
6. 移行後のタスクを完了して本番環境の準備を整える
7. 検証が成功したら RDS Custom インスタンスを廃止する

その他のリソース。

- Amazon RDS Custom for Oracle ユーザーガイド
- Oracle Database のドキュメント
- Oracle RMAN のドキュメント
- Oracle Data Guard のドキュメント
- AWS Database Migration Service
- AWS 規範的ガイダンス

サポート

移行に関するサポートについては、以下までお問い合わせください。

- AWS マネジメントコンソールから AWS サポートに問い合わせる
- データベース固有の質問について、Oracle サポートに問い合わせる

ドキュメント情報

最終更新日: 2026 年 3 月

寄稿者:

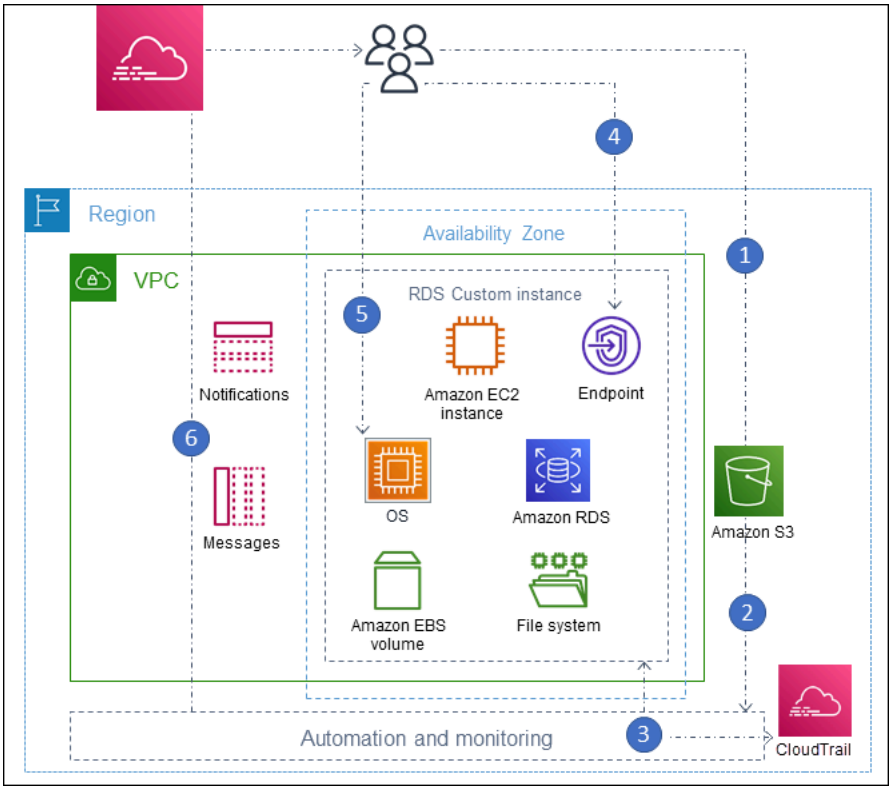
- Amazon Web Services、データベーススペシャリストソリューションアーキテクト、Sharath Chandra Kampili
- Amazon Web Services、データベーススペシャリストソリューションアーキテクト、Ibrahim Emara

- Amazon Web Services、データベーススペシャリストソリューションアーキテクト、Vetrivel Subramani

RDS Custom for Oracle ワークフロー

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

次の図表は、RDS Custom for Oracle の一般的なワークフローを示しています。



ステップは次のとおりです。

1. Amazon S3 バケットにデータベースソフトウェアをアップロードします。

   詳細については、「ステップ 3: Amazon S3 へのインストールファイルをアップロードする」を参照してください。

2. メディアから RDS Custom for Oracle カスタムエンジンバージョン (CEV) を作成します。

   CDB アーキテクチャまたは従来の非 CDB アーキテクチャを選択します。詳細については、「CEV の作成」を参照してください。
3. CEV から RDS Custom for Oracle DB インスタンスを作成します。

   詳細については、「RDS Custom for Oracle DB インスタンスを作成する」を参照してください。
4. アプリケーションを DB インスタンスエンドポイントに接続します。

   詳細については、「SSH を使用した RDS Custom DB インスタンスへの接続」および「Session Manager を使用して RDS Custom DB インスタンスに接続する」を参照してください。
5. (オプション) ホストにアクセスしてソフトウェアをカスタマイズします。
6. RDS Custom オートメーションによって生成された通知とメッセージを監視します。

データベースインストールファイル

Amazon RDS と RDS Custom の主な違いは、お客様のメディアに対する責任です。フルマネージドサービスである Amazon RDS は、Amazon マシンイメージ(AMI) とデータベースソフトウェアを提供します。Amazon RDS データベースソフトウェアがプリインストールされているため、データベースエンジンとバージョンを選択し、データベースを作成するだけです。

RDS Custom では、ご自身で独自のメディアをご用意ください。カスタムエンジンバージョンを作成すると、RDS Custom が指定したメディアがインストールします。RDS Custom メディアには、データベースインストールファイルとパッチが含まれています。このサービスモデルはBYOM (Bring Your Own Media = 自分のメディアを持参する) と呼ばれています。

Amazon RDS Custom for Oracle のカスタムエンジンバージョン

RDS Custom for Oracle カスタムエンジンバージョン(CEV) は、データベースバージョンと AMI のバイナリボリュームスナップショットです。デフォルトでは、RDS Custom for Oracle は Amazon EC2 が公開している最新の AMI を使用します。既存の AMI の再利用も選択できます。

CEV マニフェスト

Oracle データベースインストールファイルを Oracle からダウンロードしたら、それらを Amazon S3 バケットにアップロードします。CEV を作成するときには、JSON ドキュメントで「CEV マニフェスト」というファイル名を指定します。RDS Custom for Oracle は、指定されたファイルと AMI を使用して CEV を作成します。

RDS Custom for Oracle には、サポートされている Oracle データベースの各リリース用に、推奨する.zip ファイルが含まれた JSON マニフェストテンプレートが用意されています。例えば、次のテンプレートは 19.17.0.0.0 RU 用です。

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p34419443_190000_Linux-x86-64.zip",
        "p34411846_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
        "p31335037_190000_Linux-x86-64.zip",
        "p32327201_190000_Linux-x86-64.zip",
        "p33613829_190000_Linux-x86-64.zip",
        "p34006614_190000_Linux-x86-64.zip",
        "p34533061_190000_Linux-x86-64.zip",
        "p34533150_190000_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29213893_1917000DBRU_Generic.zip",
        "p33125873_1917000DBRU_Linux-x86-64.zip",
        "p34446152_1917000DBRU_Linux-x86-64.zip"
    ]
}
```

JSON マニフェストでインストールパラメータを指定することもできます。例えば、Oracle ベース、Oracle ホーム、および UNIX/Linux ユーザーとグループの ID と名前にデフォルト以外の値を設定できます。詳細については、「CEV マニフェストの JSON フィールド」を参照してください。

CEV 命名フォーマット

お客様の指定する文字列で RDS Custom for Oracle CEV の名前を付けます。Oracle Database のリリースに応じて、名前の形式は次のようになります。

- 19.*customized_string*

- 18.*customized_string*
- 12.2.*customized_string*
- 12.1.*customized_string*

ユーザーネームに使用できるのは、1 ~ 50 文字の英数字、アンダースコア (- . _)、ダッシュ、ピリオドのみです。例えば、CEV19.my_cev1 に名前を付けることができます。

RDS Custom for Oracle の Oracle マルチテナントアーキテクチャ

Oracle マルチテナントアーキテクチャでは、Oracle データベースをコンテナデータベース (CDB) として機能させることができます。CDB には、お客様が作成したプラグ可能なデータベース (PDB) を 0 個、1 個、または複数含みます。PDB は、スキーマとオブジェクトの移植可能なコレクションであり、アプリケーションには従来の非 CDB として表示されます。

RDS Custom for Oracle CEV を作成する際、CDB または非 CDB アーキテクチャのいずれかを指定します。RDS Custom for Oracle CDB を作成できるのは、作成に使用する CEV が Oracle マルチテナントアーキテクチャを使用している場合のみです。詳細については、「Amazon RDS Custom for Oracle のカスタムエンジンバージョンでの作業」を参照してください。

RDS Custom for Oracle の DB インスタンスを作成する

CEV を作成すると、使用できるようになります。複数の CEV を作成でき、どの CEV からも複数の RDS Custom for Oracle DB インスタンスを作成できます。また、CEV のステータスを変更し、CEV を使用可能または非アクティブにすることができます。

RDS Custom for Oracle DB インスタンスは、Oracle マルチテナントアーキテクチャ (custom-oracle-ee-cdb や custom-oracle-se2-cdb エンジンタイプ) または従来の非 CDB アーキテクチャ (custom-oracle-ee や custom-oracle-se2 エンジンタイプ) を使用して作成できます。コンテナデータベース (CDB) を作成すると、そのデータベースには 1 つのプラグ可能なデータベース (PDB) と 1 つの PDB シードが含まれます。Oracle SQL を使用して、手動で追加の PDB を作成できます。

RDS Custom for Oracleの DB インスタンスを作成するには、create-db-instanceコマンドを使用します。このコマンドで、使用するCEVを指定します。この手順は、Amazon RDS DB インスタンスの作成と似ています。ただし、一部のパラメータは異なります。詳細については、「Amazon RDS Custom for Oracle DB インスタンスの設定」を参照してください。

データベース接続

Amazon RDS DB インスタンスと同様に、RDS Custom DB インスタンスは 仮想プライベートクラウド (VPC) 内にあります。アプリケーションは、Oracle リスナーを使用して Oracle データベースに接続します。

データベースが CDB の場合は、リスナーを使用して `L_RDSCDB_001` を CDB ルートと PDB に接続できます。非 CDB を CDB に接続する場合は、移行したアプリケーションが同じ設定を維持するように `USE_SID_AS_SERVICE_LISTENER = ON` を設定してください。

非 CDB に接続する場合、マスターユーザーは非 CDB のユーザーです。CDB に接続する場合、マスターユーザーは PDB のユーザーです。CDB ルートに接続するには、ホストにログインして SQL クライアントを起動し、SQL コマンドを使用して管理ユーザーを作成します。

RDS Custom のカスタマイズ

RDS Custom ホストにアクセスして、ソフトウェアをインストールまたはカスタマイズできます。変更と RDS カスタムオートメーションの間の競合を回避するために、指定した期間オートメーションを一時停止できます。この期間中、RDS Custom はモニタリングまたはインスタンスのリカバリを実行しません。RDS Custom は期間終了時にフルオートメーションを再開します。(詳しくは、「RDS Custom DB インスタンスの一時停止と再開」を参照してください。)

Amazon RDS Custom for Oracle のデータベースアーキテクチャ

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle は、Oracle マルチテナントアーキテクチャと非マルチテナントアーキテクチャの両方をサポートしています。

トピック

- サポートされている Oracle データベースアーキテクチャ
- サポートされているエンジンタイプ

- Oracle マルチテナントアーキテクチャでサポートされる機能

サポートされている Oracle データベースアーキテクチャ

Oracle マルチテナントアーキテクチャ (CDB アーキテクチャ) により、Oracle データベースをコンテナデータベース (CDB) として機能させることができます。CDB にはプラガブルデータベース (PDB) が含まれます。PDB は、スキーマとオブジェクトのコレクションであり、アプリケーションには従来の Oracle データベースとして表示されます。詳細については、「Oracle Multitenant 管理者ガイド」の「マルチテナントアーキテクチャの概要」を参照してください。

CDB アーキテクチャと非 CDB アーキテクチャは相互に排他的です。Oracle データベースが CDB でない場合、そのデータベースは非 CDB なので、PDB を含めることはできません。RDS Custom for Oracle では、Oracle Database 19c のみが CDB アーキテクチャをサポートしています。したがって、以前の Oracle データベースリリースを使用して DB インスタンスを作成する場合、作成できるのは非 CDB のみです。詳細については、「マルチテナントアーキテクチャの考慮事項」を参照してください。

サポートされているエンジンタイプ

Amazon RDS Custom for Oracle CEV または DB インスタンスを作成するときは、CDB エンジンタイプまたは非 CDB エンジンタイプを選択します。

- `custom-oracle-ee-cdb` および `custom-oracle-se2-cdb`

  これらのエンジンタイプは、Oracle マルチテナントアーキテクチャを指定します。このオプションは、Oracle Database 19c でのみ使用できます。マルチテナントアーキテクチャを使用して RDS for Oracle DB インスタンスを作成すると、CDB には以下のコンテナが含まれます。

  - CDB ルート (`CDB$ROOT`)
  - PDB シード (`PDB$SEED`)
  - 初期 PDB

  Oracle SQL コマンド `CREATE PLUGGABLE DATABASE` を使用して、さらに多くの PDB を作成できます。RDS API を使用して PDB を作成または削除することはできません。

- `custom-oracle-ee` および `custom-oracle-se2`

  これらのエンジンタイプは、従来の非 CDB アーキテクチャを指定します。非 CDB には、プラグ可能なデータベース (PDB) を含めることはできません。

詳細については、「マルチテナントアーキテクチャの考慮事項」を参照してください。

Oracle マルチテナントアーキテクチャでサポートされる機能

RDS Custom for Oracle CDB インスタンスは、次の機能をサポートしています。

- バックアップ
- バックアップからのリストアとポイントタイムリストア (PITR)
- リードレプリカ
- マイナーバージョンのアップグレード

RDS Custom for Oracle の機能の可用性とサポート

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

このトピックでは、クイックリファレンスとして、RDS Custom for Oracle の機能の可用性とサポートについての概要を参照できます。

トピック

- AWS リージョン およびデータベースバージョンは RDS Custom for Oracleをサポートします
- RDS Custom for Oracle のデータベースバージョンのサポート
- RDS Custom for Oracle のエディションとライセンスのサポート
- RDS Custom for Oracle での DB インスタンスクラスのサポート
- RDS Custom for Oracle のオプショングループのサポート

AWS リージョン およびデータベースバージョンは RDS Custom for Oracleをサポートします

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。RDS Custom for Oracle でのバージョンとリージョンの可用性の詳細については、「RDS Custom でサポートされているリージョンと DB エンジン」を参照してください。

RDS Custom for Oracle のデータベースバージョンのサポート

RDS Custom for Oracle は Oracle データベースの次のバージョンをサポートします。

- Oracle Database 19c
- Oracle Database 18c
- Oracle Database 12c Release 2 (12.2)
- Oracle Database 12c Release 1 (12.1)

RDS Custom for Oracle のエディションとライセンスのサポート

RDS Custom for Oracle は、BYOL モデルで Enterprise Edition (EE) と Standard Edition 2 (SE2) をサポートしています。

Standard Edition 2 には、以下の制約事項があることに注意してください。

- Oracle Data Guard はサポートされていません。したがって、Oracle リードレプリカを作成することはできません。
- vCPU 数が 16 以下の DB インスタンスクラス (最大 4xlarge) のみを使用できます。
- Standard Edition 2 の CDB インスタンスは、最大 3 つのテナントデータベースをサポートします。
- Enterprise Edition と Standard Edition 2 の間でデータを移行することはできません。

RDS Custom for Oracle での DB インスタンスクラスのサポート

RDS Custom for Oracle では、次の DB インスタンスクラスをサポートしています。Standard Edition 2 で DB インスタンスを作成する場合、vCPU 数が 16 以下のインスタンスクラス (最大 4xlarge) のみを使用できます。

タイプ	サイズ
db.m7i	db.m7i.large |db.m7i.xlarge |db.m7i.2xlarge |db.m7i.4xlarge |db.m7i.8xlarge |db.m7i.12xlarge |db.m7i.16xlarge |db.m7i.24xlarge |db.m7i.48xlarge |db.m7i.metal-24xl |db.m7i.metal-48xl
db.m6i	db.m6i.large |db.m6i.xlarge |db.m6i.2xlarge |db.m6i.4xlarge |db.m6i.8xlarge |db.m6i.12xlarge |db.m6i.16xlarge |db.m6i.24xlarge |db.m6i.32xlarge |db.m6i.metal
db.m6id	db.m6id.metal
db.m6in	db.m6in.metal
db.m5	db.m5.large |db.m5.xlarge |db.m5.2xlarge |db.m5.4xlarge | db.m5.8xlarge |db.m5.12xlarge |db.m5.16xlarge |db.m5.24xlarge
db.r7i	db.r7i.large |db.r7i.xlarge |db.r7i.2xlarge |db.r7i.4xlarge |db.r7i.8xlarge |db.r7i.12xlarge |db.r7i.16xlarge |db.r7i.24xlarge |db.r7i.48xlarge |db.r7i.metal-24xl |db.r7i.metal-48xl
db.r6i	db.r6i.large |db.r6i.xlarge |db.r6i.2xlarge |db.r6i.4xlarge |db.r6i.8xlarge |db.r6i.12xlarge |db.r6i.16xlarge |db.r6i.24xlarge |db.r6i.32xlarge |db.r6i.metal
db.r6id	db.r6id.metal
db.r6in	db.r6in.metal
db.r5b	db.r5b.large |db.r5b.xlarge |db.r5b.2xlarge |db.r5b.4xlarge |db.r5b.8xlarge |db.r5b.12xlarge |db.r5b.16xlarge |db.r5b.24xlarge
db.r5	db.r5.large |db.r5.xlarge |db.r5.2xlarge |db.r5.4xlarge | db.r5.8xlarge |db.r5.12xlarge |db.r5.16xlarge |db.r5.24xlarge

タイプ	サイズ
db.x2iedn	db.x2iedn.xlarge | db.x2iedn.2xlarge | db.x2iedn.4xlarge | db.x2iedn.8xlarge | db.x2iedn.16xlarge | db.x2iedn.24xlarge | db.x2iedn.32xlarge | db.x2iedn.metal
db.x2idn	db.x2idn.metal
db.x2iezn	db.x2iezn.2xlarge | db.x2iezn.4xlarge | db.x2iezn.6xlarge | db.x2iezn.8xlarge | db.x2iezn.12xlarge | db.x2iezn.metal
db.t3	db.t3.medium | db.t3.large | db.t3.xlarge | db.t3.2xlarge

RDS Custom for Oracle のオプショングループのサポート

RDS Custom for Oracle DB インスタンスを作成または変更するときに、オプショングループを指定できます。詳細については、「RDS Custom for Oracle のオプショングループ使用する」を参照してください。

RDS Custom for Oracle の要件と制限

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

このトピックでは、クイックリファレンスとして、Amazon RDS Custom for Oracle の可用性と制限についての要約を参照できます。

トピック

- RDS Custom for Oracle の一般的な要件
- RDS Custom for Oracle の一般的な制限事項
- RDS Custom for Oracle の CEV と AMI の制限

- 作成および変更ワークフローでサポートされていない設定
- AWS アカウント の DB インスタンスクォータ

RDS Custom for Oracle の一般的な要件

Amazon RDS Custom for Oracle については、次の要件に従ってください。

- RDS Custom for Oracle のインストールファイルとパッチのサポート対象リストをダウンロードするための、My Oracle Support と Oracle Software Delivery Cloud へのアクセス権がある。不明なパッチを使用すると、カスタムエンジンバージョン (CEV) の作成に障害が発生します。この場合、RDS Custom サポートチームに連絡して、不足パッチの追加を依頼してください。詳細については、「ステップ 2: Oracle Software Delivery Cloud からデータベースインストールファイルおよびパッチをダウンロードする」を参照してください。
- Amazon S3 へのアクセス権がある。このサービスが必要な理由は次のとおりです。
  - Oracle インストールファイルを S3 バケットにアップロードします。アップロードしたインストールファイルは、RDS Custom CEV の作成に使用します。
  - RDS Custom for Oracle は、内部で定義された S3 バケットからダウンロードされたスクリプトを使用して DB インスタンスでアクションを実行します。これらのスクリプトは、オンボーディングと RDS カスタムオートメーションに必要です。
  - RDS Custom for Oracle は、特定のファイルをお客様のアカウントにある S3 バケットにアップロードします。これらのバケットは、`do-not-delete-rds-custom-`*`account_id`*`-`*`region`*`-`*`uuid`* という命名形式を使用します。例えば、`do-not-delete-rds-custom-123456789012-us-east-1-12a3b4` という名前のバケットがあるとします。

  詳細については、「ステップ 3: Amazon S3 へのインストールファイルをアップロードする」および「CEV の作成」を参照してください。
- RDS Custom for Oracle での DB インスタンスクラスのサポート にリストされている DB インスタンスクラスを使用して、RDS Custom for Oracle DB インスタンスを作成する。
- RDS Custom for Oracle DB インスタンスは、Oracle Linux 8 (推奨) または Oracle Linux 7 を実行します。Oracle Linux 7 が必要な場合は、サポート にお問い合わせください。詳細については、「RDS Custom for Oracle のデータベースアップグレードに関する考慮事項」を参照してください。
- Amazon EBS ストレージ用の gp2、gp3、または io1 ソリッドステートドライブを指定する。最大ストレージサイズは 64 TiB です。

- RDS Custom for Oracle DB インスタンスを作成するための AWS KMS キーがある。詳細については、「ステップ 1: 対称暗号化 AWS KMS キーを作成または再利用する」を参照してください。
- RDS Custom for Oracle DB インスタンスの作成に必要な AWS Identity and Access Management (IAM) ロールとインスタンスプロファイルがある。詳細については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。
- CEV または RDS Custom DB インスタンスを作成する AWS Identity and Access Management (IAM) ユーザーに、IAM、CloudTrail、Amazon S3 への必要なアクセス許可がある。

  詳細については、「ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する」を参照してください。
- 仮想プライベートクラウド (VPC) とセキュリティグループの設定を指定する。詳細については、「ステップ 6: RDS Custom for Oracle 用に VPC を設定する」を参照してください。
- RDS Custom for Oracle が他の AWS のサービス にアクセスするために使用できるネットワーク構成を指定する。特定の要件については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。

RDS Custom for Oracle の一般的な制限事項

RDS Custom for Oracle には以下の制限が適用されます。

- 既存の RDS Custom for Oracle DB インスタンスの DB インスタンス識別子を変更することはできません。
- Oracle Database 19c 以外のリリースでは Oracle マルチテナントアーキテクチャを指定できません。
- 単一の RDS Custom for Oracle インスタンスでは、複数の Oracle データベースインスタンスを作成できます。
- RDS Custom for Oracle の DB インスタンスまたはその基盤となる Amazon EC2 インスタンスを停止することはできません。RDS Custom for Oracle DB インスタンスの請求は停止できません。
- RDS Custom for Oracle は自動メモリ管理のみをサポートしているため、自動共有メモリ管理は使用できません。詳細は、Oracle Database 管理者ガイドの「自動メモリ管理」を参照してください。
- プライマリ DB インスタンスの `DB_UNIQUE_NAME` は変更しないでください。名前を変更すると、復元オペレーションが停止します。
- 同じリージョンで同時に 20 個を超えるスナップショットコピーを作成することはできません。

- RDS Custom for Oracle DB インスタンスに `describe-reserved-db-instances` API を使用することはできません。

RDS Custom for Oracle DB インスタンスの変更に特有の制限事項については、「RDS Custom for Oracle DB インスタンスを変更する」を参照してください。レプリケーションの制限事項については、「RDS Custom for Oracle レプリケーションの一般的な制限事項」を参照してください。

RDS Custom for Oracle の CEV と AMI の制限

RDS Custom for Oracle CEV と AMI には以下の制限が適用されます。

- RDS Custom for Oracle CEV で使用する独自の AMI を提供することはできません。指定できるのは、Oracle Linux 8 を使用するデフォルトの AMI、または RDS Custom for Oracle CEV で以前に使用したことのある AMI だけです。

> **Note**
>
> 共通脆弱性とエクスポージャーが検出されると、RDS Custom for Oracle は新しいデフォルト AMI をリリースします。一定のスケジュールや保証はありません。RDS Custom for Oracle は、30 日ごとに新しいデフォルト AMI を公開する傾向があります。

- 別の AMI を使用するように CEV を変更することはできません。
- `custom-oracle-ee` または `custom-oracle-se2` エンジンタイプを使用する CEV から CDB インスタンスを作成することはできません。CEV では、`custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` を使用する必要があります。
- RDS Custom for Oracle では、現在、RDS API コールを使用して RDS Custom for Oracle DB インスタンスの OS をアップグレードすることはできません。回避策として、次のコマンドを使用して OS を手動で更新できます: `sudo yum update --security`

作成および変更ワークフローでサポートされていない設定

RDS Custom for Oracle DB インスタンスを作成または変更する場合、次の操作を実行することはできません。

- DB インスタンスクラスの CPU コア数とコアごとのスレッド数を変更します。
- ストレージのオートスケーリングを有効にします。
- バックアップ保持機能を0に設定します。

- Kerberos 認証の設定
- 独自の DB パラメータグループまたはオプショングループを指定します。
- Performance Insights をオンにします。
- 自動マイナーバージョンアップグレードを実行します。

AWS アカウント の DB インスタンスクォータ

RDS Custom と Amazon RDS DB インスタンスの合計数が、クォータ制限を超えていないことを確認してください。例えば、Amazon RDS のクォータが 40 DB インスタンスの場合、20のRDS Custom for Oracle DB インスタンスと 20 の Amazon RDS DB インスタンスを持つことができます。

Amazon RDS Custom for Oracle の環境設定

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS Custom for Oracle DB インスタンスを作成する前に、次のタスクを実行します。

トピック

- ステップ 1: 対称暗号化 AWS KMS キーを作成または再利用する
- ステップ 2: AWS CLI をダウンロードしてインストールする
- ステップ 3: RDS Custom for Oracle の CloudFormation テンプレートを抽出する
- ステップ 4: RDS Custom for Oracle 用に IAM を設定する
- ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する
- ステップ 6: RDS Custom for Oracle 用に VPC を設定する

ステップ 1: 対称暗号化 AWS KMS キーを作成または再利用する

カスタマーマネージドキーは、お客様が作成、所有、管理している AWS アカウントの AWS KMS keys です。RDS Custom にはカスタマーマネージド対称暗号化 KMS キーが必要です。RDS Custom

for Oracle DB インスタンス作成の際、KMS キー 識別子を指定します。(詳しくは、「Amazon RDS Custom for Oracle DB インスタンスの設定」を参照してください。)

次のオプションがあります。

- AWS アカウント に既存のカスタマーマネージド KMS キーがある場合は、RDS Custom で使用できます。これ以上の操作は不要です。
- RDS Custom エンジンのカスタマーマネージド対称暗号化 KMS キーを既に作成している場合は、同じ KMS キーを再利用できます。これ以上の操作は不要です。
- アカウントに既存のカスタマーマネージド対称暗号化 KMS キーがない場合は、AWS Key Management Service デベロッパーガイドの「キーの作成」の手順に従って KMS キーを作成します。
- CEV または RDS Custom DB インスタンスを作成していて、KMS キーが別の AWS アカウント にある場合は、必ず AWS CLI を使用してください。クロスアカウントの KMS キーでは AWS コンソールを使用できません。

> **Important**
>
> RDS Custom はAWSマネージド KMS キーに対応していません。

対称暗号化キーが、IAM インスタンスプロファイルの AWS Identity and Access Management (IAM) ロールに `kms:Decrypt` および `kms:GenerateDataKey` オペレーションへのアクセスを許可していることを確認してください。アカウントに新しい対称暗号化キーがある場合、変更は必要ありません。それ以外の場合は、対称暗号化キーのポリシーがこれらのオペレーションへのアクセスを許可していることを確認してください。

詳細については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。

RDS Custom for Oracle の IAM の設定の詳細については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。

ステップ 2: AWS CLI をダウンロードしてインストールする

AWSは、RDS Custom 機能を使用するためのコマンドラインインターフェイスを提供します。AWS CLI のバージョン 1 またはバージョン 2 を使用できます。

AWS CLI のダウンロードおよびインストールについては、「AWS CLI の最新バージョンを使用してインストールまたは更新を行う」を参照してください。

次のいずれかに該当する場合には、この手順をスキップします。

- AWS マネジメントコンソール からのみ RDS Custom にアクセスする予定です。
- Amazon RDS または異なる RDS Custom エンジンの AWS CLI を既にダウンロードしています。

ステップ 3: RDS Custom for Oracle の CloudFormation テンプレートを抽出する

セットアップを簡素化するために、CloudFormation テンプレートを使用して CloudFormation スタックを作成することを強くお勧めします。IAM と VPC を手動で設定する場合、このステップは省略します。

トピック

- ステップ 3a: CloudFormation テンプレートファイルをダウンロードする
- ステップ 3b: custom-oracle-iam.json を抽出する
- ステップ 3c: custom-vpc.json を抽出する

ステップ 3a: CloudFormation テンプレートファイルをダウンロードする

CloudFormation テンプレートとは、スタックを構成する AWS リソースの宣言です。テンプレートは JSON ファイルとして保存されます。

CloudFormation テンプレートファイルをダウンロードするには

1. リンクのコンテキスト (右クリック) メニューから[custom-oracle-iam.zip] を開き、[Save Link As] (名前を付けてリンクを保存) を選択します。
2. ファイルをコンピュータに保存します。
3. リンクの [custom-vpc.zip] に対して前のステップを繰り返します。

   VPC を RDS Custom 用に既に設定している場合は、このステップをスキップします。

ステップ 3b: custom-oracle-iam.json を抽出する

ダウンロードした `custom-oracle-iam.zip` ファイルを開き、ファイル `custom-oracle-iam.json` を抽出します。ファイルの先頭は次のようになります。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "EncryptionKey": {
      "Type": "String",
      "Default": "*",
      "Description": "KMS Key ARN for encryption of data managed by RDS Custom and by
 DB Instances."
    }
  },
  "Resources": {
    "RDSCustomInstanceServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "RoleName": { "Fn::Sub": "AWSRDSCustomInstanceRole-${AWS::Region}" },
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "sts:AssumeRole",
              "Effect": "Allow",
              "Principal": {
                "Service": "ec2.amazonaws.com"
              }
            }
          ]
        },...
```

ステップ 3c: custom-vpc.json を抽出する

> **Note**
>
> RDS Custom for Oracle の VPC が既に設定されている場合、このステップは省略します。詳細については、「RDS Custom for Oracle 用に VPC を手動で設定する」を参照してください。

ダウンロードした custom-vpc.zip ファイルを開き、ファイル custom-vpc.json を抽出します。ファイルの先頭は次のようになります。

```
{
```

```
 "AWSTemplateFormatVersion": "2010-09-09",
 "Parameters": {
   "PrivateVpc": {
     "Type": "AWS::EC2::VPC::Id",
     "Description": "Private VPC Id to use for RDS Custom DB Instances"
   },
   "PrivateSubnets": {
     "Type": "List<AWS::EC2::Subnet::Id>",
     "Description": "Private Subnets to use for RDS Custom DB Instances"
   },
   "RouteTable": {
     "Type": "String",
     "Description": "Route Table that must be associated with the PrivateSubnets and
used by S3 VPC Endpoint",
     "AllowedPattern": "rtb-[0-9a-z]+"
   }
 },
 "Resources": {
   "DBSubnetGroup": {
     "Type": "AWS::RDS::DBSubnetGroup",
     "Properties": {
       "DBSubnetGroupName": "rds-custom-private",
       "DBSubnetGroupDescription": "RDS Custom Private Network",
       "SubnetIds": {
         "Ref": "PrivateSubnets"
       }
     }
   },...
```

ステップ 4: RDS Custom for Oracle 用に IAM を設定する

IAM ロールまたは IAM ユーザー (IAM エンティティと呼ばれます) を使用して、コンソールまたは AWS CLI を使用して RDS Custom DB インスタンスを作成します。この IAM エンティティには、インスタンス作成に必要なアクセス許可が付与されている必要があります。

IAM は、CloudFormation または手動ステップを使用して設定できます。

> **Important**
>
> CloudFormation を使用して RDS Custom for Oracle 環境を設定することを強くお勧めします。この方法は、最も簡単でエラーが起こりにくいものです。

トピック

- CloudFormation を使用して IAM を設定する
- IAM ロールおよびインスタンスプロファイルをマニュアルで作成する

CloudFormation を使用して IAM を設定する

IAM 用の CloudFormation テンプレートを使用すると、次の必要なリソースが作成されます。

- `AWSRDSCustomInstanceProfile-region` という名前のインスタンスプロファイル
- `AWSRDSCustomInstanceRole-region` という名前のサービスロール
- サービスロールにアタッチしている `AWSRDSCustomIamRolePolicy` という名前のアクセスポリシー

CloudFormation を使用して IAM を設定するには

1. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. 「スタックの作成」 ウィザードを起動し、スタックの作成を選択します。
3. [Create stack] (スタックの作成) ページで、次の手順を実行します。
   a. [Prepare template] (テンプレートの準備) の [Template is ready] (テンプレートの準備が完了しました) を選択します。
   b. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。
   c. [ファイルを選択] に移動して、[custom-oracle-iam.json] を選択します。
   d. [次へ] を選択します。
4. [スタックの詳細を指定] ページで、以下を実行します。
   a. [スタック名] に「**custom-oracle-iam**」と入力します。
   b. [次へ] を選択します。
5. [スタックオプションの設定] ページで、[次へ] をクリックします。
6. [Review custom-oracle-iam] (カスタムオラクル-IAM のレビュー) ページで、以下の作業を行います。
   a. [CloudFormation がカスタム名で IAM リソースを作成する可能性があることに同意する] チェックボックスを選択します。
   b. [Submit] を選択してください。

CloudFormation は、 RDS Custom for Oracle が必要とする IAM ロールを作成します。左側のパネルで、[custom-oracle-iam] に [CREATE_COMPLETE] と表示されたら、次のステップに進みます。

7. 左のパネルで、[custom-oracle-iam] を選択します。右のパネルで、以下の操作を行います。

   a. [スタック情報] を選択します。スタックの ID は、arn:aws:cloudformation:***region***:***account-no***:stack/custom-oracle-iam/***identifier*** という形式です。

   b. [リソース] をクリックします。次のように表示されます。

      - AWSRDSCustomInstanceProfile-***region*** という名前のインスタンスプロファイル
      - AWSRDSCustomInstanceRole-***region*** という名前のサービスロール

      RDS Custom DB インスタンスを作成するときは、インスタンスプロファイル ID を指定する必要があります。

IAM ロールおよびインスタンスプロファイルをマニュアルで作成する

CloudFormation を使用すると、設定が最も簡単になります。ただし、IAM は手動で設定することもできます。手動セットアップの場合は、以下を実行します。

- ステップ 1: AWSRDSCustomInstanceRoleForRdsCustomInstance IAM ロールを作成する.
- ステップ 2: AWSRDSCustomInstanceRoleForRdsCustomInstance にアクセスポリシーを追加します。.
- ステップ 2: AWSRDSCustomInstanceRoleForRdsCustomInstance にアクセスポリシーを追加します。.
- ステップ 4: AWSRDSCustomInstanceRoleForRdsCustomInstance を AWSRDSCustomInstanceProfile に追加する.

ステップ 1: AWSRDSCustomInstanceRoleForRdsCustomInstance IAM ロールを作成する

このステップでは、命名形式 `AWSRDSCustomInstanceRole-`***region*** を使用してロールを作成します。信頼ポリシーを使用して、Amazon EC2 がロールを引き受けることができます。次の例では、DB インスタンスを作成する環境変数 `$REGION` を AWS リージョン に設定していることを前提としています。

```
aws iam create-role \
  --role-name AWSRDSCustomInstanceRole-$REGION \
  --assume-role-policy-document '{
    "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Principal": {
              "Service": "ec2.amazonaws.com"
          }
        }
      ]
    }'
```

ステップ 2: AWSRDSCustomInstanceRoleForRdsCustomInstance にアクセスポリシーを追加します。

IAM ロールにインラインポリシーを埋め込むと、そのインラインポリシーはロールのアクセス (アクセス権) ポリシーの一部として使用されます。Amazon EC2 にメッセージの送受信やさまざまなアクションを許可するAWSRDSCustomIamRolePolicyポリシーを作成します。

次の例では、AWSRDSCustomIamRolePolicyという名前のアクセスポリシーを作成し、IAM ロールAWSRDSCustomInstanceRole-*region*に追加します。この例では、次の環境変数を設定していることを前提としています。

$REGION

この変数を DB インスタンスを作成する予定の AWS リージョン に設定します。

$ACCOUNT_ID

この変数を自分の AWS アカウント 番号に設定します。

$KMS_KEY

この変数を、RDS Custom DB インスタンスに使用する AWS KMS key の Amazon リソースネーム (ARN) に設定します。複数の KMS キーを指定するには、ステートメントID (Sid) 11 のResourcesセクションに追加してください。

```
aws iam put-role-policy \
  --role-name AWSRDSCustomInstanceRole-$REGION \
```

```
  --policy-name AWSRDSCustomIamRolePolicy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:GetDeployablePatchSnapshotForInstance",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:GetManifest",
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:ListAssociations",
                "ssm:ListInstanceAssociations",
                "ssm:PutInventory",
                "ssm:PutComplianceItems",
                "ssm:PutConfigurePackageResult",
                "ssm:UpdateAssociationStatus",
                "ssm:UpdateInstanceAssociationStatus",
                "ssm:UpdateInstanceInformation",
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceInformation",
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "2",
            "Effect": "Allow",
            "Action": [
                "ec2messages:AcknowledgeMessage",
                "ec2messages:DeleteMessage",
                "ec2messages:FailMessage",
                "ec2messages:GetEndpoint",
                "ec2messages:GetMessages",
                "ec2messages:SendReply"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "3",
            "Effect": "Allow",
            "Action": [
                "logs:PutRetentionPolicy",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:'$REGION':'$ACCOUNT_ID':log-group:rds-custom-instance*"
            ]
        },
        {
            "Sid": "4",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::do-not-delete-rds-custom-*/*"
            ]
        },
        {
            "Sid": "5",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
```

```
                        "RDSCustomForOracle/Agent"
                    ]
                }
            }
        },
        {
            "Sid": "6",
            "Effect": "Allow",
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "7",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:'$REGION':'$ACCOUNT_ID':secret:do-not-delete-
rds-custom-*",
                "arn:aws:secretsmanager:'$REGION':'$ACCOUNT_ID':secret:rds-custom!
oracle-do-not-delete-*"
            ]
        },
        {
           "Sid": "8",
           "Effect": "Allow",
           "Action": [
             "s3:ListBucketVersions"
           ],
           "Resource": [
             "arn:aws:s3:::do-not-delete-rds-custom-*"
           ]
         },
         {
            "Sid": "9",
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshots",
            "Resource": [
```

```
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            }
        },
        {
            "Sid": "10",
            "Effect": "Allow",
            "Action": "ec2:CreateSnapshots",
            "Resource": [
                "arn:aws:ec2:*::snapshot/*"
            ]
        },
        {
            "Sid": "11",
            "Effect": "Allow",
            "Action": [
              "kms:Decrypt",
              "kms:GenerateDataKey"
            ],
            "Resource": [
              "arn:aws:kms:'$REGION':'$ACCOUNT_ID':key/'$KMS_KEY'"
            ]
        },
        {
            "Sid": "12",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:CreateAction": [
                        "CreateSnapshots"
                    ]
                }
            }
        },
        {
            "Sid": "13",
            "Effect": "Allow",
```

```
            "Action": [
                "sqs:SendMessage",
                "sqs:ReceiveMessage",
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl"
            ],
            "Resource": "arn:aws:sqs:'$REGION':'$ACCOUNT_Id':do-not-delete-rds-custom-
*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            }
        }
    ]
}'
```

ステップ 3: RDS Custom インスタンスプロファイル AWSRDSCustomInstanceProfile を作成する

インスタンスプロファイルは、1 つの IAM ロールを含むコンテナです。RDS Custom はインスタンスプロファイルを使用してインスタンスにロールを渡します。

CLI を使用してロールを作成する場合、ロールとインスタンスプロファイルを別個のアクションとして作成します。名前は異なる可能性があります。IAM インスタンスプロファイルを次のように作成し、AWSRDSCustomInstanceProfile-*region* という形式を使用して名前を付けます。次の例では、DB インスタンスを作成する環境変数 $REGION を AWS リージョン に設定していることを前提としています。

```
aws iam create-instance-profile \
    --instance-profile-name AWSRDSCustomInstanceProfile-$REGION
```

ステップ 4: AWSRDSCustomInstanceRoleForRdsCustomInstance を AWSRDSCustomInstanceProfile に追加する

以前に作成したインスタンスプロファイルに IAM ロールを追加します。次の例では、DB インスタンスを作成する環境変数 $REGION を AWS リージョン に設定していることを前提としています。

```
aws iam add-role-to-instance-profile \
    --instance-profile-name AWSRDSCustomInstanceProfile-$REGION \
    --role-name AWSRDSCustomInstanceRole-$REGION
```

ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する

CEV または RDS Custom DB インスタンスを作成する IAM プリンシパル (ユーザーまたはロール) に、次のいずれかのポリシーがあることを確認してください。

- `AdministratorAccess` ポリシー
- Amazon S3 と AWS KMS、CEV の作成、および DB インスタンスの作成に必要なアクセス許可を含む `AmazonRDSFullAccess` ポリシー。

トピック

- Amazon S3 と AWS KMS に必要な IAM アクセス許可
- CEV を作成するために必要な IAM アクセス許可
- CEV から DB インスタンスを作成するために必要な IAM アクセス許可

Amazon S3 と AWS KMS に必要な IAM アクセス許可

CEV または RDS Custom for Oracle DB インスタンスを作成するには、IAM プリンシパルで Amazon S3 および AWS KMS にアクセスする必要があります。次のサンプル JSON ポリシーにより、必要なアクセス許可が付与されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:PutBucketObjectLockConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
```

```
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

kms:CreateGrant アクセス許可の詳細については、「AWS KMS key 管理」を参照してください。

CEV を作成するために必要な IAM アクセス許可

CEV を作成する場合、IAM プリンシパルで次の追加のアクセス許可が必要です。

```
s3:GetObjectAcl
s3:GetObject
s3:GetObjectTagging
s3:ListBucket
mediaimport:CreateDatabaseBinarySnapshot
```

次のサンプル JSON ポリシーにより、バケット *my-custom-installation-files* とその内容へのアクセスに必要な追加のアクセス許可が付与されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessToS3MediaBucket",
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectAcl",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-custom-installation-files",
```

```
                "arn:aws:s3:::my-custom-installation-files/*"
            ]
        },
        {
            "Sid": "PermissionForByom",
            "Effect": "Allow",
            "Action": [
                "mediaimport:CreateDatabaseBinarySnapshot"
            ],
            "Resource": "*"
        }
    ]
}
```

S3 バケットポリシーを使用して、Amazon S3 への同様のアクセス許可を発信者のアカウントに付与できます。

CEV から DB インスタンスを作成するために必要な IAM アクセス許可

既存の CEV から RDS Custom for Oracle DB インスタンスを作成する場合、IAM プリンシパルで次の追加のアクセス許可が必要です。

```
iam:SimulatePrincipalPolicy
cloudtrail:CreateTrail
cloudtrail:StartLogging
```

次のサンプル JSON ポリシーは、IAM ロールの検証や情報を AWS CloudTrail に記録するために必要なアクセス許可を付与します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ValidateIamRole",
            "Effect": "Allow",
            "Action": "iam:SimulatePrincipalPolicy",
            "Resource": "*"
        },
        {
```

```
            "Sid": "CreateCloudTrail",
            "Effect": "Allow",
            "Action": [
                "cloudtrail:CreateTrail",
                "cloudtrail:StartLogging"
            ],
            "Resource": "arn:aws:cloudtrail:*:*:trail/do-not-delete-rds-custom-*"
        }
    ]
}
```

ステップ 6: RDS Custom for Oracle 用に VPC を設定する

RDS Custom DB インスタンスは、Amazon EC2 インスタンスまたは Amazon RDS インスタンスと同様に、Amazon VPC サービスに基づく Virtual Private Cloud (VPC) にあります。独自の VPC を提供して設定します。RDS Custom for SQL Server とは異なり、RDS Custom for Oracle はアクセスコントロールリストやセキュリティグループを作成しません。独自のセキュリティグループ、サブネット、ルートテーブルをアタッチする必要があります。

仮想プライベートクラウド (VPC) は、CloudFormation または手動プロセスを使用して設定できます。

> **Important**
>
> CloudFormation を使用して RDS Custom for Oracle 環境を設定することを強くお勧めします。この方法は、最も簡単でエラーが起こりにくいものです。

トピック

- CloudFormation を使用して VPC を設定する (推奨)
- RDS Custom for Oracle 用に VPC を手動で設定する

CloudFormation を使用して VPC を設定する (推奨)

VPC を異なる RDS Custom エンジン用に既に設定し、既存の VPC を再利用する場合は、このステップをスキップします。このセクションでは、以下を想定しています。

- 既に CloudFormation を使用して IAM インスタンスプロファイルとロールを作成しました。

- ルートテーブル ID はわかっています。

  DB インスタンスがプライベートであるためには、プライベートサブネットに存在する必要があります。サブネットをプライベートにするには、デフォルトのインターネットゲートウェイを持つルートテーブルに関連付けてはいけません。詳細については、Amazon VPC ユーザーガイドの「ルートテーブルを設定する」を参照してください。

VPC 用の CloudFormation テンプレートを使用すると、以下のリソースが作成されます。

- プライベート VPC
- `rds-custom-private` という名前のサブネットグループ
- DB インスタンスが依存 AWS のサービス と通信するために使用する、以下の VPC エンドポイント。
  - `com.amazonaws.region.ec2messages`
  - `com.amazonaws.region.events`
  - `com.amazonaws.region.logs`
  - `com.amazonaws.region.monitoring`
  - `com.amazonaws.region.s3`
  - `com.amazonaws.region.secretsmanager`
  - `com.amazonaws.region.ssm`
  - `com.amazonaws.region.ssmmessages`

  マルチ AZ 配置を作成する場合。

  - `com.amazonaws.region.sqs`

> **Note**
>
> 既存のアカウントで複雑なネットワーク設定を行う場合、アクセスがまだ存在しない場合は、依存サービスへのアクセスを手動で設定することをお勧めします。詳細については、「VPC が依存 AWS のサービスにアクセスできることを確認する」を参照してください。

CloudFormation を使用して VPC を設定するには

1. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。

2. [スタックの作成] ウィザードを起動し、[スタックの作成] を選択し、[新しいリソースを使用 (標準)] を選択します。
3. [Create stack] (スタックの作成) ページで、次の手順を実行します。
   a. [Prepare template] (テンプレートの準備) の [Template is ready] (テンプレートの準備が完了しました) を選択します。
   b. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。
   c. ファイルを選択で、`custom-vpc.json`に移動して選択します。
   d. [次へ] を選択します。
4. [スタックの詳細を指定] ページで、以下を実行します。
   a. [スタック名] に「**`custom-vpc`**」と入力します。
   b. パラメータで、RDS Custom DB インスタンスに使用するプライベートサブネットを選択します。
   c. RDS Custom DB インスタンスに使用するプライベート VPC ID を選択します。
   d. プライベートサブネットに関連付けられているメインルートテーブルを入力します。
   e. [次へ] を選択します。
5. [スタックオプションの設定] ページで、[次へ] を選択します。
6. [確認] ページで [送信] を選択します。

   CloudFormation は、プライベート VPC を設定します。左側のパネルで、[custom-vpc] に [CREATE_COMPLETE] と表示されたら、次のステップに進みます。
7. (オプション) VPC の詳細を確認します。[スタック] ペインで、[custom-vpc] を選択します。右のペインで、以下の操作を行います。
   a. [スタック情報] を選択します。スタックの ID は、arn:aws:cloudformation:***`region`***:***`account-no`***:stack/custom-vpc/***`identifier`*** という形式です。
   b. [リソース] をクリックします。rds-custom-private という名前のサブネットグループと、vpce-***`string`***という命名形式を使用する VPC エンドポイントがいくつか表示されるはずです。各エンドポイントは、RDS Custom が通信する必要のある AWS のサービス に対応しています。詳細については、「VPC が依存 AWS のサービスにアクセスできることを確認する」を参照してください。

c. [Parameters] (パラメータ) を選択します。スタックを作成したときに指定したプライベートサブネット、プライベート VPC、ルートテーブルが表示されます。DB インスタンスを作成するときは、VPC ID とサブネットグループを指定する必要があります。

RDS Custom for Oracle 用に VPC を手動で設定する

CloudFormation で VPC の作成を自動化する代わりに、VPC を手動で設定できます。このオプションは、既存のリソースを使用する複雑なネットワーク設定がある場合に最適です。

トピック

- VPC が依存 AWS のサービスにアクセスできることを確認する
- インスタンスメタデータサービスの設定

VPC が依存 AWS のサービスにアクセスできることを確認する

RDS Custom は DB インスタンスから他の AWS のサービス に通信を送信します。RDS Custom DB インスタンスを作成するサブネットから以下のサービスにアクセスできることを確認します。

- Amazon CloudWatch (`com.amazonaws.`*`region`*`.monitoring`)
- Amazon CloudWatch Logs (`com.amazonaws.`*`region`*`.logs`)
- Amazon CloudWatch Events (`com.amazonaws.`*`region`*`.events`)
- Amazon EC2 (`com.amazonaws.`*`region`*`.ec2` および `com.amazonaws.`*`region`*`.ec2messages`)
- Amazon S3 (`com.amazonaws.`*`region`*`.s3`)
- AWS Secrets Manager (`com.amazonaws.`*`region`*`.secretsmanager`)
- AWS Systems Manager (`com.amazonaws.`*`region`*`.ssm` および `com.amazonaws.`*`region`*`.ssmmessages`)

マルチ AZ 配置を作成する場合

- Amazon Simple Queue Service (`com.amazonaws.`*`region`*`.sqs`)

RDS Custom は必要なサービスと通信できない場合、次のイベントを公開します。

```
Database instance in incompatible-network. SSM Agent connection not available. Amazon
 RDS can't connect to the dependent AWS services.
```

```
Database instance in incompatible-network. Amazon RDS can't connect to dependent AWS
 services. Make sure port 443 (HTTPS) allows outbound connections, and try again.
 "Failed to connect to the following services: s3 events"
```

incompatible-network エラーを避けるため、RDS Custom DB インスタンスと AWS のサービス間の通信に関与している VPC コンポーネントが次の要件を満たしていることを確認します。

- DB インスタンスは、ポート 443 で他の AWS のサービス へのアウトバウンド接続を行うことができます。
- VPC は、RDS Custom DB インスタンスから発信されるリクエストへの受信レスポンスを許可します。
- RDS Custom は、各 AWS のサービス に対してエンドポイントのドメイン名を正しく解決できます。

異なる RDS Custom DB エンジン用に VPC を既に設定している場合は、その VPC を再利用してこのプロセスをスキップできます。

インスタンスメタデータサービスの設定

以下を実行して、インスタンスが に接続できることを確認します。

- インスタンスメタデータにアクセスするために、Instance Metadata Service のバージョン 2 (IMDSv2) を使用します。
- ポート 80 (HTTP) を介して IMDS リンク IP アドレスへのアウトバウンド通信を許可します。
- http://169.254.169.254からのインスタンスメタデータのリクエスト、IMDSv2 のリンクです。

詳細については、「Amazon EC2 ユーザーガイド」の「IMDSv2 の使用」を参照してください。

RDS Custom for Oracle オートメーションでは、基盤となる Amazon EC2 インスタンスにHttpTokens=enabledを設定し、デフォルトで IMDSv2 を使用しています。ただし、必要に応じて IMDSv1 を使用することもできます。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータオプションを設定する」を参照してください。

Amazon RDS Custom for Oracle のカスタムエンジンバージョンでの作業

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS Custom for Oracle のカスタムエンジンバージョン (CEV) は、データベースエンジンと特定の Amazon マシンイメージ (AMI) のバイナリボリュームスナップショットです。RDS Custom for Oracle はデフォルトで入手可能な最新の AMI を使用しますが、以前の CEV で使用していた AMI を指定できます。データベースのインストールファイルを Amazon S3 に保存します。RDS Custom は、インストールファイルと AMI を使用して CEV を作成します。

トピック

- CEV 作成の準備
- CEV の作成
- CEV ステータスの変更
- Amazon RDS Custom for Oracle の CEV 詳細を表示する
- CEV の削除

CEV 作成の準備

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

CEV を作成するには、次のいずれかのリリースの Amazon S3 バケットに保存されているインストールファイルおよびパッチにアクセスします。

- Oracle Database 19c
- Oracle Database 18c
- Oracle Database 12c Release 2 (12.2)
- Oracle Database 12c Release 1 (12.1)

例えば、Oracle Database 19c の 2021 年 4 月 RU/RUR、またはインストールファイルとパッチの有効な組み合わせを使用できます。RDS Custom for Oracle でサポートされているバージョンとリージョンの詳細については、「RDS Custom with RDS for Oracle」を参照してください。

トピック

- ステップ 1 (オプション): マニフェストテンプレートをダウンロードする
- ステップ 2: Oracle Software Delivery Cloud からデータベースインストールファイルおよびパッチをダウンロードする
- ステップ 3: Amazon S3 へのインストールファイルをアップロードする
- ステップ 4 (オプション): S3 にあるインストールメディアを AWS アカウント 間で共有する
- ステップ 5: CEV マニフェストを準備する
- ステップ 6 (オプション): CEV マニフェストを検証する
- ステップ 7: 必要な IAM アクセス許可を追加する

ステップ 1 (オプション): マニフェストテンプレートをダウンロードする

CEV マニフェストは、CEV のデータベースインストール .zip ファイルのリストを含む JSON ドキュメントです。CEV を作成するには、以下の手順を実行します。

1. CEV に含める Oracle データベースのインストールファイルを特定します。
2. インストールファイルをダウンロードします。
3. インストールファイルを一覧表示する JSON マニフェストを作成します。

RDS Custom for Oracle には、サポートされている Oracle データベースの各リリース用に、推奨する.zip ファイルが含まれた JSON マニフェストテンプレートが用意されています。例えば、次のテンプレートは 19.17.0.0.0 RU 用です。

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
```

```
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p34419443_190000_Linux-x86-64.zip",
        "p34411846_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
        "p31335037_190000_Linux-x86-64.zip",
        "p32327201_190000_Linux-x86-64.zip",
        "p33613829_190000_Linux-x86-64.zip",
        "p34006614_190000_Linux-x86-64.zip",
        "p34533061_190000_Linux-x86-64.zip",
        "p34533150_190000_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29213893_1917000DBRU_Generic.zip",
        "p33125873_1917000DBRU_Linux-x86-64.zip",
        "p34446152_1917000DBRU_Linux-x86-64.zip"
    ]
}
```

各テンプレートには、パッチのダウンロード手順、.zip ファイルの URL、およびファイルチェックサムが記載された Readme が関連付けられています。これらのテンプレートはそのまま使用することも、独自のパッチで修正することもできます。テンプレートを確認するには、custom-oracle-manifest.zip をローカルディスクにダウンロードし、ファイルアーカイブアプリケーションで開きます。詳細については、「ステップ 5: CEV マニフェストを準備する」を参照してください。

ステップ 2: Oracle Software Delivery Cloud からデータベースインストールファイルおよびパッチをダウンロードする

CEV に必要なインストールファイルを特定したら、ローカルシステムにダウンロードします。Oracle Database のインストールファイルおよびパッチは、Oracle Software Delivery Cloudでホストされています。各 CEV には Oracle Database 19c や Oracle Database 12c Release 2 (12.2) などのベースのリリースが必要です。

Oracle Database のデータベースインストールファイルをダウンロードするには

1. https://edelivery.oracle.com/に移動してサインインします。

2. 検索ボックスに「**Oracle Database Enterprise Edition**」または「**Oracle Database Standard Edition 2**」と入力し、[検索] を選択します。
3. 次のベースのリリースのいずれかを選択します。

データベースのバージョン	Enterprise Edition	Standard Edition 2
Oracle Database 19c	DLP: Oracle Database 19c Enterprise Edition 19.3.0.0.0 (Oracle Database Enterprise Edition)	DLP: Oracle Database 19c Standard Edition 2 19.3.0.0.0 (Oracle Database Standard Edition 2)
Oracle Database 18c	DLP: Oracle Database 18c Enterprise Edition 18.0.0.0.0 (Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 18.0.0.0.0 (Oracle Database Standard Edition 2)
Oracle Database 12c Release 2 (12.2.0.1)	DLP: Oracle Database 12c Enterprise Edition 12.2.0.1.0 (Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 12.2.0.1.0 (Oracle Database Standard Edition 2)
Oracle Database 12c Release 1 (12.1.0.2)	DLP: Oracle Database 12c Enterprise Edition 12.1.0.2.0 (Oracle Database Enterprise Edition)	DLP: Oracle Database Standard Edition 2 12.1.0.2.0 (Oracle Database Standard Edition 2)

4. [続行] をクリックしてください。
5. [Download Queue] (キューのダウンロード) チェックボックスをオフにします。
6. ベースのリリースに対応するオプションを選択します。
   - Oracle Database 19.3.0.0.0 - 長期リリース。
   - Oracle Database 18.0.0.0.0
   - Oracle Database 12.2.0.1.0.
   - Oracle Database 12.1.0.2.0.
7. プラットフォーム/言語でLinux x86-64を選択します。

8. [続行] を選択し、Oracle ライセンス契約に署名します。
9. データベースリリースに対応する.zip ファイルを選択します。

データベースのリリースとエディション	Zip ファイル	SHA-256 ハッシュ
19c EE と SE2	V982063-01.zip	BA8329C757133DA313ED3B6D7F86C5AC42CD9970A28BF2E6233F3235233AA8D8
18c EE と SE2	V978967-01.zip	C96A4FD768787AF98272008833FE10B172691CF84E42816B138C12D4DE63AB96
12.2.0.1 EE と SE2	V839960-01.zip	96ED97D21F15C1AC0CCE3749DA6C3DAC7059BB60672D76B008103FC754D22DDE
12.1.0.2 EE	V46095-01_1of2.zip V46095-01_2of2.zip	31FDC2AF41687B4E547A3A18F796424D8C1AF36406D2160F65B0AF6A9CD47355 for V46095-01_1of2.zip 03DA14F5E875304B28F0F3BB02AF0EC33227885B99C9865DF70749D1E220ACCD for V46095-01_2of2.zip
12.1.0.2 SE2	V77388-01_1of2.zip V77388-01_2of2.zip	73873369753230F5A0921F95ACEADB591388CB06ED72A7F3AEA7BCBCEA2403BC for V77388-01_1of2.zip 2492E1BE1E3E3531DA83D0843C09C08E435AC8CEFD9A00C0DF56BE4F15CEEBF3 for V77388-01_2of2.zip

10. 必要な Oracle パッチを updates.oracle.com または support.oracle.com からローカルシステムにダウンロードします。パッチの URL は次の場所にあります。
    - ステップ 1 (オプション): マニフェストテンプレートをダウンロードする でダウンロードした .zip ファイル内の Readme ファイル

- Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle の 各 Release Update (RU) に一覧表示されているパッチ

ステップ 3: Amazon S3 へのインストールファイルをアップロードする

AWS CLIを使用して、Oracle インストールファイルとパッチファイルを Amazon S3 にアップロードします。インストールファイルを含む S3 バケットはCEV と同じAWSリージョンにある必要があります。

このセクションの例では、次のプレースホルダを使用します。

- *install-or-patch-file.zip* - Oracle インストールメディアファイルです。例えば、p32126828_190000_Linux-x86-64.zip はパッチです。
- *amzn-s3-demo-destination-bucket* - アップロードされたインストールファイル用に設計された Amazon S3 バケットです。
- *123456789012/cev1* – Amazon S3 バケット内のオプションのプレフィックスです。
- *amzn-s3-demo-source-bucket* – オプションでファイルをステージングできる Amazon S3 バケットです。

トピック

- ステップ 3a: S3 バケットが正しい AWS リージョン にあることを確認する
- ステップ 3b: S3 バケットポリシーに正しい権限があることを確認する
- ステップ 3c: cp コマンドまたは sync コマンドを使用してファイルをアップロードする
- ステップ 3 d: S3 バケットにあるファイルをリストする

ステップ 3a: S3 バケットが正しい AWS リージョン にあることを確認する

S3 バケットが、`create-custom-db-engine-version`コマンドを実行する予定のAWSリージョンにあることを確認します。

```
aws s3api get-bucket-location --bucket amzn-s3-demo-destination-bucket
```

ステップ 3b: S3 バケットポリシーに正しい権限があることを確認する

CEV はゼロから作成することも、ソース CEV から作成することもできます。ソース CEV から新しい CEV を作成する場合は、S3 バケットポリシーに正しい権限があることを確認してください。

1. RDS Custom によって予約されている S3 バケットを特定します。バケットの形式は do-not-delete-rds-custom-*account*-*region*-*string* になります。例えば、バケットには do-not-delete-rds-custom-*123456789012*-*us-east-1*-*abc123EXAMPLE* という名前が付けられる可能性があります。
2. S3 バケットポリシーに次の権限が追加されていることを確認してください。do-not-delete-rds-custom-*123456789012*-*us-east-1*-*abc123EXAMPLE* をバケットの名前に置き換えます。

```
{
    "Sid": "AWSRDSCustomForOracleCustomEngineVersionGetObject",
    "Effect": "Allow",
    "Principal": {
        "Service": "custom.rds.amazonaws.com"
    },
    "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging"
    ],
    "Resource": "arn:aws:s3:::do-not-delete-rds-custom-123456789012-us-
east-1-abc123EXAMPLE/CustomEngineVersions/*"
}, ...
```

ステップ 3c: cp コマンドまたは sync コマンドを使用してファイルをアップロードする

次のいずれかのオプションを選択します。

- aws s3 cpを使用して、単一の.zip ファイルをアップロードします。

  各インストール.zip ファイルを個別にアップロードします。.zip ファイルを単一の.zip ファイルに結合しないでください。
- aws s3 syncを使用して、ディレクトリをアップロードします。

Example

次の例では、*install-or-patch-file.zip*をRDS Custom Amazon S3 バケット内のフォルダ*123456789012/cev1*にアップロードします。アップロードする .zip ごとにaws s3コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws s3 cp install-or-patch-file.zip \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows の場合:

```
aws s3 cp install-or-patch-file.zip ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Example

以下の例では、ローカルの*cev1*フォルダのファイルを、Amazon S3 バケットの*123456789012/cev1*フォルダにアップロードします。

Linux、macOS、Unix の場合:

```
aws s3 sync cev1 \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows の場合:

```
aws s3 sync cev1 ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Example

次の例では、*amzn-s3-demo-source-bucket* のすべてのファイルを Amazon S3 バケット内の *123456789012/cev1* フォルダにアップロードしています。

Linux、macOS、Unix の場合:

```
aws s3 sync s3://amzn-s3-demo-source-bucket/ \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

Windows の場合:

```
aws s3 sync s3://amzn-s3-demo-source-bucket/ ^
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

ステップ 3 d: S3 バケットにあるファイルをリストする

次の例では、`s3 ls`コマンドを使用して、RDS Custom Amazon S3 バケットにあるファイルを指定します。

```
aws s3 ls \
    s3://amzn-s3-demo-destination-bucket/123456789012/cev1/
```

ステップ 4 (オプション): S3 にあるインストールメディアを AWS アカウント 間で共有する

このセクションでは、アップロードした Oracle インストールファイルを含む Amazon S3 バケットをメディアバケットとします。組織では、AWS リージョン で複数の AWS アカウント を使用する場合があります。その場合、メディアバケットに埋め込むために1つの AWS アカウント を使用し、CEV を作成するために別の AWS アカウント を使用する場合もあります。メディアバケットを共有する予定がない場合は、次のセクションにスキップしてください。

このセクションでは、以下を想定しています。

- メディアバケットを作成したアカウントと、CEV を作成する予定の別のアカウントにアクセスできます。
- CEV を 1 つの AWS リージョン だけで作成します。複数のリージョンを使用する場合は、リージョンごとにメディアバケットを作成してください。
- CLI を使用しています。Amazon S3 コンソールを使用している場合は、次のステップを適用します。

メディアバケットを AWS アカウント 全体で共有するように設定するには

1. インストールメディアをアップロードした S3 バケットを含む AWS アカウント にログインします。
2. 空白の JSON ポリシーテンプレートまたは適用可能な既存のポリシーから開始します。

   次のコマンドは、既存のポリシーを取得して *my-policy.json* として保存します。この例では、インストールファイルを含む S3 バケットに *amzn-s3-demo-bucket* という名前を付けています。

   ```
   aws s3api get-bucket-policy \
       --bucket amzn-s3-demo-bucket \
       --query Policy \
   ```

```
    --output text > my-policy.json
```

3. メディアバケットの権限を次のように編集します。

   - テンプレートの `Resource` で、Oracle Database インストールファイルをアップロードした S3 バケットを指定します。
   - `Principal` 要素で、CEV を作成するために使用する予定のすべての AWS アカウント の ARN を指定します。ルート、ユーザー、またはロールを S3 バケット許可リストに追加できます。詳細については、AWS Identity and Access Management ユーザーガイドの「IAM ID」を参照してください。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "GrantAccountsAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:user/user-name-with-path",
                    "arn:aws:iam::123456789012:role/role-name-with-path"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

4. ポリシーをメディアバケットにアタッチします。

   次の例では、*amzn-s3-demo-bucket* は、インストールファイルを含む S3 バケットの名前であり、*my-policy.json* は JSON ファイルの名前です。

   ```
   aws s3api put-bucket-policy \
       --bucket amzn-s3-demo-bucket \
       --policy file://my-policy.json
   ```

5. CEV を作成する AWS アカウント にログインします。
6. このアカウントが、メディアバケットを作成した AWS アカウント でアクセスできることを確認します。

   ```
   aws s3 ls --query "Buckets[].Name"
   ```

   詳細については、AWS CLI コマンドリファレンスの「aws s3 ls」を参照してください。
7. CEV の作成 のステップに従って、CEV を作成します。

ステップ 5: CEV マニフェストを準備する

CEV マニフェストは、以下を含む JSON ドキュメントです。

- (必須) Amazon S3 にアップロードされたインストールの .zip ファイルのリスト。RDS Custom は、マニフェストにリストされている順にパッチを適用します。
- (オプション) Oracle ベース、Oracle ホーム、および UNIX/Linux ユーザーとグループの ID と名前のデフォルト以外の値を設定するインストールパラメータ。既存の CEV または既存の DB インスタンスのインストールパラメータは変更できないことに注意してください。また、インストールパラメータの設定が異なる場合、ある CEV から別の CEV にアップグレードすることはできません。

CEV マニフェストのサンプルについては、ステップ 1 (オプション): マニフェストテンプレートをダウンロードする でダウンロードした JSON テンプレートを参照してください。CEV マニフェストの例 でサンプルを確認することもできます。

トピック

- CEV マニフェストの JSON フィールド
- CEV マニフェストの作成

- CEV マニフェストの例

CEV マニフェストの JSON フィールド

次の表では、マニフェストの JSON フィールドについて説明します。

JSON フィールド	説明
`MediaImportTemplateVersion`	CEV マニフェストのバージョンです。日付は `YYYY-MM-DD` 形式です。
`databaseInstallationFileNames`	データベースのインストールファイルの順序一覧です。
`opatchFileNames`	Oracle DB エンジンで使用される OPatch インストーラの順序一覧です。有効な値は 1 つだけです。`opatchFileNames` の値は `p6880880_` で始まる必要があります。
`psuRuPatchFileNames`	このデータベースの PSU および RU パッチです。 **Important** `psuRuPatchFileNames` が含まれている場合は、`opatchFileNames` が必要です。`opatchFileNames` の値は `p6880880_` で始まる必要があります。
`OtherPatchFileNames`	PSU および RU パッチのリストにないパッチです。RDS Custom は、PSU および RU パッチ適用後に、これらのパッチを適用します。 **Important** `OtherPatchFileNames` が含まれている場合は、`opatchFileNames` が必要です。`opatchFileNames` の値は `p6880880_` で始まる必要があります。

JSON フィールド	説明
`installationParameters`	Oracle ベース、Oracle ホーム、および UNIX/Linux ユーザーとグループの ID と名前のデフォルト以外の設定。以下のパラメータを設定できます。 oracleBase Oracle バイナリがインストールされているディレクトリ。ファイルを保存するバイナリボリュームのマウントポイントです。Oracle ベースディレクトリには、複数の Oracle ホームを含めることができます。例えば、`/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1` が Oracle ホームディレクトリの 1 つであれば、`/home/oracle` は Oracle ベースディレクトリです。ユーザー指定の Oracle ベースディレクトリはシンボリックリンクではありません。 Oracle ベースを指定しない場合、デフォルトディレクトリは `/rdsdbbin` です。 oracleHome Oracle データベースバイナリがインストールされているディレクトリ。例えば、`/home/oracle/` を Oracle ベースとして指定した場合、`/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1/` Oracle ホームとして指定できます。ユーザー指定の Oracle ホームディレクトリはシンボリックリンクではありません。Oracle ホーム値は `$ORACLE_HOME` 環境変数によって参照されます。 Oracle ホームを指定しない場合、デフォルトの命名形式は `/rdsdbbin/oracle.` *major-engine-version* `.custom.r1.` *engine-edition* `.1` です。 unixUname Oracle ソフトウェアを所有する UNIX ユーザーの名前。RDS Custom は、ローカルデータベースコマンドを実行するときにこのユーザーを引き継ぎます。`unixUid` と `unixUname` の

JSON フィールド	説明
	両方を指定すると、RDS Custom では、ユーザーが存在しない場合はユーザーを作成し、最初の UID と異なる場合はユーザーに UID を割り当てます。 デフォルトのユーザー名は `rdsdb` です。 unixUid Oracle ソフトウェアを所有する UNIX ユーザーの ID (UID)。`unixUid` と `unixUname` の両方を指定すると、RDS Custom では、ユーザーが存在しない場合はユーザーを作成し、最初の UID と異なる場合はユーザーに UID を割り当てます。 デフォルトの UID は `61001` です。これは、ユーザー `rdsdb` の UID です。 unixGroupName UNIX グループの名前。Oracle ソフトウェアを所有する UNIX ユーザーは、このグループに属します。 デフォルトのグループ名は `rdsdb` です。 unixGroupId UNIX ユーザーが属する UNIX グループの ID。 デフォルトのグループ ID は `1000` です。これはグループ `rdsdb` の ID です。

Oracle データベースのリリースごとに、サポートされるインストールファイルのリストは異なります。CEV マニフェストを作成するときは、RDS Custom for Oracle でサポートされているファイルのみを指定してください。そうしないと、CEV の作成はエラーで失敗します。Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle で一覧表示されているすべてのパッチがサポートされています。

CEV マニフェストの作成

CEV マニフェストを作成するには

1. 適用する予定のすべてのインストールファイルを、適用する順序で一覧表示します。
2. インストールファイルを、CEV マニフェストの JSON フィールド で説明されている JSON フィールドと関連付けます。
3. 次のいずれかを実行します。
   - CEV マニフェストを JSON テキストファイルとして作成します。
   - コンソールで CEV を作成するときに、CEV マニフェストテンプレートを編集します。(詳しくは、「CEV の作成」を参照してください。)

CEV マニフェストの例

次の例は、さまざまな Oracle Database リリースの CEV マニフェストファイルを示しています。マニフェストに JSON フィールドを含める場合は、それが空でないことを確認してください。例えば、次の CEV マニフェストは `otherPatchFileNames` が空のため有効ではありません。

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p32126828_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
    ]
}
```

トピック

- Sample CEV manifest for Oracle Database 12c Release 1 (12.1)
- Sample CEV manifest for Oracle Database 12c Release 2 (12.2)
- Sample CEV manifest for Oracle Database 18c

- Sample CEV manifest for Oracle Database 19c

Example Oracle Database 12c Release 1 (12.1) のサンプル CEV マニフェスト

次の Oracle Database 12c Release 1 (12.1) の 2021 年 7 月 PSU の例では、RDS Custom は、指定された順序でパッチを適用します。したがって、RDS Custom は、p32768233、次に p32876425、次に p18759211 を適用することになります。この例では、UNIX ユーザーとグループ、Oracle ホームと Oracle ベースに新しい値を設定します。

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V46095-01_1of2.zip",
        "V46095-01_2of2.zip"
    ],
    "opatchFileNames":[
        "p6880880_121010_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p32768233_121020_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p32876425_121020_Linux-x86-64.zip",
        "p18759211_121020_Linux-x86-64.zip",
        "p19396455_121020_Linux-x86-64.zip",
        "p20875898_121020_Linux-x86-64.zip",
        "p22037014_121020_Linux-x86-64.zip",
        "p22873635_121020_Linux-x86-64.zip",
        "p23614158_121020_Linux-x86-64.zip",
        "p24701840_121020_Linux-x86-64.zip",
        "p25881255_121020_Linux-x86-64.zip",
        "p27015449_121020_Linux-x86-64.zip",
        "p28125601_121020_Linux-x86-64.zip",
        "p28852325_121020_Linux-x86-64.zip",
        "p29997937_121020_Linux-x86-64.zip",
        "p31335037_121020_Linux-x86-64.zip",
        "p32327201_121020_Linux-x86-64.zip",
        "p32327208_121020_Generic.zip",
        "p17969866_12102210119_Linux-x86-64.zip",
        "p20394750_12102210119_Linux-x86-64.zip",
        "p24835919_121020_Linux-x86-64.zip",
        "p23262847_12102201020_Linux-x86-64.zip",
```

```
        "p21171382_12102201020_Generic.zip",
        "p21091901_12102210720_Linux-x86-64.zip",
        "p33013352_12102210720_Linux-x86-64.zip",
        "p25031502_12102210720_Linux-x86-64.zip",
        "p23711335_12102191015_Generic.zip",
        "p19504946_121020_Linux-x86-64.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.12.1.0.2",
        "oracleBase": "/home/oracle"
    }
}
```

Example Oracle Database 12c Release 2 (12.2) のサンプル CEV マニフェスト

次の Oracle Database 12c Release 1 (12.2) の 2021 年 10 月 PSU の例では、RDS Custom が p33261817、p33192662、p29213893 などを順に適用しています。この例では、UNIX ユーザーとグループ、Oracle ホームと Oracle ベースに新しい値を設定します。

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V839960-01.zip"
    ],
    "opatchFileNames":[
        "p6880880_122010_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p33261817_122010_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p33192662_122010_Linux-x86-64.zip",
        "p29213893_122010_Generic.zip",
        "p28730253_122010_Linux-x86-64.zip",
        "p26352615_12201211019DBOCT2021RU_Linux-x86-64.zip",
        "p23614158_122010_Linux-x86-64.zip",
        "p24701840_122010_Linux-x86-64.zip",
        "p25173124_122010_Linux-x86-64.zip",
        "p25881255_122010_Linux-x86-64.zip",
```

```
        "p27015449_122010_Linux-x86-64.zip",
        "p28125601_122010_Linux-x86-64.zip",
        "p28852325_122010_Linux-x86-64.zip",
        "p29997937_122010_Linux-x86-64.zip",
        "p31335037_122010_Linux-x86-64.zip",
        "p32327201_122010_Linux-x86-64.zip",
        "p32327208_122010_Generic.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.12.2.0.1",
        "oracleBase": "/home/oracle"
    }
}
```

Example Oracle Database 18c のサンプル CEV マニフェスト

次の Oracle Database 18c の 2021 年 10 月 PSU の例では、RDS Custom が p32126855、p28730253、p27539475 などを順に適用しています。この例では、UNIX ユーザーとグループ、Oracle ホームと Oracle ベースに新しい値を設定します。

```
{
    "mediaImportTemplateVersion":"2020-08-14",
    "databaseInstallationFileNames":[
        "V978967-01.zip"
    ],
    "opatchFileNames":[
        "p6880880_180000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames":[
        "p32126855_180000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames":[
        "p28730253_180000_Linux-x86-64.zip",
        "p27539475_1813000DBRU_Linux-x86-64.zip",
        "p29213893_180000_Generic.zip",
        "p29374604_1813000DBRU_Linux-x86-64.zip",
        "p29782284_180000_Generic.zip",
        "p28125601_180000_Linux-x86-64.zip",
        "p28852325_180000_Linux-x86-64.zip",
```

```
        "p29997937_180000_Linux-x86-64.zip",
        "p31335037_180000_Linux-x86-64.zip",
        "p31335142_180000_Generic.zip"
    ]
    "installationParameters": {
        "unixGroupName": "dba",
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/18.0.0.0.ru-2020-10.rur-2020-10.r1",
        "oracleBase": "/home/oracle/"
    }
}
```

Example Oracle Database 19c のサンプル CEV マニフェスト

次の Oracle Database 19c の例では、RDS Custom が p32126828、p29213893、p29782284、などを順に適用しています。この例では、UNIX ユーザーとグループ、Oracle ホームと Oracle ベースに新しい値を設定します。

```
{
    "mediaImportTemplateVersion": "2020-08-14",
    "databaseInstallationFileNames": [
        "V982063-01.zip"
    ],
    "opatchFileNames": [
        "p6880880_190000_Linux-x86-64.zip"
    ],
    "psuRuPatchFileNames": [
        "p32126828_190000_Linux-x86-64.zip"
    ],
    "otherPatchFileNames": [
        "p29213893_1910000DBRU_Generic.zip",
        "p29782284_1910000DBRU_Generic.zip",
        "p28730253_190000_Linux-x86-64.zip",
        "p29374604_1910000DBRU_Linux-x86-64.zip",
        "p28852325_190000_Linux-x86-64.zip",
        "p29997937_190000_Linux-x86-64.zip",
        "p31335037_190000_Linux-x86-64.zip",
        "p31335142_190000_Generic.zip"
    ],
    "installationParameters": {
        "unixGroupName": "dba",
```

```
        "unixGroupId": 12345,
        "unixUname": "oracle",
        "unixUid": 12345,
        "oracleHome": "/home/oracle/oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1",
        "oracleBase": "/home/oracle"
    }
}
```

ステップ 6 (オプション): CEV マニフェストを検証する

オプションで、マニフェストが有効な JSON ファイルであることを`json.tool`Python スクリプトを実行して確認します。例えば、`manifest.json`という名前の CEV マニフェストを含むディレクトリに変更した場合、以下のコマンドを実行します。

```
python -m json.tool < manifest.json
```

ステップ 7: 必要な IAM アクセス許可を追加する

CEV を作成する IAM プリンシパルに、ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する で説明されている必要なポリシーがあることを確認してください。

CEV の作成

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

AWS マネジメントコンソールまたはAWS CLIを使用して、CEV を作成できます。マルチテナントまたは非マルチテナントアーキテクチャのいずれかを指定します。詳細については、「マルチテナントアーキテクチャの考慮事項」を参照してください。

通常、CEV の作成には約 2 時間かかります。CEV を作成したら、それを使用して RDS Custom DB インスタンスを作成またはアップグレードできます。詳細については、「RDS Custom for Oracle DB インスタンスを作成する」および「RDS Custom for Oracle DB インスタンスのアップグレード」を参照してください。

> **Note**
>
> DB インスタンスが現在 Oracle Linux 7.9 を使用している場合は、最新の AMI を使用する新しい CEV を作成して、Oracle Linux 8 を使用します。次に、新しい CEV を使用するようにインスタンスを変更します。

CEV の作成に関する次の要件と制限事項に注意してください。

- インストールファイルを含む Amazon S3 バケットは CEV と同じ AWS リージョン にある必要があります。そうではない場合、作成プロセスは失敗します。
- CEV 名は、*major-engine-version*.*customized_string* の形式にする必要があります (例: 19.cdb_cev1)。
- CEV 名には、1～50 個の英数字、アンダースコア、ダッシュ、またはピリオドを含める必要があります。
- CEV 名には、19..cdb_cev1 のように連続したピリオドを含めることはできません。

コンソール

CEV を作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。

   カスタムエンジンバージョンページには、現在存在するすべての CEV が表示されます。CEV をまだ作成していない場合、ページは空です。
3. 「カスタムエンジンバージョンの作成」を選択します。
4. [エンジンオプション] で、次の操作を行います。

   a. [エンジンのタイプ] で、[Oracle] を選択します。

   b. [アーキテクチャ設定] で、オプションとして [マルチテナントアーキテクチャ] を選択し、DB エンジン custom-oracle-ee-cdb または custom-oracle-se2-cdb を使用する Oracle マルチテナント CEV を作成します。RDS Custom for Oracle CDB はマルチテナント CEV でのみ作成できます。このオプションを選択しない場合、CEV はエンジン custom-oracle-ee または custom-oracle-se2 を使用する非 CDB です。

> **Note**
>
> 選択したアーキテクチャは、CEV の永続的な特性になります。後で別のアーキテクチャを使用するように CEV を変更することはできません。

c. 次のいずれかのオプションを選択します。

- 新規 CEV の作成 — CEV を最初から作成します。この場合、データベースバイナリを指定する JSON マニフェストを指定する必要があります。
- ソースからの CEV の作成 — [コピーする CEV を指定してください] で、ソース CEV として使用する既存の CEV を選択します。この場合、新しい Amazon マシンイメージ (AMI) を指定できますが、別のデータベースバイナリを指定することはできません。

d. [エンジンバージョン] で、エンジンのメジャーバージョンを選択します。

5. [バージョンの詳細] で次の操作を行います。

a. 有効な名前を [カスタムエンジンバージョン名] に入力します。例えば、 名前を**19.cdb_cev1**と入力します。

b. (オプション) CEV の説明を入力します。

6. [インストールメディア] で、次の操作を行います。

a. (オプション) [AMI ID] に、サービス提供の最新のAMIを使用する場合はフィールドを空白のままにするか、以前に CEV の作成に使用した AMI を入力します。有効な AMI ID を取得するには、次のいずれかの方法を使用します。

- コンソールで、左側のナビゲーションペインで [カスタムエンジンバージョン] を選択し、CEV の名前を選択します。CEV が使用する AMI ID が [設定] タブに表示されます。
- AWS CLI で、`describe-db-engine-versions` コマンドを使用します。`ImageID` の出力を検索します。

b. マニフェストファイルの S3 の場所で、ステップ 3: Amazon S3 へのインストールファイルをアップロードするで指定した Amazon S3 バケットの場所を入力します。例えば、 **s3://my-custom-installation-files/123456789012/cev1/**と入力します。

> **Note**
>
> CEV を作成する AWS リージョン は、S3 バケットとしてのリージョンと同じである必要があります。

c. (新規 CEV のみを作成) [CEV マニフェスト] には、CEV マニフェストの作成 で作成した JSON マニフェストを入力します。

7. [KMS キー] セクションで、[キー ARN の入力] を選択し、利用可能な AWS KMS キーをリストアップします。次に、リストから KMS キーを選択します。

   RDS Custom にはAWS KMSキーが必要です。詳細については、「ステップ 1: 対称暗号化 AWS KMS キーを作成または再利用する」を参照してください。

8. (オプション) [新しいタグを追加] を選択して、CEV のキーと値のペアを作成します。

9. [カスタムエンジンバージョンの作成] を選択します。

   JSON マニフェストの形式が無効な場合は、コンソールに「CEV マニフェストの検証中にエラーが発生しました」と表示されます。問題を修正して、再試行してください。

カスタムエンジンバージョンページが表示されます。CEV が作成ステータスとともに表示されます。CEV の作成プロセスは、約 2 時間継続します。

AWS CLI

AWS CLI を使用して CEV を作成するには、create-custom-db-engine-version コマンドを実行します。

以下のオプションは必須です。

- `--engine` — エンジンタイプを指定します。CDB の場合は、`custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` を指定します。非 CDB の場合は、`custom-oracle-ee` または `custom-oracle-se2` を指定します。CDB は、`custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` で作成した CEV からのみ作成できます。非 CDB は、`custom-oracle-ee` または `custom-oracle-se2` で作成した CEV からのみ作成できます。
- `--engine-version` — エンジンバージョンを指定します。形式は、*`major-engine-version`*、*`customized_string`* です。CEV 名には、1～50 個の英数字、アンダースコア、ダッシュ、またはピリオドを含める必要があります。CEV 名には、`19..cdb_cev1` のように連続したピリオドを含めることはできません。
- `--kms-key-id` – AWS KMS key を指定します。
- `--manifest` - *`manifest_json_string`* または `--manifest file:`*`file_name`* のいずれかを指定します。改行文字は、*`manifest_json_string`* で許可されていません。JSON コードでは、必ず、二重引用符 (") の前にバックスラッシュ (\) 付けてエスケープしてください。

次の例は、「ステップ 5: CEV マニフェストを準備する」の 19c の *manifest_json_string* について示しています。この例では、Oracle ベース、Oracle ホーム、UNIX/Linux ユーザーとグループの ID と名前に新しい値を設定します。次の文字列をコピーする場合は、コマンドに貼り付ける前にすべての改行文字を削除してください。

```
"{\"mediaImportTemplateVersion\": \"2020-08-14\",
\"databaseInstallationFileNames\": [\"V982063-01.zip\"],
\"opatchFileNames\": [\"p6880880_190000_Linux-x86-64.zip\"],
\"psuRuPatchFileNames\": [\"p32126828_190000_Linux-x86-64.zip\"],
\"otherPatchFileNames\": [\"p29213893_1910000DBRU_Generic.zip\",
\"p29782284_1910000DBRU_Generic.zip\",\"p28730253_190000_Linux-
x86-64.zip\",\"p29374604_1910000DBRU_Linux-x86-64.zip\",
\"p28852325_190000_Linux-x86-64.zip\",\"p29997937_190000_Linux-x86-64.zip
\",\"p31335037_190000_Linux-x86-64.zip\",\"p31335142_190000_Generic.zip
\"]\"installationParameters\":{ \"unixGroupName\":\"dba\",
\ \"unixUname\":\"oracle\", \ \"oracleHome\":\"/home/oracle/
oracle.19.0.0.0.ru-2020-04.rur-2020-04.r1.EE.1\", \ \"oracleBase\":\"/
home/oracle/\"}}"
```

- `--database-installation-files-s3-bucket-name` — ステップ 3: Amazon S3 へのインストールファイルをアップロードする で指定したのと同じバケット名を指定します。`create-custom-db-engine-version` を実行する AWS リージョン は、Amazon S3 バケットと同じリージョンにある必要があります。

また、以下のオプションを指定することもできます。

- `--description` — CEV の説明を指定します。
- `--database-installation-files-s3-prefix` — ステップ 3: Amazon S3 へのインストールファイルをアップロードする で指定したのと同じフォルダー名を指定します。
- `--image-id` — 再利用する AMI ID を指定します。有効な ID を見つけるには、`describe-db-engine-versions` コマンドを実行し、`ImageID` の出力を検索します。デフォルトでは、RDS Custom for Oracle は入手可能な最新の AMI を使用します。

次の例では、`19.cdb_cev1` という名前の Oracle マルチテナント CEV を作成します。この例では、入手可能な最新の AMI を使用するのではなく、既存の AMI を再利用しています。CEV の名前が、エンジンのメジャーバージョン番号で始まっていることを確認してください。

Example

Linux、macOS、Unix の場合:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-se2-cdb \
    --engine-version 19.cdb_cev1 \
    --database-installation-files-s3-bucket-name us-east-1-123456789012-custom-
installation-files \
    --database-installation-files-s3-prefix 123456789012/cev1 \
    --kms-key-id my-kms-key \
    --description "test cev" \
    --manifest manifest_string \
    --image-id ami-012a345678901bcde
```

Windows の場合:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-se2-cdb ^
    --engine-version 19.cdb_cev1 ^
    --database-installation-files-s3-bucket-name us-east-1-123456789012-custom-
installation-files ^
    --database-installation-files-s3-prefix 123456789012/cev1 ^
    --kms-key-id my-kms-key ^
    --description "test cev" ^
    --manifest manifest_string ^
    --image-id ami-012a345678901bcde
```

Example

`describe-db-engine-versions`コマンドを実行してCEV に関する詳細を入手します。

```
aws rds describe-db-engine-versions \
    --engine custom-oracle-se2-cdb \
    --include-all
```

次の部分出力例は、エンジン、パラメータグループ、マニフェストおよびその他の情報を示しています。

```
{
    "DBEngineVersions": [
```

```
        {
            "Engine": "custom-oracle-se2-cdb",
            "EngineVersion": "19.cdb_cev1",
            "DBParameterGroupFamily": "custom-oracle-se2-cdb-19",
            "DBEngineDescription": "Containerized Database for Oracle Custom SE2",
            "DBEngineVersionDescription": "test cev",
            "Image": {
                "ImageId": "ami-012a345678901bcde",
                "Status": "active"
            },
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "MajorEngineVersion": "19",
            "DatabaseInstallationFilesS3BucketName": "us-east-1-123456789012-custom-
installation-files",
            "DatabaseInstallationFilesS3Prefix": "123456789012/cev1",
            "DBEngineVersionArn": "arn:aws:rds:us-east-1:123456789012:cev:custom-
oracle-se2-cdb/19.cdb_cev1/abcd12e3-4f5g-67h8-i9j0-k1234l56m789",
            "KMSKeyId": "arn:aws:kms:us-
east-1:732027699161:key/1ab2345c-6d78-9ef0-1gh2-3456i7j89k01",
            "CreateTime": "2023-03-07T19:47:58.131000+00:00",
            "TagList": [],
            "SupportsBabelfish": false,
...
```

CEV 作成の失敗

CEV の作成プロセスが失敗した場合、RDS Custom はメッセージ `Creation failed for custom engine version` *`major-engine-version.cev_name`* を含む RDS-EVENT-0198 を発行します。そこには、失敗に関する詳細が含まれています。例えば、イベントは不足ファイルを出力します。

障害が発生した CEV を変更することはできません。削除して、失敗の原因を解決してから再度 CEV の作成を試みるしか方法がありません。CEV 作成の障害発生理由のトラブルシューティングについては、RDS Custom for Oracle のカスタムエンジンバージョン作成のトラブルシューティングを参照してください。

CEV ステータスの変更

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

AWS マネジメントコンソールまたはAWS CLIを使用して CEV を変更できます。CEV の説明またはそのアベイラビリティーステータスを変更できます。CEV には、次のいずれかのステータス値があります。

- `available` - この CEV を使用して、新しい RDS Custom DB インスタンスを作成するか、DB インスタンスをアップグレードできます。これは、新規作成された CEV のデフォルトステータスです。
- `inactive` - この CEV で RDS Custom インスタンスを作成したいアップグレードしたりすることはできません。この CEV を使用して新しい RDS Custom DB インスタンスを作成するために、DB スナップショットを復元することはできません。

CEV は、サポート対象のステータスから、他のサポート対象ステータスに変更できます。ステータスを変更することで、CEV の誤用を防いだり、中止された CEV を再び使用したりできるようになります。例えば、CEV のステータスを`available`から`inactive`に、`inactive`から`available`に変更したりします。

コンソール

CEV を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「カスタムエンジンバージョン」 を選択します。
3. 説明またはステータスを変更する CEV を選択します。
4. [アクション] で、[変更] を選択します。
5. 次の変更のいずれか、あるいはすべてを実行します。

- CEV ステータス設定で、新しいアベイラビリティーステータスを選択します。
- Version description (バージョンの説明) ページで、新しい説明を入力します。

6. [Modify State] (状態の変更) を選択します。

   CEV が使用中の場合は、コンソールに「CEV ステータスを変更することはできません」と表示されます。問題を修正して、再試行してください。

カスタムエンジンバージョンページが表示されます。

AWS CLI

AWS CLIを使用して CEV を変更するには、modify-custom-db-engine-versionコマンドを実行します。db-engine-versionsコマンドを実行して、変更するCEVを検索できます。

以下のオプションは必須です。

- `--engine` *`engine-type`* (*`engine-type`* は `custom-oracle-ee`、`custom-oracle-se2`、`custom-oracle-ee-cdb`、または `custom-oracle-se2-cdb`)
- *`cev`*が変更するカスタムエンジンバージョンの名前である`--engine-version` *`cev`*です。
- *`status`*がCEV に割り当てるアベイラビリティーステータスを示す`--status` *`status`*です。

次の例では、`19.my_cev1` という名前の CEV を現在のステータスから `inactive` に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-custom-db-engine-version \
    --engine custom-oracle-se2 \
    --engine-version 19.my_cev1 \
    --status inactive
```

Windows の場合:

```
aws rds modify-custom-db-engine-version ^
    --engine custom-oracle-se2 ^
    --engine-version 19.my_cev1 ^
```

```
    --status inactive
```

Amazon RDS Custom for Oracle の CEV 詳細を表示する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

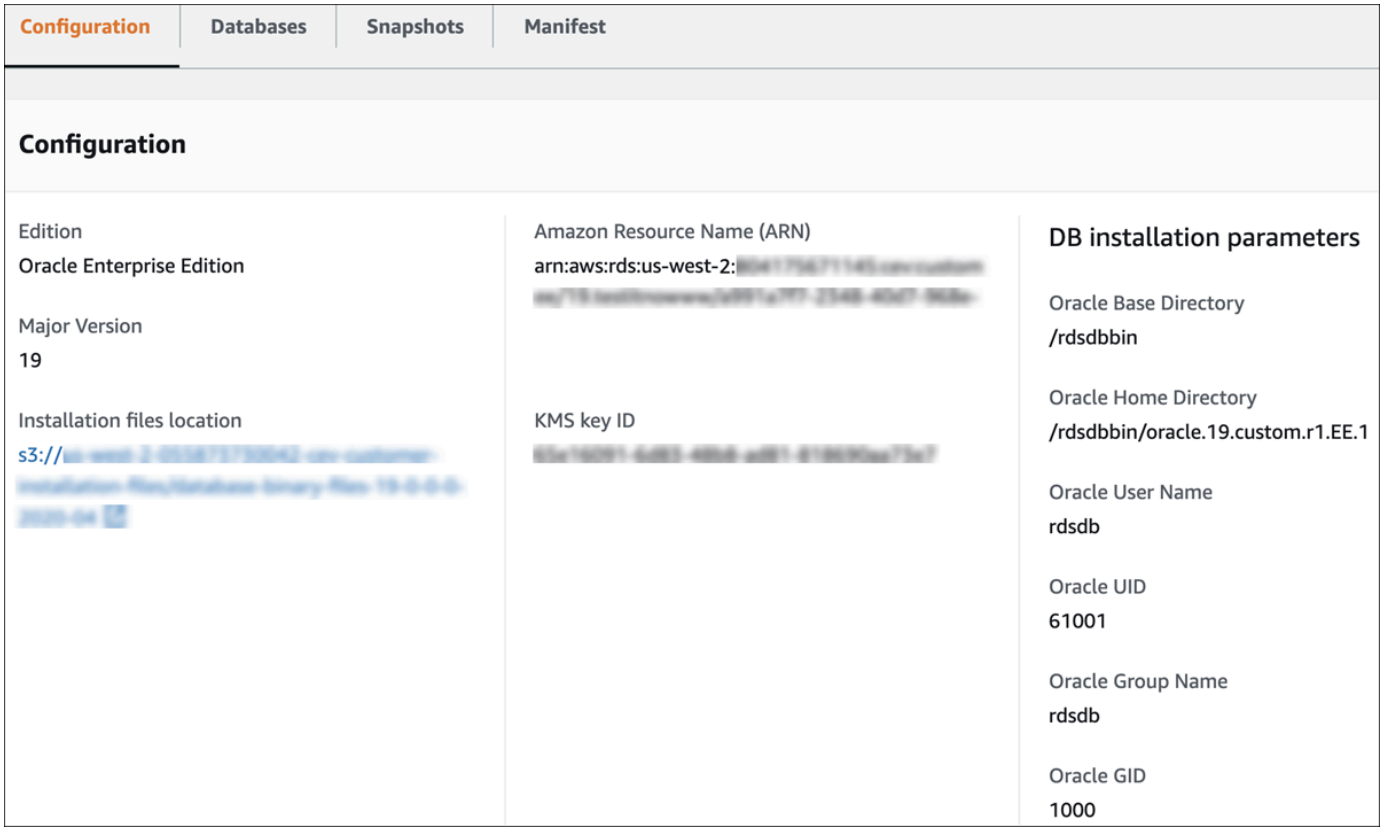
AWS マネジメントコンソール またはAWS CLI を使用して、CEV マニフェストと CEV の作成に使用したコマンドの詳細を表示できます。

コンソール

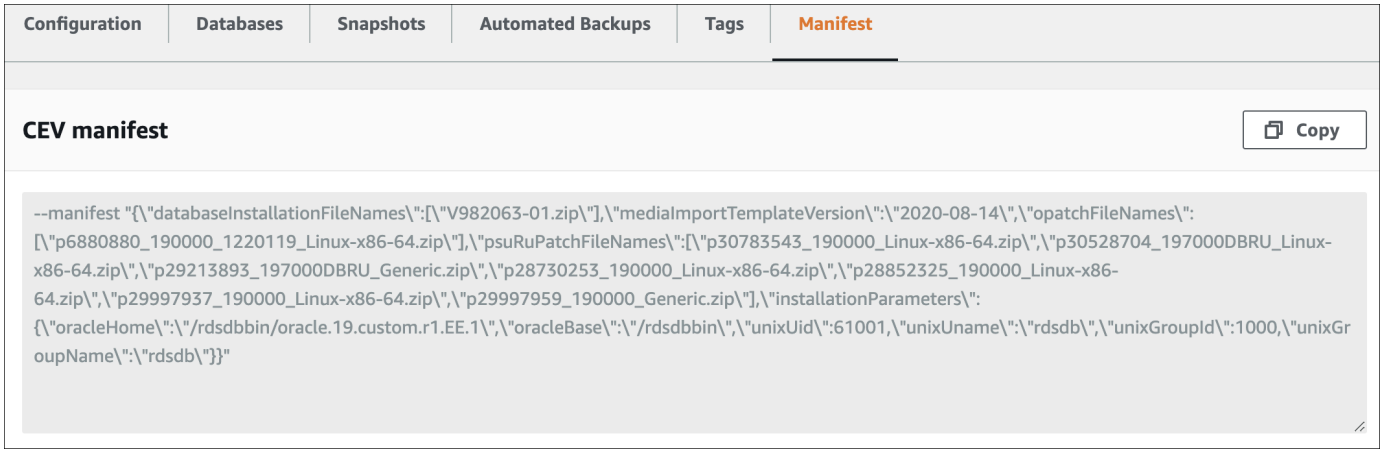
CEV の詳細を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。

   カスタムエンジンバージョンページには、現在存在するすべての CEV が表示されます。CEV をまだ作成していない場合、ページは空です。
3. 表示する CEV の名前を選択します。
4. [Configuration] (設定) を選択して、マニフェストに指定されているインストールパラメータが表示を表示します。



5. [Manifest] (マニフェスト) を選択して、`create-custom-db-engine-version` コマンドの `--manifest` オプションで指定されているインストールパラメータを表示します。このテキストをコピーし、必要に応じて値を置き換えて、新しいコマンドで使用できます。



AWS CLI

AWS CLI を使用して CEV に関する詳細を表示するには、describe-db-engine-versions コマンドを実行します。

以下のオプションは必須です。

- `--engine` *engine-type* (*engine-type* は custom-oracle-ee、custom-oracle-se2、custom-oracle-ee-cdb、または custom-oracle-se2-cdb)
- `--engine-version` *major-engine-version.customized_string*

次の例では、Enterprise Edition を使用する非 CDB の CEV を作成します。CEV の名前 19.my_cev1 は、メジャーエンジンバージョン番号で始めます (必須)。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev1
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev1
```

次の部分出力例は、エンジン、パラメータグループ、マニフェストおよびその他の情報を示しています。

```
"DBEngineVersions": [
    {
        "Engine": "custom-oracle-ee",
        "MajorEngineVersion": "19",
        "EngineVersion": "19.my_cev1",
        "DatabaseInstallationFilesS3BucketName": "us-east-1-123456789012-cev-customer-
installation-files",
        "DatabaseInstallationFilesS3Prefix": "123456789012/cev1",
        "CustomDBEngineVersionManifest": "{\n\"mediaImportTemplateVersion\":
 \"2020-08-14\",\n\"databaseInstallationFileNames\": [\n\"V982063-01.zip\"\n],
\n\"installationParameters\": {\n\"oracleBase\":\"/tmp\",\n\"oracleHome\":\"/
tmp/Oracle\"\n},\n\"opatchFileNames\": [\n\"p6880880_190000_Linux-x86-64.zip
\"\n],\n\"psuRuPatchFileNames\": [\n\"p32126828_190000_Linux-x86-64.zip
\"\n],\n\"otherPatchFileNames\": [\n\"p29213893_1910000DBRU_Generic.zip\",\n
```

```
\"p29782284_1910000DBRU_Generic.zip\",\n\"p28730253_190000_Linux-x86-64.zip\",\n
\"p29374604_1910000DBRU_Linux-x86-64.zip\",\n\"p28852325_190000_Linux-x86-64.zip\",
\n\"p29997937_190000_Linux-x86-64.zip\",\n\"p31335037_190000_Linux-x86-64.zip\",\n
\"p31335142_190000_Generic.zip\"\n]\n}\n",
        "DBParameterGroupFamily": "custom-oracle-ee-19",
        "DBEngineDescription": "Oracle Database server EE for RDS Custom",
        "DBEngineVersionArn": "arn:aws:rds:us-west-2:123456789012:cev:custom-oracle-
ee/19.my_cev1/0a123b45-6c78-901d-23e4-5678f901fg23",
        "DBEngineVersionDescription": "test",
        "KMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/ab1c2de3-f4g5-6789-h012-
h3ijk4567l89",
        "CreateTime": "2022-11-18T09:17:07.693000+00:00",
        "ValidUpgradeTarget": [
        {
            "Engine": "custom-oracle-ee",
            "EngineVersion": "19.cev.2021-01.09",
            "Description": "test",
            "AutoUpgrade": false,
            "IsMajorVersionUpgrade": false
        }
]
```

CEV の削除

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

AWS マネジメントコンソール またはAWS CLIを使用して、CEVを削除できます。これには通常数分かかります。

CEV を削除するには、CEVが次のいずれかで CEV を使用中であってはいけません。

- RDS Custom DB インスタンスを停止します。
- RDS Custom DB インスタンスのスナップショット
- RDS Custom DB インスタンスの自動バックアップ

コンソール

CEV を削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。
3. 説明またはステータスを削除する CEV を選択します。
4. [アクション] で、[削除] を選択します。

   [Delete ***cev_name***?] (cev_name を削除しますか?) ダイアログボックスが表示されます。
5. 「**delete me**」と入力し、[削除] を選択します。

   カスタムエンジンバージョンページで、バナーに CEV が削除中であることが示されます。

AWS CLI

AWS CLIを使用して CEV を削除するには、delete-custom-db-engine-versionコマンドを実行します。

以下のオプションは必須です。

- `--engine` *engine-type* (*engine-type* は `custom-oracle-ee`、`custom-oracle-se2`、`custom-oracle-ee-cdb`、または `custom-oracle-se2-cdb`)
- *CEV*が削除するカスタムエンジンバージョンの名前である`--engine-version` *cev*です。

次の例では、`19.my_cev1` という名前の CEV を削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev1
```

Windows の場合:

```
aws rds delete-custom-db-engine-version ^
```

```
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev1
```

Amazon RDS Custom for Oracle DB インスタンスの設定

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle の DB インスタンスを作成し、セキュアシェル (SSH) またはAWS Systems Managerを使用してそれに接続できます。

RDS Custom for Oracle DB インスタンスへの接続とログインの詳細については、次のトピックを参照してください。

- Session Manager を使用して RDS Custom DB インスタンスに接続する
- SSH を使用した RDS Custom DB インスタンスへの接続
- RDS Custom for Oracle データベースに SYS として ログインします。

RDS Custom for Oracle DB インスタンスを作成する

AWS マネジメントコンソールまたはAWS CLIのいずれかを使用して、Amazon RDS Custom for Oracle DB インスタンスを作成します。この手順は、Amazon RDS DB インスタンスの作成手順と似ています。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

CEV マニフェストにインストールパラメータを含めた場合、DB インスタンスは Oracle ベース、Oracle ホーム、指定した UNIX/Linux ユーザーとグループの ID と名前を使用します。この `oratab` ファイルは、インストール中に Oracle Database によって作成され、シンボリックリンクではなく、実際のインストール先を指しています。RDS Custom for Oracle がコマンドを実行するときには、デフォルトユーザーの `rdsdb` ではなく、設定された OS ユーザーとして実行します。詳細については、「ステップ 5: CEV マニフェストを準備する」を参照してください。

RDS Custom DB インスタンスの作成、または接続を試行する前に、Amazon RDS Custom for Oracle の環境設定 のタスクを完了してください。

コンソール

RDS Custom for Oracle DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. [データベースの作成] を選択します。
4. [Choose a database creation method] (データベース作成方法を選択) で [Standard Create] (スタンダード作成) を選択します。
5. [エンジンオプション] セクションで、次の操作を行います。

   a. [エンジンのタイプ] で、[Oracle] を選択します。

   b. データベース管理のタイプで、Amazon RDS Customを選択します。

   c. [アーキテクチャ設定] で、以下のいずれかを実行します。

      - [マルチテナントアーキテクチャ] を選択して、コンテナデータベース (CDB) を作成します。作成時には、CDB には 1 つの PDB シードと 1 つの初期 PDB が含まれています。

        > **Note**
        >
        > [マルチテナントアーキテクチャ] 設定は、Oracle Database 19c でのみサポートされています。

      - [マルチテナントアーキテクチャ] をクリアして、非 CDB を作成します。非 CDB には、PDB を含めることはできません。

   d. [エディション] で、[Oracle Enterprise Edition] または [Oracle Standard Edition 2] を選択します。

   e. [カスタムエンジンバージョン] で、既存の RDS Custom カスタムエンジンバージョン (CEV) を選択します。CEV の形式は *major-engine-version.customized_string* のようになっています。識別子の例は`19.cdb_cev1`のとおりです。

      前のステップで [マルチテナントアーキテクチャ] を選択した場合は、`custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` エンジンタイプを使用する CEV のみを指定できます。コンソールは、異なるエンジンタイプで作成された CEV を除外します。

6. 「テンプレート」 では、「作成」 を選択します。
7. [Settings] (設定) セクションで、以下の手順を実行します:

a. [DB インスタンス識別子] として、DB インスタンスの一意の名前を入力します。

b. [マスターユーザー名] で、ユーザー名を入力します。この値は後でコンソールから取得できます。

非 CDB に接続する場合、マスターユーザーは非 CDB のユーザーです。CDB に接続する場合、マスターユーザーは PDB のユーザーです。CDB ルートに接続するには、ホストにログインして SQL クライアントを起動し、SQL コマンドを使用して管理ユーザーを作成します。

c. [パスワードの自動生成] をオフにします。

8. [DB インスタンスクラス] を選択します。

サポートされているクラスについては、「RDS Custom for Oracle での DB インスタンスクラスのサポート」を参照してください。

9. [Storage] (ストレージ) セクションで、以下の操作を行います。

a. [ストレージタイプ] には、SSD タイプ (io1、io2、gp2、または gp3) を選択します。次の追加オプションがあります。

- io1、io2、または gp3 の場合は、[プロビジョンド IOPS] のレートを選択します。デフォルトは、io1 および io2 が 1000、gp3 が 12000 です。
- gp3 の場合は、[ストレージスループット] のレートを選択します。デフォルトは 500 MiBps です。

b. [割り当てられたストレージ] で、ストレージサイズを選択します。デフォルトは 40 GiB です。

10. [接続] で、[Virtual Private Cloud (VPC)]、[DB サブネットグループ]、[VPC セキュリティグループ (ファイアウォール)] を指定します。

11. 「RDS カスタムセキュリティ」 で、以下を実行します。

a. IAM インスタンスプロファイルで、RDS Custom for Oracle DB インスタンスのインスタンスプロファイルを選択します。

IAM インスタンスプロファイルは、`AWSRDSCustom`で始まる必要があります。例えば、*`AWSRDSCustomInstanceProfileForRdsCustomInstance`*です。

b. 「暗号化」で、キー ARN を入力を選択し、使用可能なAWS KMSキーを一覧表示します。次に、リストからキーを選択します。

AWS KMSRDS Custom にはキーが必要です。詳細については、「ステップ 1: 対称暗号化 AWS KMS キーを作成または再利用する」を参照してください。

12. [データベースオプション] で、次の操作を行います。

    a. (オプション) [システム ID (SID)] に、Oracle SID の値を入力します。これは CDB の名前でもあります。SID は、データベースファイルを管理する Oracle データベースインスタンスの名前です。この場合、「Oracle データベースインスタンス」という用語は、システムグローバルエリア (SGA) と Oracle バックグラウンドプロセスのみを指します。SID を指定しなかった場合、値はデフォルトで **RDSCDB** に設定されます。

    b. (オプション) [最初のデータベース名] に名前を入力します。デフォルト値は **ORCL** です。マルチテナントアーキテクチャでは、最初のデータベース名は PDB 名です。

    > **Note**
    >
    > SID と PDB 名は異なる必要があります。

    c. [オプショングループ] で、オプショングループを選択するか、デフォルトを受け入れます。

    > **Note**
    >
    > RDS Custom for Oracle でサポートされているオプションは `Timezone` のみです。詳細については、「Oracle のタイムゾーン」を参照してください。

    d. [バックアップ保持期間] で、値を選択します。[0 日] は選択できません。

    e. 残りのセクションで、 RDS Custom DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。次の設定はコンソールに表示されず、サポート対象外です。

    - プロセッサの機能
    - ストレージのオートスケーリング
    - データベース認証のパスワードと Kerberos 認証のオプション (パスワード認証のみサポートされています)
    - Performance Insights
    - ログのエクスポート
    - マイナーバージョン自動アップグレードの有効化
    - 削除保護

13. [データベースの作成] を選択します。

> **Important**
>
> RDS Custom for Oracle DB インスタンスを作成する際に、次のエラーが表示される場合があります。「サービスリンクロールが作成中です。後ほどもう一度試してください。」 表示された場合は、数分間待ってから DB インスタンスの作成を再試行します。

認証情報の詳細の表示ボタンがデータベースページに表示されます。

RDS Custom DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

マスターユーザーとして DB インスタンスに接続するには、表示されているユーザー名およびパスワードを使用します。

> **Important**
>
> コンソールでマスターユーザーパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。RDS Custom DB インスタンスが使用可能になった後にマスターユーザーパスワードを変更するには、データベースにログインして、`ALTER USER` コマンドを実行します。コンソールで [変更] を使用してパスワードをリセットすることはできません。

14. データベースを選択して、RDS Custom DB インスタンスのリストを表示します。
15. 先ほど作成した RDS Custom DB インスタンスを選択します。

RDS コンソールに、新規の RDS Custom DB インスタンスの詳細が表示されます。

- RDS Custom DB インスタンスが作成されて使用できるようになるまで、DB インスタンスのステータスは [作成中] となります。ステータスが [利用可能] に変わると、DB インスタンスに接続できます。インスタンスクラスと割り当てられたストレージによっては、新規の DB インスタンスを使用できるようになるまで数分かかることがあります。
- ロールにはインスタンス (RDS Custom)という値があります。
- [RDS カスタムオートメーションモード] には [完全なオートメーション] という値があります。この設定は、DB インスタンスが自動モニタリングとインスタンスの回復を提供することを意味します。

AWS CLI

RDS Custom DB インスタンスは、create-db-instance AWS CLI コマンドを使用して作成します。

以下のオプションは必須です。

- `--db-instance-identifier`
- `--db-instance-class` (サポートされている DB インスタンスクラスのリストについては、「」を参照してください)RDS Custom for Oracle での DB インスタンスクラスのサポート
- `--engine` *`engine-type`* (*`engine-type`* は `custom-oracle-ee`、`custom-oracle-se2`、`custom-oracle-ee-cdb`、または `custom-oracle-se2-cdb`)
- `--engine-version` *`cev`* (*`cev`*は CEV の作成で指定したカスタムエンジンバージョンの名前です)
- `--kms-key-id` *`my-kms-key`*
- `--backup-retention-period` *`days`* (*`days`* は `0` より大きい値です)
- `--no-auto-minor-version-upgrade`
- `--custom-iam-instance-profile AWSRDSCustomInstanceProfile-`*`us-east-1`* (*`region`* は DB インスタンスを作成する AWS リージョン です)

以下の例では、`my-cfo-cdb-instance` という名前の RDS Custom DB インスタンスを作成します。データベースは、デフォルト以外の *`MYCDB`* という名前の CDB です。デフォルト以外の PDB 名は *`MYPDB`* です。バックアップ保持期間は 3 日間です。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --engine custom-oracle-ee-cdb \
    --db-instance-identifier my-cfo-cdb-instance \
    --engine-version 19.cdb_cev1 \
    --db-name MYPDB \
    --db-system-id MYCDB \
    --allocated-storage 250 \
    --db-instance-class db.m5.xlarge \
    --db-subnet-group mydbsubnetgroup \
    --master-username myuser \
    --master-user-password mypassword \
    --backup-retention-period 3 \
```

```
    --port 8200 \
    --kms-key-id my-kms-key \
    --no-auto-minor-version-upgrade \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfile-us-east-1
```

Windows の場合:

```
aws rds create-db-instance ^
    --engine custom-oracle-ee-cdb ^
    --db-instance-identifier my-cfo-cdb-instance ^
    --engine-version 19.cdb_cev1 ^
    --db-name MYPDB ^
    --db-system-id MYCDB ^
    --allocated-storage 250 ^
    --db-instance-class db.m5.xlarge ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username myuser ^
    --master-user-password mypassword ^
    --backup-retention-period 3 ^
    --port 8200 ^
    --kms-key-id my-kms-key ^
    --no-auto-minor-version-upgrade ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfile-us-east-1
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

`describe-db-instances` コマンドを使用して、インスタンスの詳細を入手します。

Example

```
aws rds describe-db-instances --db-instance-identifier my-cfo-cdb-instance
```

次の部分出力は、エンジン、パラメータグループ、およびその他の情報を示しています。

```
        {
            "DBInstanceIdentifier": "my-cfo-cdb-instance",
            "DBInstanceClass": "db.m5.xlarge",
            "Engine": "custom-oracle-ee-cdb",
```

```
            "DBInstanceStatus": "available",
            "MasterUsername": "admin",
            "DBName": "MYPDB",
            "DBSystemID": "MYCDB",
            "Endpoint": {
                "Address": "my-cfo-cdb-instance.abcdefghijkl.us-
east-1.rds.amazonaws.com",
                "Port": 1521,
                "HostedZoneId": "A1B2CDEFGH34IJ"
            },
            "AllocatedStorage": 100,
            "InstanceCreateTime": "2023-04-12T18:52:16.353000+00:00",
            "PreferredBackupWindow": "08:46-09:16",
            "BackupRetentionPeriod": 7,
            "DBSecurityGroups": [],
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-0a1bcd2e",
                    "Status": "active"
                }
            ],
            "DBParameterGroups": [
                {
                    "DBParameterGroupName": "default.custom-oracle-ee-cdb-19",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
...
```

マルチテナントアーキテクチャの考慮事項

Oracle マルチテナントアーキテクチャ (`custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` エンジンタイプ) で Amazon RDS Custom for Oracle DB インスタンスを作成する場合、データベースはコンテナデータベース (CDB) になります。Oracle マルチテナントアーキテクチャを指定しない場合、データベースは `custom-oracle-ee` または `custom-oracle-se2` エンジンタイプを使用する従来の非 CDB になります。非 CDB には、プラグ可能なデータベース (PDB) を含めることはできません。詳細については、「Amazon RDS Custom for Oracle のデータベースアーキテクチャ」を参照してください。

RDS Custom for Oracle CDB インスタンスを作成するときには、次の点を考慮してください。

- マルチテナントデータベースは、Oracle Database 19c CEV からのみ作成できます。

- CDB インスタンスを作成できるのは、CEV が `custom-oracle-ee-cdb` または `custom-oracle-se2-cdb` エンジンタイプを使用している場合のみです。
- Standard Edition 2 を使用して CDB インスタンスを作成する場合、CDB には最大 3 つの PDB を含めることができます。
- デフォルトでは、CDB には Oracle System ID (Oracle SID) と同じ `RDSCDB` という名前が付けられます。別の名前を選択できます。
- CDB には初期 PDB が 1 つだけ含まれています。PDB のデフォルト名は、`ORCL` です。初期 PDB に別の名前を選択できますが、Oracle SID と PDB の名前を同じにすることはできません。
- RDS Custom for Oracle は PDB 用の API を提供していません。追加の PDB を作成するには、Oracle SQL コマンド `CREATE PLUGGABLE DATABASE` を使用してください。RDS Custom for Oracle では、作成可能な PDB 数に制限はありません。一般に、オンプレミスデプロイと同様、PDB の作成と管理はお客様の責任となります。
- RDS API を使用して PDB を作成、変更、削除することはできません。Oracle SQL ステートメントを使用する必要があります。Oracle SQL を使用して PDB を作成した場合は、ポイントインタイムリカバリ (PITR) を実行する必要がある場合に備えて、手動でスナップショットを取ることをお勧めします。
- Amazon RDS API を使用して既存の PDB の名前を変更することはできません。また、`modify-db-instance` コマンドを使用して CDB の名前を変更することはできません。
- CDB ルートのオープンモードは、プライマリデータベースでは `READ WRITE`、マウントされたスタンバイデータベースでは `MOUNTED` です。RDS Custom for Oracle は、CDB を開く際にすべての PDB を開こうとします。RDS Custom for Oracle がすべての PDB を開くことができない場合は、イベント `tenant database shutdown` が発行されます。

RDS Custom サービスにリンクされたロール

service-linked role は、AWS アカウント のリソースへのアクセス権を Amazon RDS Custom に付与します。これにより、必要なアクセス許可を手動で追加する必要がなくなるため、RDS Custom の使用が簡単になります。RDS Custom は、サービスにリンクされたロールのアクセス許可を定義し、別途定義されている場合を除き、RDS Custom のみがその ロールを引き受けることができます。定義されるアクセス許可には、信頼ポリシーやアクセス許可ポリシーなどがあり、そのアクセス許可ポリシーを他の IAM エンティティに添付することはできません。

RDS Custom DB インスタンスを作成すると、Amazon RDS と RDS Custom サービスにリンクされたロールの両方が作成され (まだ存在しない場合)、使用されます。詳細については、「Amazon RDS のサービスにリンクされたロールの使用」を参照してください。

RDS Custom for Oracle DB インスタンスを初めて作成するときに、「サービスにリンクされたロールが作成中です」というエラーが表示される場合があります。後ほどもう一度試してください。」表示された場合は、数分間待ってから DB インスタンスの作成を再試行します。

RDS Custom for Oracle DB インスタンスに追加のソフトウェアコンポーネントをインストールする

新しく作成された DB インスタンスでは、データベース環境には Oracle バイナリ、データベース、およびデータベースリスナーが含まれます。DB インスタンスのホストオペレーティングシステムに追加のソフトウェアをインストールしたい場合があります。例えば、Oracle Application Express (APEX)、Oracle Enterprise Manager (OEM) エージェント、または Guardium S-TAP エージェントをインストールしたい場合があります。ガイドラインと大まかな手順については、詳細な AWS ブログ記事「Amazon RDS Custom for Oracle に追加のソフトウェアコンポーネントをインストールする」を参照してください。

Session Manager を使用して RDS Custom DB インスタンスに接続する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom DB インスタンスを作成した後、AWS Systems Manager Session Manager を使用してインスタンスに接続できます。DB インスタンスが一般にアクセスできない場合に推奨される方法です。

Session Manager では、ブラウザベースのシェルまたは AWS CLI を介して、Amazon EC2 インスタンスにアクセスできます。詳細については、「AWS Systems Manager Session Manager」を参照してください。

コンソール

Session Manager を使用して DB インスタンスに接続するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、「データベース」 を選択し、停止する RDS Custom DB インスタンスを選択します。
3. [設定] を選択します。
4. DB インスタンスのリソース IDに注意してください。例えば、リソース ID はdb-ABCDEFGHIJKLMNOPQRS0123456のようになります。
5. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
6. ナビゲーションペインで、[インスタンス] を選択してください。
7. EC2 インスタンスの名前を探し、それに関連付けられているインスタンス ID をクリックします。例えば、インスタンス ID はi-abcdefghijklm01234のようになります。
8. 接続 を選択します。
9. セッションマネージャーを選択します。
10. 接続 を選択します。

    セッションのウィンドウが開きます。

AWS CLI

AWS CLIを使用して RDS Custom DB インスタンスに接続できます。この方法では、AWS CLIのセッションマネージャープラグインが必要です。プラグインをインストールする方法については、「Install the Session Manager plugin for the AWS CLI」を参照してください。

RDS Custom DB インスタンスの DB リソース ID を見つけるには、`aws rds describe-db-instances`を使用します。

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

次のサンプル出力は、RDS Custom インスタンスのリソース ID を示しています。プレフィックスはdb-です。

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

DB インスタンスの EC2 インスタンス ID を見つけるには、`aws ec2 describe-instances`を使用します。次の例ではリソース ID に db-ABCDEFGHIJKLMNOPQRS0123456 を使用しています。

```
aws ec2 describe-instances \
```

```
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

次の出力例は、EC2 インスタンス ID を示しています。

```
i-abcdefghijklm01234
```

`aws ssm start-session`コマンドで、`--target`パラメータに EC2 インスタンス ID を指定します。

```
aws ssm start-session --target "i-abcdefghijklm01234"
```

接続に成功した場合の応答は次のようになります。

```
Starting session with SessionId: yourid-abcdefghijklm1234
[ssm-user@ip-123-45-67-89 bin]$
```

SSH を使用した RDS Custom DB インスタンスへの接続

セキュアシェルプロトコル (SSH) は、セキュリティで保護されていないネットワーク上での暗号化通信をサポートするネットワークプロトコルです。RDS Custom DB インスタンスを作成すると、SSH クライアントを使用してこのインスタンスに接続できます。詳細については、「SSH を使用した Linux インスタンスへの接続」を参照してください。

SSH 接続方法は、DB インスタンスがプライベートかどうか、つまりパブリックインターネットからの接続を受け付けないかどうかによって異なります。この場合、SSH トンネリングを使用して ssh ユーティリティをインスタンスに接続する必要があります。この方法では、既存の SSH セッション内の専用データストリーム (トンネル) を使用してデータを転送します。SSH トンネリングは AWS Systems Manager を使用して設定できます。

> **Note**
>
> プライベートインスタンスへのアクセスにはさまざまな戦略がサポートされています。踏み台ホストを使用して SSH クライアントをプライベートインスタンスに接続する方法については、「AWS での Linux 踏み台ホスト」を参照してください。ポート転送を設定する方法については、「AWS Systems Manager Session Manager を使用したポート転送」を参照してください。

DB インスタンスがパブリックサブネットにあり、パブリックに利用可能な設定になっている場合は、SSH トンネリングは必要ありません。パブリック Amazon EC2 インスタンスと同じように SSH で接続できます。

SSH クライアントを DB インスタンスに接続するには、次のステップを完了します。

1. ステップ 1: SSH 接続を許可するように DB インスタンスを設定する
2. ステップ 2: SSH シークレットキーと EC2 インスタンス ID を取得する
3. ステップ 3: SSH ユーティリティを使用して EC2 インスタンスに接続します

ステップ 1: SSH 接続を許可するように DB インスタンスを設定する

DB インスタンスが SSH 接続を受け入れるには、次の手順を実行します。

- DB インスタンスのセキュリティグループが TCP のポート 22 でインバウンド接続を許可していることを確認します。

  DB インスタンスのセキュリティグループを設定する方法については、「セキュリティグループによるアクセス制御」を参照してください。
- SSH トンネリングを使用する予定がない場合は、DB インスタンスがパブリックサブネットにあり、パブリックにアクセス可能であることを確認してください。

  コンソールでは、データベース詳細ページの [接続とセキュリティ] タブで関連するフィールドが一般公開されています。CLI で設定を確認するには、次のコマンドを実行します。

  ```
  aws rds describe-db-instances \
  --query 'DBInstances[*].
  {DBInstanceIdentifier:DBInstanceIdentifier,PubliclyAccessible:PubliclyAccessible}' \
  --output table
  ```

  DB インスタンスのアクセシビリティ設定を変更するには、「Amazon RDS DB インスタンスを変更する」を参照してください。

ステップ 2: SSH シークレットキーと EC2 インスタンス ID を取得する

SSH を使用して、DB インスタンスに接続するには、インスタンスに関連付けられている SSH キーペアが必要です。RDS Custom は、命名規則 `do-not-delete-rds-custom-ssh-privatekey-`*`resource_id`*`-`*`uuid`* または `rds-custom!oracle-do-not-`

delete-*resource_id*-*uuid*-ssh-privatekey を使用して、ユーザーに代わって SSH キーペアを作成します。AWS Secrets Manager は、SSH プライベートキーをシークレットとして保存します。

AWS マネジメントコンソール または AWS CLI を使用して SSH シークレットキーを取得します。インスタンスにパブリック DNS があり、SSH トンネリングを使用する予定がない場合は、DNS 名も取得してください。公開接続の DNS 名を指定します。

コンソール

シークレット SSH キーを取得するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「データベース」 を選択し、停止する RDS Custom DB インスタンスを選択します。
3. [設定] を選択します。
4. リソース ID値に注意してください。例えば、DB インスタンスリソース ID は db-ABCDEFGHIJKLMNOPQRS0123456 になります。
5. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
6. ナビゲーションペインで、[インスタンス] を選択してください。
7. EC2 インスタンスの名前を見つけ、それに関連付けられているインスタンス ID を選択します。例えば、EC2 インスタンス ID はi-abcdefghijklm01234です。
8. 詳細で、キーペア名を見つけます。ペア名には DB インスタンスリソース ID が含まれます。例えば、ペア名は do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c または rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey のようになります。
9. EC2 インスタンスがパブリックの場合は、パブリック IPv4 DNS を書き留めておきます。この例では、公開ドメインネームシステム (DNS) アドレスはec2-12-345-678-901.us-east-2.compute.amazonaws.comのようになります。
10. AWS Secrets Managerコンソールを https://console.aws.amazon.com/secretsmanager/で開きます。
11. キーペアと同じ名前のシークレットを選択します。
12. [シークレットの値を取得する] を選択します。
13. SSH プライベートキーをテキストファイルにコピーし、.pem 拡張子を付けてファイルを保存します。例えば、ファイルを /tmp/do-not-delete-rds-custom-ssh-privatekey-db-

ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem または /tmp/rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem として保存します。

AWS CLI

SSH プライベートキーを取得して.pem ファイルに保存するには、AWS CLI を使用できます。

1. aws rds describe-db-instances を使用して RDS Custom DB インスタンスの DB リソース ID を見つけます。

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

次のサンプル出力は、RDS Custom インスタンスのリソース ID を示しています。プレフィックスはdb-です。

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

2. aws ec2 describe-instances を使用して DB インスタンスの EC2 インスタンス ID を見つけます。次の例ではリソース ID に db-ABCDEFGHIJKLMNOPQRS0123456 を使用しています。

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

次のサンプル出力は、EC2 インスタンス ID を示しています。

```
i-abcdefghijklm01234
```

3. キー名を見つけるには、EC2 インスタンス ID を指定します。次の例では、EC2 インスタンス i-0bdc4219e66944afa について説明しています。

```
aws ec2 describe-instances \
    --instance-ids i-0bdc4219e66944afa \
    --output text \
    --query 'Reservations[*].Instances[*].KeyName'
```

次の出力例は、命名形式 do-not-delete-rds-custom-ssh-privatekey-*resource_id*-*uuid* または rds-custom!oracle-do-not-delete-*resource_id*-*uuid*-ssh-privatekey を使用するキー名を示しています。

```
do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c
rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey
```

4. プライベートキーは、aws secretsmanager を使用してキーの名前を付けた .pem ファイルに保存します。

   次の例では、do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c キーを /tmp ディレクトリのファイルに保存します。

   ```
   aws secretsmanager get-secret-value \
       --secret-id do-not-delete-rds-custom-ssh-privatekey-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c \
       --query SecretString \
       --output text >/tmp/do-not-delete-rds-custom-ssh-privatekey-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem
   ```

   次の例では、rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey キーを /tmp ディレクトリのファイルに保存します。

   ```
   aws secretsmanager get-secret-value \
       --secret-id rds-custom!oracle-do-not-delete-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey \
       --query SecretString \
       --output text >/tmp/rds-custom!oracle-do-not-delete-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem
   ```

ステップ 3: SSH ユーティリティを使用して EC2 インスタンスに接続します

接続方法は、プライベート DB インスタンスに接続するか、パブリックインスタンスに接続するかによって異なります。プライベート接続では、AWS Systems Manager による SSH トンネリングを設定する必要があります。

SSH ユーティリティを使用して EC2 インスタンスに接続するには

1. プライベート接続の場合は、コマンドを AWS Systems Manager Session Manager にプロキシするように SSH 設定ファイルを変更してください。公開接続の場合は、ステップ 2 に進みます。

   ~/.ssh/config に次の行を追加します。これらの行は、前が i- または mi- で始まるホストの SSH コマンドをプロキシします。

   ```
   Host i-* mi-*
       ProxyCommand sh -c "aws ssm start-session --target %h --document-name AWS-
   StartSSHSession --parameters 'portNumber=%p'"
   ```

2. .pem ファイルを含むディレクトリに移動します。chmodを使用して、許可を400に設定します。

   次の例では、/tmp ディレクトリに変更し、.pem ファイル do-not-delete-rds-custom-ssh-privatekey-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem のアクセス許可を設定します。

   ```
   cd /tmp
   chmod 400 do-not-delete-rds-custom-ssh-privatekey-db-
   ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem
   ```

   次の例では、/tmp ディレクトリに変更し、.pem ファイル rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-privatekey.pem のアクセス許可を設定します。

   ```
   cd /tmp
   chmod 400 rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
   privatekey.pem
   ```

3. .pem ファイルと、パブリック DNS 名 (パブリック接続用) または EC2 インスタンス ID (プライベート接続用) を指定して ssh ユーティリティを実行します。ユーザー ec2-user としてログインします。

   次の例では、DNS 名 ec2-12-345-678-901.us-east-2.compute.amazonaws.com を使用してパブリックインスタンスに接続します。

   ```
   # .pem file using naming prefix do-not-delete
   ```

```
ssh -i \
  "do-not-delete-rds-custom-ssh-privatekey-db-
ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem" \
  ec2-user@ec2-12-345-678-901.us-east-2.compute.amazonaws.com

# .pem file using naming prefix rds-custom!oracle-do-not-delete
ssh -i \
  "rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
privatekey.pem" \
  ec2-user@ec2-12-345-678-901.us-east-2.compute.amazonaws.com
```

次の例では、EC2 インスタンス ID `i-0bdc4219e66944afa` を使用してプライベートインスタンスに接続します。

```
# .pem file using naming prefix do-not-delete
ssh -i \
  "do-not-delete-rds-custom-ssh-privatekey-db-
ABCDEFGHIJKLMNOPQRS0123456-0d726c.pem" \
  ec2-user@i-0bdc4219e66944afa

# .pem file using naming prefix rds-custom!oracle-do-not-delete
ssh -i \
  "rds-custom!oracle-do-not-delete-db-ABCDEFGHIJKLMNOPQRS0123456-0d726c-ssh-
privatekey.pem" \
  ec2-user@i-0bdc4219e66944afa
```

RDS Custom for Oracle データベースに SYS としてログインします。

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom DB インスタンスを作成すると、SYSDBA 権限が付与される、ユーザー SYS として Oracle データベースにログインできます。次のログイン方法があります。

- Secrets Manager から SYS パスワードを取得し、このパスワードを SQL クライアントで指定します。
- OS 認証を使用してデータベースにログインします。この場合、パスワードは必要ありません。

RDS Custom for Oracle データベースの SYS パスワードの検索

SYS または SYSTEM として、または API コールでマスターユーザー名を指定することにより、Oracle データベースにログインできます。SYS と SYSTEM のパスワードは Secrets Manager に保存されます。

シークレットは、命名形式 `do-not-delete-rds-custom-`*`resource_id`*`-`*`uuid`* または `rds-custom!oracle-do-not-delete-`*`resource_id`*`-`*`uuid`* を使用します。パスワードは、AWS マネジメントコンソール で検索できます。

コンソール

Secrets Manager でデータベースの SYS パスワードを確認するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. RDS コンソールで、以下の手順を実行します。
   a. ナビゲーションペインで、[Databases] (データベース) を選択します。
   b. RDS Custom for Oracle DB インスタンスの名前を選択します。
   c. [設定] を選択します。
   d. [リソース ID] の下の値をコピーします。例えば、リソース ID が db-ABC12CDE3FGH4I5JKLMNO6PQR7 とします。
3. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
4. Secrets Manager コンソールで、以下の手順を実行します。
   a. 左側のナビゲーションペインで [サーバー] を選択します。
   b. ステップ 2.d でコピーしたリソース ID でシークレットをフィルタリングします。
   c. do-not-delete-rds-custom-***`resource_id`*****`-uuid`*** または rds-custom!oracle-do-not-delete-***`resource_id`*****`-uuid`*** という名前形式を使用するシークレットを選択します。*`resource_id`* は、ステップ 2.d でコピーしたリソース ID です。

例えば、リソース ID が db-ABC12CDE3FGH4I5JKLMNO6PQR7 で、UUID が 1234ab の場合、シークレットの名前は do-not-delete-rds-custom-db-ABC12CDE3FGH4I5JKLMNO6PQR7-1234ab または rds-custom!oracle-do-not-delete-db-ABC12CDE3FGH4I5JKLMNO6PQR7-1234ab です。

d. [シークレットの値] で、[シークレットの値を取得する] を選択します。

e. [キー/値] で、[パスワード] の値をコピーします。

5. DB インスタンスに SQL*Plus をインストールし、データベースに SYS としてログインします。詳細については、「ステップ 3: SQL クライアントを Oracle DB インスタンスに接続する」を参照してください。

OS 認証を使用して RDS Custom for Oracle データベースにログインする

OS ユーザー `rdsdb` は Oracle データベースのバイナリを所有しています。`rdsdb` ユーザーに切り替えて、パスワードなしで RDS Custom for Oracle データベースにログインできます。

1. AWS Systems Manager を使って DB インスタンスに接続します。詳細については、「Session Manager を使用して RDS Custom DB インスタンスに接続する」を参照してください。
2. `rdsdb` ユーザーに切り替えます。

```
sudo su - rdsdb
```

3. OS 認証を使用してデータベースにログインします。`sqlplus / as sysdba` または `sql` エイリアスを使用できます。

```
$ sqlplus / as sysdba

SQL*Plus: Release 21.0.0.0.0 - Production on Wed Apr 12 20:11:08 2023
Version 21.9.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.10.0.0.0
```

または、`sql` エイリアスを使用することもできます。

```
$ sql
```

Amazon RDS Custom for Oracle DB インスタンスの管理

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS Customは、Amazon RDS DB インスタンスの通常の管理タスクのサブセットをサポートしています。以下では、AWS マネジメントコンソールおよびAWS CLIを使用してサポートされている RDS Custom for Oracle の管理タスクについて説明します。

トピック

- RDS Custom for Oracle でのコンテナデータベース (CDB) の使用
- RDS Custom for Oracle のハイアベイラビリティー機能の使用
- RDS Custom 環境のカスタマイズ
- RDS Custom for Oracle DB インスタンスを変更する
- RDS Custom for Oracle DB インスタンスの文字セットを変更する
- RDS Custom for Oracle での NLS_LANG 値の設定
- 透過的なデータの暗号化サポート
- RDS Custom for Oracle リソースのタグ付け
- RDS Custom for Oracle DB インスタンスを削除する

RDS Custom for Oracle でのコンテナデータベース (CDB) の使用

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは

RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle DB インスタンスは、Oracle マルチテナントアーキテクチャ (`custom-oracle-ee-cdb` や `custom-oracle-se2-cdb` エンジンタイプ) または従来の非 CDB アーキテクチャ (`custom-oracle-ee` や `custom-oracle-se2` エンジンタイプ) を使用して作成できます。コンテナデータベース (CDB) を作成すると、そのデータベースには 1 つのプラグ可能なデータベース (PDB) と 1 つの PDB シードが含まれます。Oracle SQL を使用して、手動で追加の PDB を作成できます。

PDB と CDB の名前

RDS Custom for Oracle DB インスタンスを作成するとき、最初の PDB 名前を指定します。デフォルトでは、最初の PDB には `ORCL` という名前が付けられます。別の名前を選択できます。

デフォルトでは、CDB の名前は `RDSCDB` です。別の名前を選択できます。CDB 名は、CDB を管理するメモリとプロセスを一意に識別する Oracle システム識別子 (SID) の名前でもあります。Oracle SID の詳細は、「Oracle データベースの概念」の「Oracle システム識別子 (SID)」を参照してください。

Amazon RDS API を使用して既存の PDB の名前を変更することはできません。また、`modify-db-instance` コマンドを使用して CDB の名前を変更することはできません。

PDB 管理

RDS Custom for Oracle の責任共有モデルでは、PDB の管理と追加の PDB の作成はお客様の責任となります。RDS Custom では PDB の数を制限しません。CDB ルートに接続して SQL ステートメントを実行することで、PDB を手動で作成、変更、削除できます。Amazon EBS データボリュームに PDB を作成して、DB インスタンスがサポート範囲外に出ないようにします。

CDB または PDB を変更するには、以下のステップを完了する必要があります。

1. RDS Custom アクションへの干渉を防ぐため、オートメーションを一時停止します。
2. CDB または PDB を変更します。
3. 変更した PDB をすべてバックアップします。
4. RDS Custom オートメーションを再開します。

CDB ルートの自動復旧

RDS Custom は、非 CDB ルートを開いたままにするのと同じ方法で CDB ルートを開いたままにします。CDB ルートの状態が変化した場合、モニタリングとリカバリの自動化により CDB ルートを目的の状態に回復しようとします。RDS イベント通知は、非 CDB アーキテクチャと同様に、ルート CDB がシャットダウン (RDS-EVENT-0004) または再起動 (RDS-EVENT-0006) した際に受け取ります。RDS Custom は、DB インスタンスの起動時にすべての PDB を `READ WRITE` モードで開こうとします。一部の PDB を開くことができない場合、RDS Custom では `tenant database shutdown` のイベントを発行します。

RDS Custom for Oracle のハイアベイラビリティー機能の使用

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle は、マルチ AZ 配置を通じて組み込みの高可用性を提供します。または、ユースケースに基づいて、Oracle Data Guard をカスタマーマネージドオプションとして使用できます。

マルチ AZ 配置 (フルマネージド)

RDS Custom for Oracle のマルチ AZ 配置では、Amazon RDS は、異なるアベイラビリティーゾーン (AZ) で同期スタンバイレプリカを自動的にプロビジョンおよび維持します。プライマリ DB インスタンスは、同期的に AZ 間でスタンバイレプリカにレプリケートされ、データの冗長性が提供されます。マルチ AZ 配置は、Enterprise Edition と Standard Edition 2 の両方でサポートされています。詳細については、「the section called “RDS Custom for Oracle のマルチ AZ 配置の管理”」を参照してください。

Oracle Data Guard (カスタマーマネージド)

または、RDS Custom for Oracle DB インスタンス間でデータをレプリケートするように Oracle Data Guard を手動で設定することで、高可用性を実現することもできます。プライマリ DB インスタンスは、データをスタンバイインスタンスに自動的に同期します。Oracle Data Guard は Enterprise Edition でのみサポートされています。

ハイアベイラビリティー環境は、次の方法で設定できます。

- 異なるアベイラビリティーゾーン (AZ) のスタンバイインスタンスを、AZ 障害への耐性を持つように設定します。
- スタンバイデータベースをマウントモードまたは読み取り専用モードにします。読み取り専用モードには、Oracle Active Data Guard ライセンスが必要です。
- プライマリデータベースからスタンバイデータベースにフェイルオーバーまたはスイッチオーバーしても、データが失われることはありません。
- オンプレミスインスタンスのハイアベイラビリティーを設定し、RDS Custom for Oracle スタンバイデータベースにフェイルオーバーまたはスイッチオーバーしてデータを移行します。

Oracle Data Guard の高可用性の設定方法については、AWS ブログの「リードレプリカを使用した RDS Custom for Oracle での高可用性の有効化」を参照してください。以下のタスクを実行できます。

- バーチャルプライベートネットワーク (VPN) トンネルを使用して、ハイアベイラビリティーインスタンスの転送中のデータを暗号化します。転送中の暗号化は RDS Custom for Oracle によって自動的に設定されません。
- Oracle 高速フェイルオーバーオブザーバー (Fast-Failover Observer、FSFO) を設定して、ハイアベイラビリティーインスタンスをモニタリングします。
- 必要な条件が満たされると、オブザーバーが自動フェイルオーバーを実行できるようにします。

RDS Custom 環境のカスタマイズ

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle は、自動化を一時停止せずに DB インスタンス環境をカスタマイズできる組み込み機能を備えています。例えば、RDS API を使用して環境を次のようにカスタマイズできます。

- DB スナップショットを作成して復元し、クローン環境を作成します。
- リードレプリカを作成します。
- ストレージ設定を変更します。
- CEV を変更してリリースアップデートを適用します

文字セットの変更などの一部のカスタマイズでは、RDS API を使用できません。このような場合は、Amazon EC2 インスタンスにルートユーザーとしてアクセスするか、Oracle データベースに SYSDBA として口グインして、手動で環境を変更する必要があります。

インスタンスを手動でカスタマイズするには、RDS Custom のオートメーションを一時停止して再開する必要があります。この一時停止により、カスタマイズが RDS Custom オートメーションに干渉しないようにします。こうすることで、サポート境界が壊れてしまい、根本的な問題を修正するまでインスタンスが `unsupported-configuration` 状態になることを回避できます。一時停止と再開は、RDS Custom for Oracle DB インスタンス変更の際サポートされている唯一のオートメーションタスクです。

RDS Custom 環境をカスタマイズするための一般的な手順

RDS Custom DB インスタンスをカスタマイズするには、以下の手順を実行します。

1. コンソールまたは CLI を使用して、指定した期間 RDS Custom のオートメーションを一時停止します。
2. 基盤となる Amazon EC2 インスタンスを特定します。
3. SSH キーまたは AWS Systems Manager を使用して基盤となる Amazon EC2 インスタンスに接続します。
4. データベースまたはオペレーティングシステムレイヤーで現在の構成設定を確認します。

   初期設定と変更後の設定を比較することで、変更を検証できます。カスタマイズの種類に応じて、OS ツールまたはデータベースクエリを使用してください。
5. 必要に応じて RDS Custom for Oracle DB インスタンスをカスタマイズします。
6. 必要に応じて、インスタンスまたはデータベースを再起動します。

> **Note**
>
> オンプレミスの Oracle CDB では、組み込みコマンドを使用するか、起動トリガーの後に、指定した PDB のオープンモードを保存できます。このメカニズムによって、CDB の再起動時に PDB が指定された状態になります。CDB を開くと、RDS Custom オート

メーションではユーザー指定の保存状態をすべて破棄して、すべての PDB を開こうとします。RDS Custom がすべての PDB を開くことができない場合は、`The following PDBs failed to open: list-of-PDBs` のイベントが発生します。

7. 新しい構成設定を以前の設定と比較して確認します。
8. 次のいずれかの方法で RDS Custom のオートメーションを再開します。
   - オートメーションをマニュアルで再開します。
   - 一時停止期間が終了するのを待ちます。この場合、RDS Custom はモニタリングとインスタンスの回復を自動的に再開します。
9. RDS Custom のオートメーションフレームワークの検証

   前述の手順を正しく実行すると、RDS Custom は自動バックアップを開始します。コンソールのインスタンスのステータスには [利用可能] と表示されます。

ベストプラクティスと詳細な手順については、AWS ブログ記事「Make configuration changes to an Amazon RDS Custom for Oracle instance: Part 1」(Amazon RDS Custom for Oracle インスタンスの設定を変更する: パート 1) と「Recreate an Amazon RDS Custom for Oracle database: Part 2」(Amazon RDS Custom for Oracle データベースを再作成する: パート 2) を参照してください。

RDS Custom DB インスタンスの一時停止と再開

コンソールまたは CLI を使用して、DB インスタンスのオートメーションを一時停止および再開できます。

コンソール

RDS Custom オートメーションを一時停止または再開するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択し、変更する RDS Custom DB インスタンスを選択します。
3. Modify を選択します。Modify DB instance ページが表示されます。
4. 「RDS Customオートメーションモード」では、以下のいずれかのオプションを選択します。
   - 一時停止中は、RDS Custom DB インスタンスのモニタリングとインスタンスのリカバリを一時停止します。目的の一時停止時間 (分単位) を「オートメーションモードの継続期間」に入力します。最小値は 60 分です (デフォルト)。最大値は 1,440 分です。

- フルオートメーションは、オートメーションを再開します。

5. 「Continue」 選択して、変更の概要をチェックします。

   RDS Custom がすぐに変更を適用することを示すメッセージが表示されます。

6. 変更が正しければ、Modify DB Instance (DB インスタンスを変更) を選択します。または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

   RDS コンソールに、変更の詳細が表示されます。オートメーションを一時停止すると、 RDS Custom DB インスタンスのステータスがオートメーションの一時停止を示します。

7. (オプション) ナビゲーションペインで 「データベース」 を、それから RDS カスタム DB インスタンスを選択します。

   概要ペインで、RDS Custom オートメーションモードはオートメーションのステータスを示します。オートメーションが一時停止されている場合、値は一時停止です。オートメーションは ***num*** 分後に再開されます。

AWS CLI

RDS Custom オートメーションを一時停止または再開するには、`modify-db-instance` AWS CLI コマンドを使用します。必須パラメータ `--db-instance-identifier` を使用して DB インスタンスを指定します。次のパラメータを使用して、オートメーションモードを制御します。

- `--automation-mode`は、DB インスタンスの一時停止状態を指定します。有効な値は`all-paused`で、オートメーションを一時停止し、`full`は再開します。
- `--resume-full-automation-mode-minutes`は一時停止期間を指定します。デフォルト値は 60 分です。

> **Note**
>
> `--no-apply-immediately`または`--apply-immediately`を指定したかどうかにかかわらず、RDS Custom は、変更をできるだけ早く非同期的に適用します。

コマンド応答では、`ResumeFullAutomationModeTime`は UTC タイムスタンプとして再開時刻を示します。オートメーションモードが`all-paused`の場合、`modify-db-instance`を使用してオートメーションモードを再開するか、一時停止期間を延長できます。その他の`modify-db-instance`オプションはサポートされていません。

次の例では、my-custom-instance のオートメーションを 90 分間一時停止します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 90
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 90
```

次の例では、一時停止期間をさらに 30 分間延長します。ResumeFullAutomationModeTime で示された元の時刻に 30 分が加算されます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 30
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 30
```

次の例では、my-custom-instance のフルオートメーションを再開します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode full \
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode full
```

以下の部分出力例では、保留されている AutomationMode の値は full です。

```
{
    "DBInstance": {
        "PubliclyAccessible": true,
        "MasterUsername": "admin",
        "MonitoringInterval": 0,
        "LicenseModel": "bring-your-own-license",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "0123456789abcdefg"
            }
        ],
        "InstanceCreateTime": "2020-11-07T19:50:06.193Z",
        "CopyTagsToSnapshot": false,
        "OptionGroupMemberships": [
            {
                "Status": "in-sync",
                "OptionGroupName": "default:custom-oracle-ee-19"
            }
        ],
        "PendingModifiedValues": {
            "AutomationMode": "full"
        },
        "Engine": "custom-oracle-ee",
        "MultiAZ": false,
        "DBSecurityGroups": [],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.custom-oracle-ee-19",
                "ParameterApplyStatus": "in-sync"
```

```
            }
        ],
        ...
        "ReadReplicaDBInstanceIdentifiers": [],
        "AllocatedStorage": 250,
        "DBInstanceArn": "arn:aws:rds:us-west-2:012345678912:db:my-custom-instance",
        "BackupRetentionPeriod": 3,
        "DBName": "ORCL",
        "PreferredMaintenanceWindow": "fri:10:56-fri:11:26",
        "Endpoint": {
            "HostedZoneId": "ABCDEFGHIJKLMNO",
            "Port": 8200,
            "Address": "my-custom-instance.abcdefghijk.us-west-2.rds.amazonaws.com"
        },
        "DBInstanceStatus": "automation-paused",
        "IAMDatabaseAuthenticationEnabled": false,
        "AutomationMode": "all-paused",
        "EngineVersion": "19.my_cev1",
        "DeletionProtection": false,
        "AvailabilityZone": "us-west-2a",
        "DomainMemberships": [],
        "StorageType": "gp2",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVW",
        "ResumeFullAutomationModeTime": "2020-11-07T20:56:50.565Z",
        "KmsKeyId": "arn:aws:kms:us-west-2:012345678912:key/
aa111a11-111a-11a1-1a11-1111a11a1a1a",
        "StorageEncrypted": false,
        "AssociatedRoles": [],
        "DBInstanceClass": "db.m5.xlarge",
        "DbInstancePort": 0,
        "DBInstanceIdentifier": "my-custom-instance",
        "TagList": []
    }
```


RDS Custom for Oracle DB インスタンスを変更する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle DB インスタンスの変更は、Amazon RDS DB インスタンスの変更と似ています。以下のように設定を変更することができます。

- DB インスタンスクラス
- ストレージの割り当てとタイプ
- バックアップの保存期間
- 削除保護
- オプショングループ
- CEV (「RDS Custom for Oracle DB インスタンスのアップグレード」を参照)
- ポート

トピック

- DB インスタンスのストレージの変更時の要件と制限事項
- DB インスタンスクラスの変更時の要件と制限事項
- インスタンスクラスを変更したときに RDS Custom によって DB インスタンスを作成する方法
- RDS Custom for Oracle DB インスタンスを変更する

DB インスタンスのストレージの変更時の要件と制限事項

RDS Custom for Oracle DB インスタンスのストレージの変更には、次の要件と制限事項を考慮してください。

- RDS Custom for Oracleに割り当てられる最小ストレージは 40 GiB で、最大値は 64 TiB です。
- Amazon RDS と同様に、割り当てられたストレージを減らすことはできません。これは Amazon EBS ボリュームの制限事項です。
- RDS CCustom DB インスタンスでは、ストレージのオートスケーリングはサポートされていません。
- RDS CuCustom DB インスタンスにマニュアルでアタッチするストレージボリュームは、サポートペリメーター外にあります。

  詳細については、「RDS Custom サポート範囲」を参照してください。
- RDS Custom では、磁気 (スタンダード) の Amazon EBS ストレージはサポートされていません。io1、io2、gp2、または gp3 SSD ストレージタイプのみを選択できます。

Amazon EBS ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。ストレージの変更に関する一般的な情報については、「Amazon RDS DB インスタンスのストレージを使用する」を参照してください。

DB インスタンスクラスの変更時の要件と制限事項

RDS Custom for Oracle DB インスタンスのインスタンスクラスの変更には、次の要件と制限事項を考慮してください。

- DB インスタンスは `available` の状態である必要があります。
- DB インスタンスには、ルートボリューム、データボリューム、バイナリボリュームに最低 100 MiB の空き容量が必要です。
- デフォルトの elastic network interface (ENI) を使用する場合、RDS Custom for Oracle DB インスタンスに割り当てることができる Elastic IP (EIP) は 1 つだけです。DB インスタンスに複数の ENI をアタッチすると、変更オペレーションは失敗します。
- RDS Custom for Oracle タグがすべて存在している必要があります。
- RDS Custom for Oracle レプリケーションを使用する場合は、次の要件と制限事項に注意してください。
  - プライマリ DB インスタンスとリードレプリカの場合、一度に 1 つの DB インスタンスに限りインスタンスクラスを変更できます。
  - RDS Custom for Oracle DB インスタンスにオンプレミスのプライマリまたはレプリカデータベースがある場合は、必ず、変更完了後にオンプレミス DB インスタンスのプライベート IP アドレスを手動で更新してください。このアクションは Oracle DataGuard の機能を維持するために必要です。RDS Custom for Oracle では、変更が成功するとイベントを発行します。
  - プライマリまたはリードレプリカ DB インスタンスに FSFO (Fast-Start Failover) が設定されている場合、RDS Custom for Oracle DB インスタンスクラスを変更することはできません。

インスタンスクラスを変更したときに RDS Custom によって DB インスタンスを作成する方法

インスタンスクラスを変更したときに RDS Custom によって DB インスタンスを作成するには、次の手順に従います。

- Amazon EC2 インスタンスを作成します。
- 最新の DB スナップショットからルートボリュームを作成します。RDS Custom for Oracle は、最新の DB スナップショットの後にルートボリュームに追加された情報を保持しません。
- Amazon CloudWatch アラームを作成します。

- 元のキーペアを削除した場合、Amazon EC2 SSH のキーペアを作成します。それ以外の場合、RDS Custom for Oracle は元のキーペアを保持します。
- 変更を開始したときに DB インスタンスにアタッチされたタグを使用して、新しいリソースを作成します。RDS Custom では、新しいリソースが基礎となるリソースに直接アタッチされている場合、そのリソースにタグを転送しません。
- 最新の変更を加えたバイナリボリュームとデータボリュームを、新しい DB インスタンスに転送します。
- Elastic IP アドレス (EIP) を転送します。DB インスタンスがパブリックにアクセス可能な場合、RDS Custom では EIP を転送する前に、新しい DB インスタンスにパブリック IP アドレスを一時的にアタッチします。DB インスタンスがパブリックにアクセスできない場合、RDS Custom ではパブリック IP アドレス を作成しません。

RDS Custom for Oracle DB インスタンスを変更する

DB インスタンスクラスまたはストレージの変更は、コンソール、AWS CLI、または RDS API を使用して変更できます。

コンソール

RDS Custom Oracle DB インスタンスを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. (オプション) [インスタンスの設定] で、[DB インスタンスクラス] の値を選択します。サポートされているクラスについては、「RDS Custom for Oracle での DB インスタンスクラスのサポート」を参照してください。
6. (オプション) [ストレージ] で、必要に応じて次の変更を加えます。
   a. [ストレージ割り当て] に新しい値を入力します。現在値よりも大きい値かつ40 GiB ~ 64 TiB である必要があります。
   b. [ストレージタイプ] の値を [汎用 SSD (gp2)]、[汎用 SSD (gp3)]、[プロビジョンド IOPS (io1)]、または [プロビジョンド IOPS (io2)] に変更します。

c. [汎用 SSD (gp2)] 以外のストレージタイプを指定した場合は、[プロビジョンド IOPS] 値を変更できます。

7. (オプション) [追加設定] で、必要に応じて次の変更を加えます。

   - [オプショングループ] で、新しいオプショングループを選択します。詳細については、「RDS Custom for Oracle のオプショングループ使用する」を参照してください。

8. [続行] をクリックします。
9. すぐに適用または次の定期メンテナンスウィンドウ中に適用を選択します。
10. [DB インスタンスを変更] を選択します。

AWS CLI

RDS Custom for Oracle DB インスタンスのストレージを変更するには、modify-db-instance AWS CLIコマンドを使用します。以下のパラメータを必要に応じて調整します。

- `--db-instance-class` – 新しいインスタンスクラス。サポートされているクラスについては、「RDS Custom for Oracle での DB インスタンスクラスのサポート」を参照してください。
- `--allocated-storage` - DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。現在値よりも大きい値かつ40 ~ 65,536 GiB である必要があります。
- `--storage-type` - ストレージタイプ: gp2、gp3、io1、または io2。
- `--iops` - io1、io2、または gp3 ストレージタイプを使用している場合、DB インスタンスのプロビジョンド IOPS。
- `--apply-immediately` - すぐにストレージの変更を適用するには、`--apply-immediately` を使用します。

  または `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

次の例では、`my-cfo-instance` の DB インスタンスクラスを `db.m5.16xlarge` に変更します。このコマンドはまた、ストレージサイズを `1024` (1 TiB) に、ストレージタイプを `io2` に、プロビジョンド IOPS を `3000` に、オプショングループを `cfo-ee-19-mt` に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
```

```
    --db-instance-identifier my-cfo-instance \
    --db-instance-class db.m5.16xlarge \
    --storage-type io2 \
    --iops 3000 \
    --allocated-storage 1024 \
    --option-group cfo-ee-19-mt \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-cfo-instance ^
    --db-instance-class db.m5.16xlarge ^
    --storage-type io2 ^
    --iops 3000 ^
    --allocated-storage 1024 ^
    --option-group cfo-ee-19-mt ^
    --apply-immediately
```

RDS Custom for Oracle DB インスタンスの文字セットを変更する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle は、デフォルトで文字セット US7ASCII に設定されています。言語またはマルチバイト文字の要件を満たすために、異なる文字セットを指定することもできます。RDS Custom for Oracle を使用する場合、オートメーションを一時停止し、データベースの文字セットを手動で変更します。

RDS Custom for Oracle DB インスタンスの文字セットを変更するには、次の要件があります。

- 文字を変更できるのは、アプリケーションデータを持たない空のデータベースまたはスターターデータベースを含む新たにプロビジョニングされた RDS Custom インスタンスのみです。他のすべてのシナリオでは、DMU (Unicode のデータベース移行アシスタント) を使用して文字セットを変更します。

- RDS for Oracle でサポートされている文字セットにのみ変更できます。詳細については、「サポートされている DB 文字セット」を参照してください。

RDS Custom Oracle DB インスタンスの文字セットを変更するには

1. RDS Custom オートメーションを一時停止します。詳細については、「RDS Custom DB インスタンスの一時停止と再開」を参照してください。
2. SYSDBA アクセス許可でユーザーとしてデータベースにログインします。
3. データベースを制限モードで再起動し、文字セットを変更してから、データベースを通常モードで再起動します。

   SQL クライアントで次のスクリプトを実行します。

   ```
   SHUTDOWN IMMEDIATE;
   STARTUP RESTRICT;
   ALTER DATABASE CHARACTER SET INTERNAL_CONVERT AL32UTF8;
   SHUTDOWN IMMEDIATE;
   STARTUP;
   SELECT VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER = 'NLS_CHARACTERSET';
   ```

   出力に正しい文字セットが表示されていることを確認します。

   ```
   VALUE
   --------
   AL32UTF8
   ```

4. RDS Custom オートメーションを再開します。詳細については、「RDS Custom DB インスタンスの一時停止と再開」を参照してください。

RDS Custom for Oracle での NLS_LANG 値の設定

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

ロケールとは、特定の言語と国に対する言語上の条件と文化的条件に対処するための一連の情報です。Oracle ソフトウェアのロケール動作を指定するには、クライアントホストで NLS_LANG 環境変数を設定します。この変数は、データベースセッションでクライアントアプリケーションが使用する言語、テリトリー、文字セットを設定します。

RDS Custom for Oracle では、NLS_LANG 変数で言語のみを設定できます。テリトリーと文字はデフォルトを使用します。この言語は、Oracle データベースのメッセージ、照合、曜日名、および月名に使用されます。サポートされている各言語には、アメリカ英語、フランス語、ドイツ語などの固有の名前が付けられています。言語が指定されない場合、値はデフォルトで [アメリカ英語] になります。

RDS Custom for Oracle データベースを作成したら、クライアントホストで NLS_LANG を英語以外の言語に設定できます。Oracle データベースでサポートされている言語のリストを確認するには、RDS Custom for Oracle データベースにログインし、次のクエリを実行します。

```
SELECT VALUE FROM V$NLS_VALID_VALUES WHERE PARAMETER='LANGUAGE' ORDER BY VALUE;
```

ホストコマンドラインで NLS_LANG を設定できます。次の例では、Linux で Z シェルを使用するクライアントアプリケーションの言語をドイツ語に設定します。

```
export NLS_LANG=German
```

アプリケーションは、起動時に NLS_LANG 値を読み取り、接続時にデータベースに伝達します。

詳細は、Oracle Database グローバリゼーションサポートガイドの「Choosing a Locale with the NLS_LANG Environment Variable」(NLS_LANG 環境変数によるロケールの選択) を参照してください。

透過的なデータの暗号化サポート

RDS Custom は、RDS Custom for Oracle DB インスタンスで透過的なデータの暗号化 (TDE) をサポートします。

ただし、RDS for Oracle の場合のように、カスタムオプショングループのオプションを使用して TDE を有効にすることはできません。TDE をマニュアルでオンにします。Oracle Transparent Data Encryption (透過的なデータの暗号化) については、Oracleのドキュメントで Transparent Data Encryption を使用した保存済みデータの保護を参照してください。

RDS Custom for Oracle リソースのタグ付け

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom リソースには Amazon RDS リソースと同様にタグ付けができますが、いくつかの重要な違いがあります。

- `AWSRDSCustom`RDS Custom オートメーションに必要なタグを作成または変更しないでください。そうすると、オートメーションが中断するおそれがあります。
- `Name` タグは、プレフィックス値 `do-not-delete-rds-custom` または `rds-custom!oracle-do-not-delete` で RDS カスタムリソースに追加されます。顧客から渡されたキーの値はすべて上書きされます。
- 作成時に RDS Custom DB インスタンスに追加されたタグは、他のすべての関連する RDS Custom リソースに伝搬されます。
- DB インスタンスの作成後に RDS Custom リソースにタグを追加しても、タグは伝搬されません。

リソースのタグ付けの詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

RDS Custom for Oracle DB インスタンスを削除する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

> **Warning**
>
> DB インスタンスの削除は永続的なアクションです。バックアップまたはスナップショットがない限り、削除後に DB インスタンスを復元することはできません。
> RDS Custom DB インスタンスを削除すると、AWS は基盤となる Amazon EC2 インスタンスと EBS ボリュームを自動的に削除します。Amazon RDS を介して DB インスタンスを削除する前に、Amazon EC2 インスタンスを手動で終了したり、EBS ボリュームを削除したりしないでください。これらのリソースを手動で削除すると、DB インスタンスの削除と最終的なスナップショットの作成が失敗し、復旧の可能性がなくなります。

DB インスタンスをRDS Custom DB instance で削除するには、以下を行う必要があります。

- DB インスタンスの名前を提供します。
- DB インスタンスの最終的なDBスナップショットを取得するオプションを解除します。
- 自動バックアップを保持するオプションを選択または解除します。

RDS Custom DB インスタンスはコンソールまたは CLI を使用して削除できます。DB インスタンスの削除に必要な時間は、バックアップ保持期間 (削除するバックアップの数)、削除するデータの量によって異なります。

コンソール

RDS Custom DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「データベース」 を選択し、削除したいRDS Custom DB インスタンスを選択します。RDS Custom DB インスタンスがロールインスタンス (RDS Custom)を表示します。
3. [アクション] で、[削除] を選択します。
4. 自動バックアップを保持するには、[自動バックアップの保持] を選択します。
5. ボックスに「**`delete me`**」と入力します。
6. [削除] を選択します。

AWS CLI

RDS Custom DB インスタンスを削除するには、delete-db-instance AWS CLIコマンドを使用します。必要なパラメータ--db-instance-identifierを使用して DB インスタンスを指定します。残りのパラメータは Amazon RDS DB インスタンスの場合と同じですが、次の例外があります。

- --skip-final-snapshot「」 は必須です。
- --no-skip-final-snapshot はサポートされていません。
- --final-db-snapshot-identifier はサポートされていません。

次の例では、my-custom-instanceという名前の RDS Custom DB インスタンスを削除し、自動バックアップを保持します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier my-custom-instance \
    --skip-final-snapshot \
    --no-delete-automated-backups
```

Windows の場合:

```
aws rds delete-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --skip-final-snapshot ^
    --no-delete-automated-backups
```

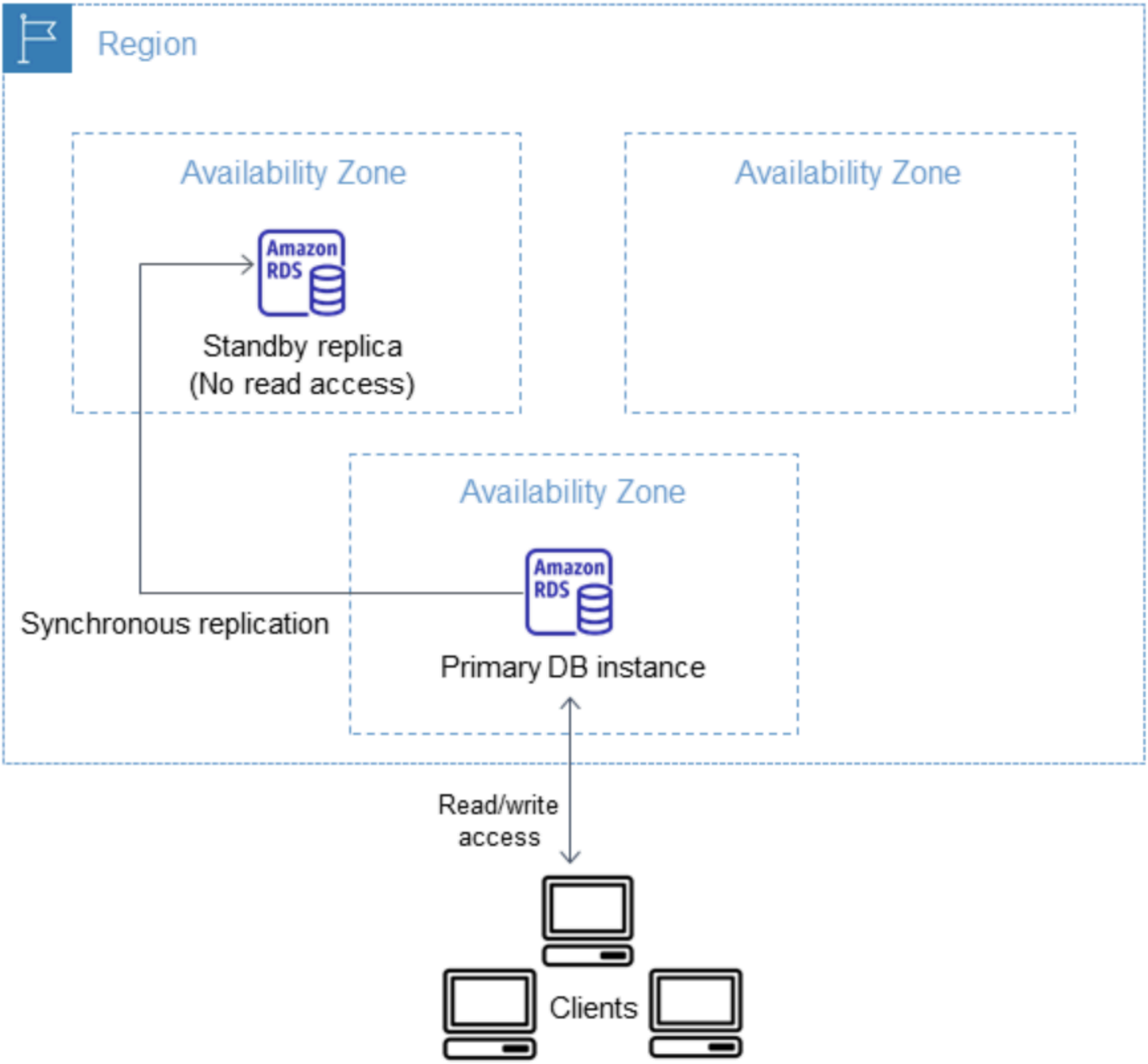
RDS Custom for Oracle のマルチ AZ 配置の管理

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle のマルチ AZ DB インスタンスのデプロイでは、Amazon RDS は、異なるアベイラビリティーゾーン (AZ) で同期スタンバイレプリカを自動的にプロビジョンおよび維持します。プライマリ DB インスタンスは、同期的にアベイラビリティーゾーン間でスタンバイレプリカにレプリケートされ、データの冗長性が提供されます。

マルチ AZ DB インスタンスのデプロイにより、計画的なシステムメンテナンス中の可用性が向上します。予定されたデータベースメンテナンスまたは予期しないサービス中断の発生時に、Amazon RDS は自動的に最新のスタンバイ DB インスタンスにフェイルオーバーします。データベースオペレーションを手動介入なしで速やかに再開できます。プライマリインスタンスおよびスタンバイインスタンスは、同じエンドポイントを使用します。このエンドポイントの物理的なネットワークアドレスは、フェイルオーバープロセスの一環としてスタンバイレプリカに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。

次の図は、RDS Custom for Oracle のマルチ AZ アーキテクチャを示しています。



Amazon RDS コンソールには、スタンバイレプリカ (セカンダリ AZ) のアベイラビリティーゾーンが表示されます。また、`describe-db-instances` CLI コマンドまたは `DescribeDBInstances` API オペレーションを使用してセカンダリ AZ を見つけることもできます。

RDS Custom for Oracle のマルチ AZ 配置のリージョンとバージョンの可用性

RDS Custom for Oracle のマルチ AZ 配置は、Enterprise Edition (EE) と Standard Edition 2 (SE2) の両方で次のリリースでサポートされています。

- Oracle Database 19c
- Oracle Database 12c Release 2 (12.2)
- Oracle Database 12c Release 1 (12.1)

> **Note**
>
> RDS Custom for Oracle のマルチ AZ 配置は、Oracle Database 18c ではサポートされていません。

RDS Custom for Oracle のマルチ AZ 配置は、RDS Custom for Oracle が利用できるすべてのリージョンで利用できます。RDS Custom for Oracle のマルチ AZ 配置の利用可能なリージョンの詳細については、「the section called “RDS Custom for Oracle でサポートされているリージョンと DB エンジン”」を参照してください。

RDS Custom for Oracle を使用したマルチ AZ 配置の制限事項

RDS Custom for Oracle のマルチ AZ 配置には、次のような制限事項があります。

- 2025 年 6 月 30 日より前に DB インスタンスを作成した場合、シングル AZ 配置からマルチ AZ 配置に変換することはできません。基盤となるカスタムエンジンバージョン (CEV) は、マルチ AZ のサポートがない古いサービス提供の AMI で構築されました。2025 年 6 月 30 日以降、サービスが提供する AMI で作成した CEV を使用して、データベースを新しい DB インスタンスに移行する必要があります。詳細については、「the section called “2025 年 6 月 30 日より前の CEV の移行”」を参照してください。
- クロスリージョンマルチ AZ 配置を作成することはできません。
- データベースの読み取りアクティビティを受け入れるように、セカンダリ DB インスタンスを設定することはできません。
- マルチ AZ 配置でカスタムエンジンバージョン (CEV) を使用する場合、スタンバイ DB インスタンスは同じ CEV を使用します。スタンバイ DB インスタンスは別の CEV を使用できません。
- マルチ AZ 配置インスタンスでリードレプリカを作成することはできません。また、マルチ AZ 配置を持つようにリードレプリカのプライマリインスタンスを変更することもできません。
- マルチ AZ 配置を使用する RDS Custom for Oracle DB インスタンスは、シングル AZ 配置と比較して書き込みとコミットのレイテンシーが増加する可能性があります。この増加は DB インスタンス間の同期データレプリケーションが原因で起こる可能性があります。AWS はアベイラビリティーゾーン間でのネットワーク接続レイテンシーが低くなるように設計されていますが、配置が

スタンバイレプリカにフェイルオーバーした場合はレイテンシーに変化が見られる可能性があります。

RDS Custom for Oracle でマルチ AZ 配置を作成する

マルチ AZ 配置を使用した RDS Custom for Oracle DB インスタンスを作成するには、the section called “RDS Custom for Oracle の環境設定” の手順に従って、前提条件を満たすように環境をセットアップします。

> **Important**
>
> セットアップを簡単にするために、ネットワーク設定手順に記載されている最新の CloudFormation テンプレートファイルを使用することをお勧めします。詳細については、「the section called “CloudFormation でデプロイする”」を参照してください。

Amazon RDS コンソールでデータベースインスタンスの作成中にマルチ AZ オプションを選択すると、マルチ AZ 配置を使用した RDS Custom for Oracle インスタンスを作成できます。または、AWS CLI 内の Amazon RDS `create-db-instance` コマンドで `--multi-az` パラメータを指定することもできます。

RDS Custom for Oracle のマルチ AZ 配置の前提条件

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle のマルチ AZ 配置は、RDS for Oracle のマルチ AZ 配置とは異なります。RDS for Oracle のマルチ AZ とは異なり、マルチ AZ DB インスタンスを作成する前に、RDS Custom for Oracle の前提条件を満たす必要があります。これは、RDS Custom が独自のアカウント内で実行され、アクセス許可が必要なためです。また、サービスが提供している最新の (マルチ AZ 配置をサポートする) AMI を使用して、CEV 上にマルチ AZ インスタンスを作成する必要があります。

前提条件を満たさないと、マルチ AZ DB インスタンスは実行に失敗するか、自動的にシングル AZ DB インスタンスに戻る可能性があります。前提条件の詳細については、「the section called “前提条件”」を参照してください。

RDS Custom for Oracle では、シングル AZ 配置からマルチ AZ 配置に変換するときに特定の前提条件が必要です。前提条件が不完全な場合、マルチ AZ セットアップが失敗します。手動セットアップまたはネットワークセットアップ手順に記載されている最新の CloudFormation テンプレートファイルを使用します。詳細については、「ステップ 3: RDS Custom for Oracle の CloudFormation テンプレートを抽出する」を参照してください。

前提条件を手動で完了するには、the section called “シングル AZ からマルチ AZ への変換” のステップに従い、次の点に注意してください。

- RDS Custom for Oracle DB インスタンスが、2025 年 6 月 30 日以降に作成された CEV を、最新のサービス提供 AMI と共に使用します。
- Amazon RDS セキュリティグループのインバウンドおよびアウトバウンドルールを更新して、ポート 1120 を許可します。
- 新しい Amazon SQS VPC エンドポイントを作成して、Amazon RDS Custom for Oracle DB インスタンスが Amazon SQS と通信できるようにします。
- インスタンスプロファイルロールの Amazon SQS アクセス許可を更新します。

> **Important**
>
> RDS Custom for Oracle オートメーションを一時停止せずにインスタンスにログインして、マルチ AZ プライマリ DB インスタンスを手動で再起動しないでください。

2025 年 6 月 30 日より前に作成された CEV を使用する DB インスタンスの移行手順

2025 年 6 月 30 日より前に RDS Custom for Oracle DB インスタンスを作成した場合、基盤となる CEV はマルチ AZ サポートのない古いサービス提供の AMI で構築されているため、マルチ AZ 配置に直接変更することはできません。2025 年 6 月 30 日以降に作成した CEV を使用して、データベースを新しいインスタンスに移行する必要があります。移行には、次の操作を行います。

1. サービス提供している最新の AMI (2025 年 6 月 30 日以降) を使用してソースから CEV を作成します。
2. 新しい CEV を使用して新しい DB インスタンスを起動します。

3. 以下のいずれかの方法を使用して、マルチ AZ 配置をサポートしていない既存の DB インスタンスから新しく作成されたインスタンスにデータベースを移行します。
   - Data Guard を用いて Oracle データベースを Amazon RDS Custom に物理コピーで移行する方法
   - RMAN 複製を用いて Oracle データベースを Amazon RDS Custom に物理コピーで移行する方法

RDS Custom for Oracle でのシングル AZ 配置からマルチ AZ 配置への変換

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

既存のマルチ AZ 対応の RDS Custom for Oracle インスタンスをシングル AZ 配置からマルチ AZ 配置に変換することができます。DB インスタンスを変更すると、Amazon RDS はいくつかのアクションを実行します。

- プライマリ DB インスタンスのスナップショットを取得します。
- スナップショットからスタンバイレプリカ用の新しいボリュームを作成します。これらのボリュームはバックグラウンドで初期化され、データが完全に初期化された後に最大のボリュームパフォーマンスが得られます。
- プライマリおよびスタンバイ DB インスタンス間の同期ブロックレベルレプリケーションをオンにします。

> **Important**
>
> アクティビティのピーク時には、RDS Custom for Oracle DB インスタンスを本稼働 DB インスタンス上でシングル AZ 配置からマルチ AZ 配置に変更しないことをお勧めします。

AWS では、シングル AZ からマルチ AZ への変換時にダウンタイムを回避するためにスナップショットを使用してスタンバイインスタンスを作成しますが、マルチ AZ への変換時および変換後に

パフォーマンスに影響が出ることがあります。この影響は、書き込みレイテンシーに敏感なワークロードにとって重大な可能性があります。この機能により、スナップショットから大量のボリュームをすばやく復元できますが、同期レプリケーションのため、I/O 操作のレイテンシーが著しく増加する可能性があります。このレイテンシーはデータベースのパフォーマンスに影響を与える可能性があります。

CloudFormation を使用してシングル AZ 配置からマルチ AZ 配置に変更するための前提条件の設定

ステップ 3: RDS Custom for Oracle の CloudFormation テンプレートを抽出する に従って VPC と IAM プロファイルを再度設定し、IAM プロファイルに SQS VPC エンドポイントと SQS アクセス許可を追加します。

シングル AZ 配置をマルチ AZ 配置に手動で変更するための前提条件を設定する

前提条件を手動で設定する場合は、以下のタスクを実行します。

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開きます。
2. [エンドポイント] を選択します。[Create Endpoint] (エンドポイントの作成) ページが表示されます。
3. [サービスカテゴリ] で、[AWS のサービス] を選択します。
4. [サービス] で SQS を検索します。
5. [VPC] で、RDS Custom for Oracle DB インスタンスがデプロイされている VPC を選択します。
6. [サブネット] で、RDS Custom for Oracle DB インスタンスがデプロイされているサブネットを選択します。
7. [セキュリティグループ] で、RDS Custom for Oracle DB インスタンスがデプロイされているセキュリティグループを選択します。
8. [ポリシー] で [カスタム] を選択します。
9. カスタムポリシーで、`AWS partition`、`Region`、`accountId`、および `IAM-Instance-role` を独自の値に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Condition": {
```

```
                "StringLike": {
                    "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
                }
            },
            "Action": [
                "SQS:SendMessage",
                "SQS:ReceiveMessage",
                "SQS:DeleteMessage",
                "SQS:GetQueueUrl"
            ],
            "Resource": "arn:aws:sqs:us-east-1:111122223333:do-not-delete-rds-
custom-*",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{IAM-Instance-role}"
            }
        }
    ]
}
```

Amazon SQS にアクセスするアクセス許可を持つ [インスタンスプロファイル] を更新します。AWS partition、Region、および accountId の値を独自の値に置き換えます。

```
{
    "Sid": "13",
    "Effect": "Allow",
    "Action": [
        "SQS:SendMessage",
        "SQS:ReceiveMessage",
        "SQS:DeleteMessage",
        "SQS:GetQueueUrl"
    ],
    "Resource": [
        {
            "Fn::Sub": "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:do-
not-delete-rds-custom-*"
        }
    ],
    "Condition": {
        "StringLike": {
            "aws:ResourceTag/AWSRDSCustom": "custom-oracle"
        }
```

```
    }
}
```

Amazon RDS セキュリティグループのインバウンドおよびアウトバウンドルールを更新して、ポート 1120 を許可します。

- [セキュリティグループ] で、RDS Custom for Oracle DB インスタンスがデプロイされているグループを選択します。
- [インバウンドルール] で、ソースグループからのポート `1120` を許可するカスタム TCP ルールを作成します。
- [アウトバウンドルール] で、カスタム TCP ルールを作成して、送信先グループにポート `1120` を許可します。

RDS コンソール、AWS CLI、または RDS API を使用して変更する

前提条件を満たしたら、Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、RDS Custom for Oracle DB インスタンスをシングル AZ 配置からマルチ AZ 配置に変更できます。

コンソール

既存の RDS Custom for Oracle シングル AZ 配置をマルチ AZ 配置に変更するには

1. AWS マネジメントコンソールにサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。[データベース] ペインが表示されます。
3. 変更する RDS Custom for Oracle DB インスタンスを選択します。
4. [アクション] で、[マルチ AZ 配置に変換] を選択します。
5. 変更をすぐに適用するには、[確認] ページで [すぐに適用] を選択します。このオプションを選択してもダウンタイムは発生しませんが、パフォーマンスに影響する可能性があります。または、次のメンテナンスウィンドウの間に更新を適用することもできます。詳細については、「the section called “スケジュールの変更”」を参照してください。
6. [確認] ページで、[マルチ AZ に変換] を選択します。

AWS CLI

AWS CLI を使用してマルチ AZ DB インスタンス配置に変換するには、modify-db-instance コマンドを呼び出して --multi-az オプションを設定します。DB インスタンス識別子と、変更する他のオプションの値を指定します。各オプションの詳細については、「」を参照してください。the section called “使用できる設定”

Example例

次のコードは、--multi-az オプションを含むことで mycustomdbinstance を変更します。変更は、--no-apply-immediately を使用して次のメンテナンスウィンドウ中に適用されます。今すぐ変更を適用するには、--apply-immediately を使用します。詳細については、「the section called “スケジュールの変更”」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --multi-az \
    [--no-apply-immediately | --apply-immediately]
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --multi-az ^
    [--no-apply-immediately | --apply-immediately]
```

RDS API

Amazon RDS API を使用してマルチ AZ DB インスタンス配置に変換するには、ModifyDBInstance オペレーションを呼び出し、MultiAZ パラメータを true に設定します。

RDS Custom for Oracle でのマルチ AZ 配置からシングル AZ 配置への変換

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは

RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

既存の RDS Custom for Oracle DB インスタンスをマルチ AZ 配置からシングル AZ 配置に変更することができます。

コンソール

RDS Custom for Oracle DB インスタンスをマルチ AZ 配置からシングル AZ 配置に変更するには

1. AWS マネジメントコンソールにサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。[データベース] ペインが表示されます。
3. 変更する RDS Custom for Oracle DB インスタンスを選択します。
4. [マルチ AZ 配置] で、[いいえ] を選択します。
5. 変更をすぐに適用するには、[確認] ページで [すぐに適用] を選択します。このオプションを選択してもダウンタイムは発生しませんが、パフォーマンスに影響する可能性があります。または、次のメンテナンスウィンドウの間に更新を適用することもできます。詳細については、「the section called “スケジュールの変更”」を参照してください。
6. [確認] ページで、[DB インスタンスの変更] を選択します。

AWS CLI

AWS CLI を使用してマルチ AZ 配置をシングル AZ 配置に変更するには、modify-db-instance コマンドを呼び出し、`--no-multi-az` オプションを含めます。DB インスタンス識別子と、変更する他のオプションの値を指定します。各オプションの詳細については、「」を参照してください。the section called “使用できる設定”

Example例

次のコードは、`--no-multi-az` オプションを含むことで `mycustomdbinstance` を変更します。変更は、`--no-apply-immediately` を使用して次のメンテナンスウィンドウ中に適用されます。今すぐ変更を適用するには、`--apply-immediately` を使用します。詳細については、「the section called “スケジュールの変更”」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --no-multi-az \
    [--no-apply-immediately | --apply-immediately]
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --no-multi-az ^
    [--no-apply-immediately | --apply-immediately]
```

RDS API

Amazon RDS API を使用してマルチ AZ 配置をシングル AZ 配置に変更するには、ModifyDBInstance オペレーションを呼び出し、`MultiAZ` パラメータを `false` に設定します。

RDS Custom for Oracle マルチ AZ 配置での OS のカスタマイズ

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle マルチ AZ 配置を使用すると、オペレーティングシステムをカスタマイズし、プライマリ EC2 インスタンスとスタンバイ EC2 インスタンスの両方にサードパーティーソフトウェアをインストールできます。Amazon RDS とは異なり、RDS Custom for Oracle はデータベース環境と基盤となるオペレーティングシステムへの管理アクセスを提供し、Oracle データベースに加えてモニタリングツール、セキュリティエージェント、またはカスタムアプリケーションをインストールできます。

マルチ AZ 配置で OS をカスタマイズする場合、プライマリインスタンスとスタンバイインスタンスの両方にカスタマイズが存在することを確認する責任があります。このアプローチにより、マルチ AZ フェイルオーバー中のアプリケーションの継続性が確保され、両方のインスタンスで一貫した機能が維持されます。

RDS Custom for Oracle マルチ AZ 配置で OS をカスタマイズするための要件

マルチ AZ 配置で OS をカスタマイズする前に、次の要件に注意してください。

- `/rdsdbdata` マウントポイントにのみサードパーティーソフトウェアをインストールします。データボリューム (`/rdsdbdata`) は、マルチ AZ 配置でレプリケートされる唯一のデータです。ルートボリューム (`/`) は OS パッチ適用中に置き換えられ、バイナリボリューム (`/rdsdbbin`) はデータベースパッチ適用中に置き換えられます。ルートボリュームとバイナリボリュームにインストールされているソフトウェアは、パッチ適用中に失われます。
- すべてのカスタマイズが AWS および Oracle ライセンスの利用規約に準拠していることを確認します。
- シングル AZ からマルチ AZ に変換する前に、`/etc/sysctl.conf` の HugePages 設定が正しく動作することを確認してください。

RDS Custom for Oracle マルチ AZ 配置での EC2 インスタンスの識別

マルチ AZ インスタンスをカスタマイズするときは、RDS Custom for Oracle デプロイのプライマリおよびスタンバイとして機能する Amazon EC2 インスタンスを特定します。

プライマリ EC2 インスタンスとスタンバイ EC2 インスタンスを識別するには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Databases] (データベース) を選択します。
3. マルチ AZ RDS Custom for Oracle DB インスタンスを選択します。
4. [設定] セクションで、[リソース ID] (形式: db-*nnnnnnn*) を書き留めます。
5. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
6. ナビゲーションペインで、[インスタンス] を選択してください。
7. 検索ボックスに、ステップ 4 のリソース ID を入力します。
8. 検索結果には、プライマリとセカンダリの 2 つのインスタンスが表示されます。アクティブな RDS Custom for Oracle データベースを持つインスタンスがプライマリです。

RDS Custom for Oracle マルチ AZ 配置を作成する前に OS をカスタマイズする

このシナリオでは、現在のデプロイはシングル AZ DB インスタンスです。OS をカスタマイズし、DB インスタンスをマルチ AZ 配置に変換できます。サードパーティーソフトウェアをインストールしていて、複数のボリュームでファイルを変更した場合は、この方法をお勧めします。

シングル AZ をマルチ AZ に変換する前に OS をカスタマイズするには

1. AWS Systems Manager Session Manager または SSH を使用して、シングル AZ デプロイの EC2 インスタンスに接続します。
2. 次のいずれかのカスタマイズを実行します。
   - データボリューム (/rdsdbdata) にサードパーティーソフトウェアをインストールします。
   - ルートボリューム (/) のファイルをカスタマイズします。
3. ソフトウェアまたはルートボリュームのカスタマイズをテストして、正しく動作することを確認します。
4. 「RDS Custom for Oracle でのシングル AZ 配置からマルチ AZ 配置への変換」の手順に従って、シングル AZ DB インスタンスをマルチ AZ 配置に変換します。
5. マルチ AZ 配置の両方のインスタンスにカスタマイズが存在することを確認します。詳細については、「RDS Custom for Oracle マルチ AZ 配置での EC2 インスタンスの識別」を参照してください。

RDS Custom for Oracle マルチ AZ 配置を作成した後に OS をカスタマイズする

マルチ AZ 配置を終了している場合は、AWS Systems Manager または手動操作を使用してカスタマイズをデプロイできます。

AWS Systems Manager を使用したマルチ AZ 配置での OS のカスタマイズ

既存のマルチ AZ DB インスタンスでは、プライマリインスタンスとスタンバイインスタンスの両方に同時にカスタマイズを適用する最も信頼性の高い方法として Systems Manager をお勧めします。このアプローチにより、一貫性が確保されます。このサービスの一般的な概要については、「AWS Systems Manager とは」を参照してください。両方の DB インスタンスに同時にソフトウェアをインストールする方法については、「Distributor パッケージをインストールまたは更新する」を参照してください。

マルチ AZ 配置での OS の手動カスタマイズ

このシナリオでは、マルチ AZ 配置は既に存在しますが、カスタマイズのデプロイには AWS Systems Manager を使用しません。OS は、次のいずれかの方法で手動でカスタマイズできます。

プライマリインスタンスで OS をカスタマイズし、変更をレプリケートする

マルチ AZ 配置では、rdsdbdata ボリュームが自動的にレプリケートされます。OS は以下の方法でカスタマイズできます。

- サードパーティー製ソフトウェアを /rdsdbdata マウントポイントに直接インストールします。
- ルートボリューム (/) のファイルを変更するには、データボリュームにファイルを作成し、ルートボリュームファイルからデータボリュームファイルへのシンボリックリンクを作成します。

プライマリインスタンスとスタンバイインスタンスの OS を個別にカスタマイズする

この方法では、プライマリインスタンスの OS をカスタマイズします。次に、スタンバイインスタンスで同じカスタマイズを実行します。

プライマリインスタンスの OS を自動的にレプリケートするようにカスタマイズするには

1. RDS Custom for Oracle マルチ AZ 配置での EC2 インスタンスの識別 の手順に従って、プライマリ DB インスタンスとスタンバイ DB インスタンスを特定します。
2. Session Manager または SSH を使用してプライマリ EC2 インスタンスに接続します。
3. ビジネス要件に応じて、次のいずれかの手法を使用します。

   サードパーティーソフトウェアをインストールする

   /rdsdbdata マウントポイントにソフトウェアをインストールします。

   ```
   sudo mkdir -p /rdsdbdata/custom-software
   cd /rdsdbdata/custom-software
   # Install your software here
   ```

   ルートボリュームをカスタマイズする

   ルートボリュームの OS 設定ファイルからデータボリュームのファイルへのシンボリックリンクを作成します。例えば、ファイル /rdsdbdata/customizations/sysctl.conf を作成し、/rdsdbdata/customizations/sysctl.conf を指すシンボリックリンクを /etc/sysctl.conf に作成します。

   ```
   sudo mkdir -p /rdsdbdata/customizations
   sudo mv /etc/sysctl.conf /rdsdbdata/customizations/sysctl.conf
   sudo ln -sf /rdsdbdata/customizations/sysctl.conf /etc/sysctl.conf
   ```

4. ソフトウェアまたはルートボリュームのカスタマイズをテストして、正しく動作することを確認します。

5. スタンバイインスタンスに接続し、同期レプリケーションがソフトウェアまたはルートボリュームのカスタマイズを /rdsdbdata ディレクトリにコピーしたことを確認します。

RDS Custom for Oracle マルチ AZ 配置でのバイナリボリュームのカスタマイズ

RDS Custom for Oracle マルチ AZ 配置のバイナリボリューム (/rdsdbbin) にデータベースパッチを適用できます。プライマリインスタンスとスタンバイインスタンスにパッチを適用する必要があります。以下のガイドラインを検討します。

- 1 回限りのパッチを実行する場合は、マニフェストに含まれている新しい 1 回限りのパッチを使用して新しい CEV を作成することをお勧めします。
- 1 回限りのパッチを手動で適用するには、プライマリ EC2 インスタンスとセカンダリ EC2 インスタンスの両方で 1 回限りのパッチを解凍してください。パッチの適用と datapatch の実行は、プライマリマルチ AZ インスタンスでのみ必要です。
- 別の CEV を使用してデータベースにパッチを適用すると、バイナリボリュームが置き換えられます。新しい CEV のマニフェストに 1 回限りのパッチを必ず含めてください。

OS をカスタマイズするためのベストプラクティス

RDS Custom for Oracle マルチ AZ インスタンスで OS をカスタマイズする場合は、次のベストプラクティスに従ってください。

本番稼働用環境以外でテストする

本番稼働用インスタンスに適用する前に、本番稼働用以外の環境でカスタマイズを必ずテストしてください。

すべての変更を文書化する

今後の参照とトラブルシューティングのために、すべてのカスタマイズの詳細なドキュメントを維持します。カスタマイズは、念のため、いつでも適用できるスクリプトに保存することをお勧めします。

両方のインスタンスで検証する

プライマリインスタンスとスタンバイインスタンスの両方でカスタマイズが存在し、正しく機能していることを定期的に確認します。

Systems Manager を使用して一貫性を保つ

Systems Manager を使用して、インスタンス全体、特に既存のマルチ AZ 配置の変更を一貫して適用します。

AWS CloudFormation で RDS Custom for Oracle をデプロイする

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

提供された AWS CloudFormation テンプレートを使用して RDS Custom for Oracle のデプロイを自動化します。リソースをデプロイする前に、次の前提条件を完了してください。

前提条件

必要な Oracle ファイルのダウンロード

CloudFormation テンプレートを作成する前に、特定の Oracle インストールファイルが必要です。デプロイする前に、これらのファイルをダウンロードしてください。

1. Oracle Database 19c (19.3) に移動する
2. ファイル `LINUX.X64_193000_db_home.zip` を見つけてダウンロードする
3. ファイルの名前を `V982063-01.zip` に変更する
4. 残りのパッチをダウンロードし、プラットフォームまたは言語を `Linux x86-64` として選択します。

最新の OPatch ユーティリティ

パッチ 6880880

2023 年 1 月 PSU パッチ

データベース PSU および RU パッチ

- パッチ 34765931
- パッチ 34786990

追加の必須パッチ

- パッチ 35099667
- パッチ 35099674
- パッチ 28730253
- パッチ 29213893
- パッチ 35012866

2023 年 4 月 PSU パッチ

データベース PSU および RU パッチ

- パッチ 35042068
- パッチ 35050341

追加の必須パッチ

- パッチ 28730253
- パッチ 29213893
- パッチ 33125873
- パッチ 35220732
- パッチ 35239280

Amazon S3 バケットのセットアップ

1. AWS アカウント で Amazon S3 バケットを作成するか、既存のバケットを選択します。
2. 以下の例のようなフォルダ構造をバケットに作成します。

```
<bucket-name>/
### oracle_cev/
    ### V982063-01.zip
    ### p6880880_190000_Linux-x86-64.zip
```

```
### p34765931_190000_Linux-x86-64.zip
### p34786990_190000_Linux-x86-64.zip
### p35099667_190000_Linux-x86-64.zip
### p35099674_190000_Generic.zip
### p28730253_190000_Linux-x86-64.zip
### p29213893_1918000DBRU_Generic.zip
### p35012866_1918000DBRU_Linux-x86-64.zip
### p35042068_190000_Linux-x86-64.zip
### p35050341_190000_Linux-x86-64.zip
### p29213893_1919000DBRU_Generic.zip
### p33125873_1919000DBRU_Linux-x86-64.zip
### p35220732_190000_Linux-x86-64.zip
### p35239280_190000_Generic.zip
```

3. 以前にダウンロードしたすべての Oracle ファイルを適切なフォルダにアップロードします。

AWS CloudFormation を使用して RDS Custom for Oracle をデプロイする

ステップ 1: CloudFormation テンプレートを準備する

RDS Custom for Oracle をデプロイする前に、必要な前提条件を作成する CloudFormation テンプレートをダウンロードして設定する必要があります。

テンプレートをコピーして保存する

1. [単一および複数のアベイラビリティーゾーンを使用した RDS Custom for Oracle のデプロイ] に進む
2. テンプレートを任意の形式 (YAML または JSON) でコピーする
3. ファイルを YAML または JSON 形式で保存します。例: `rds-custom-oracle-prereqs.yaml`

AWS コンソールからスタックを起動する

1. AWS コンソールを開き、AWS CloudFormation に移動する
2. [スタックの作成]、[新しいリソースの使用 (標準)] の順に選択する
3. [既存のテンプレートを選択する] を選択する
4. [テンプレートファイルのアップロード]、[ファイルの選択] の順に選択する
5. 以前にダウンロードしたテンプレートファイルを選択する
6. デフォルトのパラメータ値を保持する

7. [次へ] を選択してスタックを作成する

AWS CLI を使用する方法

コンソールを使用する代わりに、AWS CLI を使用してスタックを作成することもできます。

```
aws cloudformation create-stack \
  --stack-name rds-custom-oracle \
  --template-body file://rds-custom-oracle-prereqs.yaml \
  --capabilities CAPABILITY_NAMED_IAM
```

ステップ 2: カスタムエンジンバージョン (CEV) と Amazon RDS インスタンスを作成する

テンプレートをコピーして保存する

1. [単一および複数のアベイラビリティーゾーンを使用した RDS Custom for Oracle のデプロイ] に進む
2. テンプレートを任意の形式 (YAML または JSON) でコピーする
3. 必要に応じて、テンプレートで次のパラメータを更新する
   - `BucketName`
   - `CEVS3Prefix`
   - データベースマスターパスワード (********** を置き換える)
4. ファイルを YAML または JSON 形式で保存する

ステップ 3: AWS コンソールを使用してデプロイする

1. AWS コンソールを開き、AWS CloudFormation に移動する
2. [スタックの作成]、[新しいリソースの使用 (標準)] の順に選択する
3. [既存のテンプレートを選択する] を選択する
4. [テンプレートファイルのアップロード]、[ファイルの選択] の順に選択する
5. 以前にダウンロードしたテンプレートファイルを選択する
6. パラメータはデフォルト値のままにしておく
7. パラメータを次のように入力する

```
BucketName: rds-custom-id
CEVS3Prefix: oracle_cev
```

```
CEVCreation: Yes
```

8. 設定を確認し、[次へ] を選択してスタックを作成する

オプション: AWS CLI を使用してデプロイする

```
aws cloudformation create-stack \
  --stack-name rds-custom-oracle \
  --template-body file://rds-custom-oracle.yaml \
  --parameters \
    ParameterKey=BucketName,ParameterValue=rds-custom-id \
    ParameterKey=CEVS3Prefix,ParameterValue=oracle_cev \
    ParameterKey=CEVCreation,ParameterValue=Yes \
  --capabilities CAPABILITY_NAMED_IAM
```

作成されたデプロイリソース

テンプレートによって以下のリソースが作成されます。

- パブリックサブネットとプライベートサブネットを持つ Amazon VPC
- セキュリティグループ
- Amazon VPC エンドポイント
- IAM ロールとポリシー
- 暗号化用の AWS KMS キー
- カスタムエンジンバージョン (CEV)
- シングル AZ 設定とマルチ AZ 設定の両方の RDS Custom for Oracle インスタンス

デプロイの進行状況をモニタリングする

CloudFormation スタックを作成したら、その進行状況をモニタリングしてデプロイが成功したことを確認します。デプロイプロセスには、カスタムエンジンバージョン (CEV) と RDS インスタンスの作成が含まれます。

デプロイの進行状況をモニタリングするには

1. CloudFormation コンソールを開きます。
2. スタック名を選択します。
3. [イベント] タブを選択して進行状況を表示し、エラーを特定します。

> **Note**
>
> CEV の作成には通常 2～3 時間かかります。CEV の作成が正常に完了すると、Amazon RDS は自動的に Amazon RDS インスタンスの作成を開始します。

デプロイ後

スタック作成プロセスが完了したら、デプロイ後の検証と設定の以下のステップを実行します。

1. Amazon RDS コンソールページから、[カスタムエンジンバージョン] に移動して CEV の作成を確認します。
2. Amazon RDS インスタンスが作成され、使用可能であることを確認します。
3. Amazon RDS インスタンスへの接続をテストします。
4. 必要に応じてモニタリングとバックアップの方法を設定します。

クリーンアップ

すべてのリソースを削除するには、以下の AWS CLI コマンドを実行します。

```
aws cloudformation delete-stack --stack-name rds-custom-oracle
```

トラブルシューティング

デプロイ中に問題が発生した場合は、次の解決策を使用して一般的な問題を解決します。

CEV 作成の失敗

- 必要なすべてのパッチが Amazon S3 にアップロードされていることを確認します。
- IAM 許可をチェックします。
- パッチのバージョンが正しいことを確認します。必要なパッチのリストについては、「the section called “前提条件”」を参照してください。

Amazon RDS インスタンスの作成が失敗する

- VPC/サブネット設定を確認する
- セキュリティグループルールを確認する
- CEV が使用可能であることを確認する

RDS Custom for Oracle の Oracle レプリカの使用

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Oracle Enterprise Edition を実行する、RDS Custom for Oracle DB インスタンスの Oracle レプリカを作成できます。コンテナデータベース (CDB) と非 CDB の両方がサポートされています。Standard Edition 2 は、Oracle Data Guard をサポートしていません。

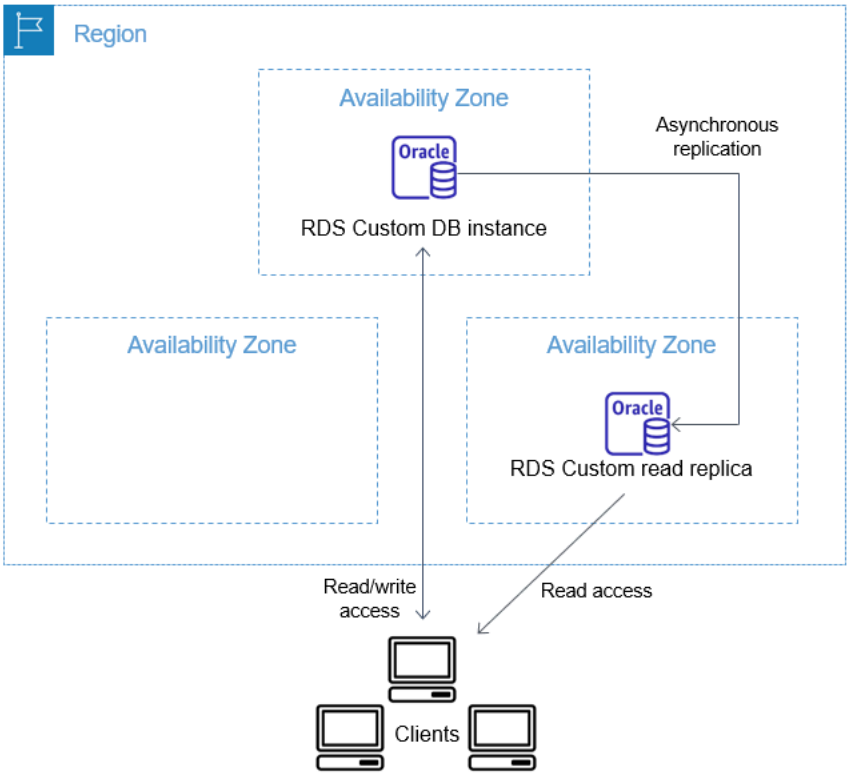
RDS Custom for Oracle レプリカの作成は、RDS for Oracle レプリカの作成に似ていますが、いくつかの重要な違いがあります。Oracle レプリカの作成および管理の一般情報については、「DB インスタンスのリードレプリカの操作」 および 「Amazon RDS for Oracle でのリードレプリカの使用」 を参照してください。

トピック

- RDS Custom for Oracle レプリケーションの概要
- Amazon RDS Custom for Oracle レプリケーションのガイドラインと制限事項
- RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格させる
- RDS Custom for Oracle プライマリインスタンスとレプリカインスタンス間の VPN トンネルの設定

RDS Custom for Oracle レプリケーションの概要

RDS Custom for Oracle レプリケーションのアーキテクチャは、RDS for Oracle レプリケーションに似ています。プライマリ DB インスタンスは 1 つ以上の Oracle レプリカに非同期でレプリケートします。



レプリカの最大数

RDS for Oracle と同様に、RDS Custom for Oracle プライマリ DB インスタンスに対して、最大 5 つのマネージド Oracle レプリカを作成できます。手動で設定した独自 (外部) の Oracle レプリカを作成することもできます。外部レプリカは DB インスタンスの制限にはカウントされません。RDS Custom サポートペリメーターの外にもあります。サポートされる形式の詳細については、「RDS Custom サポート範囲」を参照してください。

レプリカの命名規則

Oracle レプリカ名は、データベースの一意の名前に基づいています。形式は *DB_UNIQUE_NAME_X* で、文字が順次追加されます。例えば、データベースの一意の名前が ORCL であれば、最初の 2 つのレプリカの名前は ORCL_A と ORCL_B になります。初期の 6 文字 A—F は RDS Custom 用に予約されています。RDS Custom は、プライマリ DB インスタンスからレプリカにデータベースパラメータをコピーします。詳細については、Oracle ドキュメントの「DB_UNIQUE_NAME」を参照してください。

レプリカのバックアップ保持

デフォルトでは、RDS Custom Oracle レプリカは、プライマリ DB インスタンスと同じバックアップ保持期間を使用します。バックアップ保持期間を 1～35 日に変更することができます。RDS

Custom は、バックアップ、リストア、ポイントインタイムリカバリ (PITR) をサポートしています。RDS カスタム DB インスタンスのバックアップと復元の詳細については、「Amazon RDS Custom for Oracle DB インスタンスのバックアップと復元」を参照してください。

> **Note**
>
> Oracle レプリカを作成している間、RDS Custom は REDO ログファイルのクリーンアップを一時的に一時停止します。このようにして、RDS Custom は、新しい Oracle レプリカが使用可能になった後に、これらのログを確実に適用できるようにします。

レプリカの昇格

コンソール、`promote-read-replica` AWS CLI コマンド、または `PromoteReadReplica` API を使用して、RDS Custom for Oracle のマネージド Oracle レプリカを昇格させることができます。プライマリ DB インスタンスを削除し、すべてのレプリカが正常であれば、RDS Custom for Oracle はマネージドレプリカをスタンドアロンインスタンスに自動的に昇格させます。レプリカが自動化を一時停止しているか、サポートペリメーター外にある場合は、RDS Custom が自動的にレプリカを昇格させる前にレプリカを修正する必要があります。外部の Oracle レプリカは手動でのみ昇格できます。

Amazon RDS Custom for Oracle レプリケーションのガイドラインと制限事項

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle リードレプリカの作成時に、すべての RDS Oracle レプリカオプションがサポートされるわけではありません。

トピック

- RDS Custom for Oracle レプリケーションの一般的なガイドライン
- RDS Custom for Oracle レプリケーションの一般的な制限事項

- RDS Custom for Oracle レプリケーションのネットワーク要件と制限事項
- RDS Custom for Oracle の外部レプリカの制限事項

RDS Custom for Oracle レプリケーションの一般的なガイドライン

RDS Custom for Oracle を使用するときは、次のガイドラインに従ってください。

- RDS Custom for Oracle レプリケーションは、Oracle Enterprise Edition でのみ使用できます。Standard Edition 2 はサポートされていません。
- VPN トンネルを実装して、プライマリインスタンスとスタンバイインスタンス間の通信を暗号化することを強くお勧めします。詳細については、「RDS Custom for Oracle プライマリインスタンスとレプリカインスタンス間の VPN トンネルの設定」を参照してください。
- RDS_DATAGUARDユーザーを変更しないでください。このユーザーは RDS Custom for Oracle オートメーション用に予約されています。このユーザーを変更すると、RDS Custom for Oracle DB インスタンスの Oracle レプリカを作成できないなど、望ましくない結果が発生することがあります。
- レプリケーションユーザーのパスワードは変更しないでください。このパスワードは、RDS Custom ホスト上で Oracle Data Guard 設定を管理するために必要です。RDS Custom for Oracle でパスワードを変更すると、Oracle レプリカがサポートペリメーターの外に配置される可能性があります。詳細については、「RDS Custom サポート範囲」を参照してください。

  パスワードは、DB リソース ID とタグ付けされ、AWS Secrets Manager に保存されます。Secrets Manager は、各 Oracle レプリカに独自のシークレットがあります。シークレットは、次のいずれかの命名形式を使用します。

```
do-not-delete-rds-custom-db-DB_resource_id-uuid-dg
rds-custom!oracle-do-not-delete-DB_resource_id-uuid-dg
```

- プライマリ DB インスタンスの DB_UNIQUE_NAME は変更しないでください。名前を変更すると、復元オペレーションが停止します。
- RDS Custom CDB の CREATE PLUGGABLE DATABASE コマンドに句 STANDBYS=NONE を指定しないでください。これにより、フェイルオーバーが発生しても、スタンバイ CDB にすべての PDB が含まれるようになります。

RDS Custom for Oracle レプリケーションの一般的な制限事項

RDS Custom for Oracle レプリカの制限は以下のとおりです。

- RDS Custom for Oracle レプリカは読み取り専用モードで作成することはできません。ただし、レプリカのモードをマウントから読み取り専用に変更したり、読み取り専用からマウントに変更する操作を手動で行うことができます。詳細については、create-db-instance-read-replica AWS CLIコマンドのドキュメントを参照してください。
- クロスリージョン RDS Custom for Oracle レプリカを作成することはできません。
- Oracle Data Guard `CommunicationTimeout` パラメータの値を変更することはできません。RDS Custom for Oracle DB インスタンスでこのパラメータを 15 秒に設定します。

RDS Custom for Oracle レプリケーションのネットワーク要件と制限事項

ネットワーク設定が RDS Custom for Oracle レプリカをサポートしていることを確認します。以下の点を考慮してください。

- プライマリ DB インスタンスおよびすべてのレプリカの仮想プライベートクラウド (VPC) 内のインバウンドおよびアウトバウンド通信の両方で、ポート 1140 を有効にしてください。これは、Oracle Data Guard のリードレプリカ間の通信に必要です。
- RDS Custom for Oracle は、Oracle レプリカの作成中にネットワークを検証します。プライマリ DB インスタンスと新しいレプリカがネットワーク経由で接続できない場合、RDS Custom for Oracle はレプリカを作成せず、`INCOMPATIBLE_NETWORK` 状態にします。
- Amazon EC2 またはオンプレミスで作成する外部 Oracle レプリカの場合は、Oracle Data Guard レプリケーションに別のポートとリスナーを使用します。ポート 1140 を使用しようとすると、RDS Custom オートメーションと競合する可能性があります。
- `/rdsdbdata/config/tnsnames.ora`ファイルには、リスナープロトコルアドレスにマップされたネットワークサービス名が含まれています。以下の要件と推奨事項に留意してください。
  - `tnsnames.ora` に `rds_custom_` のプレフィックスが付いたエントリーは、Oracle レプリカオペレーションを処理する際に RDS Custom に予約されます。

    `tnsnames.ora`で手動でエントリーを作成する場合、このプレフィックスは使用しないでください。
  - 場合によっては、手動でスイッチオーバーまたはフェイルオーバーするか、Fast-Start Failover (FSFO) などのフェイルオーバーテクノロジーを使用することをお勧めします。その場合は、マニュアルで`tnsnames.ora`エントリーをプライマリ DB インスタンスからすべてのスタンバイインスタンスに同期してください。このレコメンデーションは、RDS Custom で管理される Oracle レプリカと外部 Oracle レプリカの両方に適用されます。

RDS Custom オートメーションは、`tnsnames.ora` エントリーをプライマリ DB インスタンスでのみアップデートします。Oracle レプリカを追加または削除するときも、必ず同期してください。

同期しない場合、`tnsnames.ora`ファイルを切り替えてマニュアルでフェイルオーバーすると、プライマリ DB インスタンス上の Oracle Data Guard が Oracle レプリカと通信できない場合があります。

RDS Custom for Oracle の外部レプリカの制限事項

RDS Custom for Oracle 外部レプリカ (オンプレミスのレプリカを含む) には次の制限があります。

- RDS Custom for Oracle は、FSFO などの外部 Oracle レプリカの手動フェイルオーバー時にインスタンスロールの変更を検出しません。

  RDS Custom for Oracle は、マネージドレプリカの変更を検出します。ロールの変更は、イベントログに記録されます。新しい状態は、describe-db-instances AWS CLIコマンドを使用して見ることができます。
- RDS Custom for Oracle は、外部 Oracle レプリカの高いレプリケーションラグを検出しません。

  RDS Custom for Oracle は、マネージドレプリカのラグを検出します。レプリケーションラグが高い場合は、`Replication has stopped`イベントが発生します。レプリケーションステータスは、describe-db-instances AWS CLIコマンドを実行して見ることができますが、更新に遅延がある可能性があります。
- RDS Custom for Oracle は、プライマリ DB インスタンスを削除しても、外部の Oracle レプリカを自動的に昇格させません。

  自動昇格機能は、マネージド Oracle レプリカでのみ使用できます。Oracle レプリカを手動で昇格する方法については、ホワイトペーパー「Amazon RDS Custom for Oracle でのデータガードによる高可用性の有効化」を参照してください。

RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格させる

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは

RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS for Oracle と同様に、RDS Custom for Oracle レプリカをスタンドアロンの DB インスタンスに昇格させることができます。Oracle レプリカを昇格させると、RDS Custom for Oracle は、使用可能になる前に DB インスタンスが再起動します。Oracle レプリカの昇格についての詳細は、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

レプリカを昇格させる場合は、以下のガイドラインに注意してください。

- RDS Custom for Oracle がレプリカを昇格させている間は、フェイルオーバーを開始しないようにしてください。そうしないと、昇格ワークフローが停止する可能性があります。
- RDS Custom for Oracle が Oracle レプリカを昇格させている間は、プライマリ DB インスタンスを切り替えないようにしてください。そうしないと、昇格ワークフローが停止する可能性があります。
- RDS Custom for Oracle が Oracle レプリカを昇格させている間は、プライマリ DB インスタンスをシャットダウンしないようにしてください。そうしないと、昇格ワークフローが停止する可能性があります。
- 新しく昇格した DB インスタンスをターゲットとしてレプリケーションを再開しようとしないでください。RDS Custom for Oracle が Oracle レプリカを昇格させると、そのレプリカはスタンドアロンの DB インスタンスになり、レプリカのロールは保有しなくなります。

RDS Custom for Oracle レプリカの昇格について、以下の制事項に注意してください。

- RDS Custom for Oracle がバックアップを行っている間は、レプリカを昇格させることはできません。
- Oracle レプリカを昇格させる場合、バックアップ保持期間を `0` に変更することはできません。
- レプリカが正常な状態でないと、レプリカを昇格させることはできません。

  プライマリ DB インスタンスで `delete-db-instance` を発行した場合、RDS Custom for Oracle は、それぞれのマネージド Oracle レプリカが正常であり、昇格に利用可能であることを検証します。自動化が一時停止されているか、サポートペリメーター外にあるため、レプリカを昇格できない場合があります。このような場合、RDS Custom for Oracle は問題を説明するイベントを公開し、Oracle レプリカを手動で修復できるようにします。

以下のステップは、DB インスタンスに Oracle レプリカを昇格させる一般的なプロセスを示しています。

1. プライマリ DB インスタンスに対するトランザクションの書き込みをすべて停止します。
2. RDS Custom for Oracle がすべての更新を Oracle レプリカに適用するまでお待ちください。
3. Amazon RDS コンソールの [昇格] オプション、AWS CLI コマンドの `promote-read-replica`、または Amazon RDS API の `PromoteReadReplica` オペレーションを使用して、Oracle レプリカを昇格させます。

Oracle レプリカの昇格には数分かかります。プロセス中、RDS Custom for Oracle はレプリケーションを停止し、レプリカを再起動します。再起動が完了すると、Oracle レプリカは新しい DB インスタンスとして使用可能になります。レプリカの昇格のトラブルシューティングについては、「RDS Custom for Oracle のレプリカプロモーションのトラブルシューティング」を参照してください。

コンソール

RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格させるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

   [データベース] ペインが表示されます。各 Oracle レプリカには、[Role] (ロール) 列に [Replica] (レプリカ) があります。
3. 昇格させる RDS Custom for Oracle レプリカを選択します。
4. [アクション] で、[Promote (昇格)] を選択します。
5. [Promote Oracle replica] (Oracle レプリカの昇格) ページで、新しく昇格された DB インスタンスのバックアップ保持期間とバックアップウィンドウを入力します。この値を 0 に設定することはできません。
6. 希望通りの設定になったら、[Promote Oracle replica] (Oracle レプリカの昇格) を選択します。

AWS CLI

RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格させるには、AWS CLI `promote-read-replica` コマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds promote-read-replica \
--db-instance-identifier my-custom-read-replica \
--backup-retention-period 2 \
--preferred-backup-window 23:00-24:00
```

Windows の場合:

```
aws rds promote-read-replica ^
--db-instance-identifier my-custom-read-replica ^
--backup-retention-period 2 ^
--preferred-backup-window 23:00-24:00
```

RDS API

RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格するには、必須のパラメータ `DBInstanceIdentifier` を指定して Amazon RDS API の `PromoteReadReplica` オペレーションを呼び出します。

RDS Custom for Oracle プライマリインスタンスとレプリカインスタンス間の VPN トンネルの設定

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

VPN トンネルは、ネットワーク上の 2 つ以上のデバイス間の暗号化された接続です。RDS Custom for Oracle で Oracle Data Guard インスタンスのセキュリティを最高レベルまで高めるために、プライマリインスタンスとスタンバイインスタンス間に VPN トンネルを実装し、通信を暗号化するすることを強くお勧めします。VPN トンネルは、インスタンス間で機密データがネットワークを移動する際のにセキュリティを保護します。この設定はオプションですが、データを保護し、規制に準拠するためのベストプラクティスとしてお勧めします。

次の前提条件を満たしていることを確認します。

- プライマリホストとスタンバイホストへのルートアクセスがあること。
- ipsec コマンドを実行するための技術的な専門知識があること。

RDS Custom for Oracle でプライマリとレプリカ間に VPN トンネルを設定するには

1. 次のルールを使用して、プライマリインスタンスとスタンバイインスタンスの両方のセキュリティグループを許可リストに追加します。

```
ACTION FLOW SOURCE PROTO PORT

ALLOW ingress this-SG 50 (ESP) all (N/A)
ALLOW egress this-SG 50 (ESP) all (N/A)

ALLOW ingress this-SG 17 (UDP) 500 (IKE)
ALLOW egress this-SG 17 (UDP) 500 (IKE)
```

2. root ユーザーに切り替えます。

```
$ sudo su - root
```

3. プライマリインスタンスとスタンバイインスタンスの両方で次のコマンドを実行して、root ユーザーのネットワークセキュリティサービス (NSS) データベースを初期化します。

```
ipsec initnss --nssdir /etc/ipsec.d
```

4. 次のように RSA キーを生成します。

   a. プライマリインスタンスで、OS のバージョンに応じて次のいずれかの ipsec コマンドを使用し、キーを生成します。

```
ipsec newhostkey --nssdir /etc/ipsec.d       ## for Oracle Linux Version 8
ipsec newhostkey --output /etc/ipsec.secrets ## for Oracle Linux version 7.9
```

   b. 構成を作成するために必要なパブリックキーを取得します。次の ipsec の例では、left が設定中のデバイスを参照し、right がトンネルの他方の終端にあるデバイスを参照するため、プライマリインスタンスは left になります。

```
ipsec showhostkey --left --ckaid ckaid-returned-in-last-statement
```

c. スタンバイインスタンスで、スタンバイインスタンスのキーを生成します。

```
ipsec newhostkey --nssdir /etc/ipsec.d       ## for Oracle Linux Version 8
ipsec newhostkey --output /etc/ipsec.secrets ## for Oracle Linux version 7.9
```

d. スタンバイ インスタンスの構成を作成するために必要なパブリックキーを取得します。次の例では、トンネルの他方の終端にあるデバイスを参照している right がスタンバイインスタンスになります。

```
ipsec showhostkey --right --ckaid ckaid-returned-in-last-statement
```

5. 取得した RSA キーに基づいて、構成を生成します。構成は、プライマリインスタンスとスタンバイインスタンスで同じになります。プライマリインスタンスの IPv4 アドレスも、スタンバイインスタンスの IPv4 アドレスも、AWS コンソールで確認できます。

   プライマリインスタンスとスタンバイインスタンスの両方で、次の構成をファイル /etc/ipsec.d/custom-fb-tunnel.conf に保存します。

```
conn custom-db-tunnel
 type=transport
 auto=add
 authby=rsasig
 left=IPV4-for-primary
 leftrsasigkey=RSA-key-generated-on-primary
 right=IPV4-for-standby
 rightrsasigkey=RSA-key-generated-on-standby
```

6. プライマリインスタンスとスタンバイインスタンスの両方のホストで ipsec デーモンを起動します。

```
ipsec setup start
```

7. プライマリインスタンスまたはスタンバイインスタンスでトンネルを起動します。出力は以下の例のようになります。

```
[root@ip-172-31-6-81 ~]# ipsec auto --up custom-db-tunnel
181 "custom-db-tunnel" #1: initiating IKEv2 connection
181 "custom-db-tunnel" #1: sent IKE_SA_INIT request to 172.31.32.196:500
182 "custom-db-tunnel" #1: sent IKE_AUTH request {cipher=AES_GCM_16_256 integ=n/a
 prf=HMAC_SHA2_512 group=DH19}
```

```
003 "custom-db-tunnel" #1: initiator established IKE SA; authenticated peer '3584-
bit PKCS#1 1.5 RSA with SHA1' signature using preloaded certificate '172.31.32.196'
004 "custom-db-tunnel" #2: initiator established Child SA using #1; IPsec transport
 [172.31.6.81-172.31.6.81:0-65535 0] -> [172.31.32.196-172.31.32.196:0-65535 0]
 {ESP/ESN=>0xda9c4815 <0xb742ca42 xfrm=AES_GCM_16_256-NONE DPD=passive}
[root@ip-172-31-6-81 ~]#
```

Amazon RDS Custom for Oracle DB インスタンスのバックアップと復元

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS 同様、RDS Custom は、DB インスタンスのバックアップ中に RDS Custom for Oracle DB インスタンスの自動バックアップを作成して保存します。また、手動で DB インスタンスをバックアップすることもできます。

手順は、Amazon RDS DB インスタンスのスナップショット取得と同じです。 RDS Custom DB インスタンスの初期のスナップショットには、フル DB インスタンスのデータが含まれています。その後のスナップショットは増分です。

AWS マネジメントコンソール または AWS CLI を使用して DB スナップショットを復元します。

トピック

- RDS Custom for Oracle スナップショットの作成
- RDS Custom for Oracle DB スナップショットからの復元
- RDS Custom for Oracle インスタンスをポイントインタイムに復元する
- RDS Custom for Oracle スナップショットの削除
- RDS Custom for Oracle 自動バックアップの削除

RDS Custom for Oracle スナップショットの作成

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、DB インスタンス全体をバックアップします。DB インスタンスにコンテナデータベース (CDB) が含まれている場合、インスタンスのスナップショットにはルート CDB とすべての PDB が含まれます。

RDS Custom for Oracle スナップショットを作成するときに、バックアップする RDS Custom DB インスタンスを指定します。次に、スナップショットに名前を付けて、後でそこから復元できるようにします。

スナップショットを作成すると、RDS Custom for Oracle は DB インスタンスに添付されたすべてのボリュームに対して Amazon EBS スナップショットを作成します。RDS Custom for Oracle は、ルートボリュームの EBS スナップショットを使用して新しい Amazon マシンイメージ (AMI) を登録します。特定の DB インスタンスと簡単に関連付けられるように、スナップショットには `DBSnapshotIdentifier`、`DbiResourceId`、`VolumeType` のタグが付けられています。

DB スナップショットを作成すると、短時間の I/O が発生します。この一時停止は、DB インスタンスのサイズとクラスに応じて、数秒から数分続く場合があります。スナップショットの作成時間は、データベースのサイズによって異なります。スナップショットにはストレージボリューム全体が含まれており、テンポラリファイルなどのファイルサイズも、スナップショットの作成時間に影響します。スナップショットの作成の詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

コンソールまたはAWS CLIを使用して RDS Custom for Oracle スナップショットを作成します。

コンソール

RDS Custom スナップショットを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで [データベース] を選択します。
3. RDS Custom DB インスタンスのリストで、スナップショットを取得する DB インスタンスを選択します。
4. アクション で、スナップショットの取得 を選択します。

   [Take DB snapshot] (DB スナップショットの取得) ウィンドウが表示されます。
5. [スナップショット名] に、スナップショットの名前を入力します。
6. スナップショットの取得 を選択します。

AWS CLI

create-db-snapshot AWS CLIコマンドを使用して、RDS Custom DB インスタンスのスナップショットを作成します。

以下のオプションを指定します。

- `--db-instance-identifier` - バックアップする RDS Custom DB インスタンスを指定します。
- `--db-snapshot-identifier` - RDS Custom スナップショットに名前を付けて、後でそこから復元できるようにします。

この例では、*my-custom-instance* という RDS Custom DB インスタンスの、*my-custom-snapshot* という DB スナップショットを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-snapshot \
    --db-instance-identifier my-custom-instance \
    --db-snapshot-identifier my-custom-snapshot
```

Windows の場合:

```
aws rds create-db-snapshot ^
    --db-instance-identifier my-custom-instance ^
    --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for Oracle DB スナップショットからの復元

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle DB インスタンスを復元するときは、DB スナップショットの名前と新しいインスタンスの名前を指定します。スナップショットから既存の RDS Custom DB インスタンスに復元することはできません。復元するときに新しい RDS Custom for Oracle DB インスタンスが作成されます。

復元プロセスは、Amazon RDS での復元とは以下の点が異なります。

- スナップショットを復元する前に、RDS Custom for Oracle は既存の設定ファイルをバックアップします。これらのファイルは、ディレクトリ`/rdsdbdata/config/backup`内の復元したインスタンスで使用できます。RDS Custom for Oracle は、DB スナップショットをデフォルトのパラメータで復元し、以前のデータベース設定ファイルを既存のもので上書きします。そのため、復元したインスタンスは、カスタムパラメータとデータベース設定ファイルの変更を保持しません。
- 復元したデータベースの名前は、スナップショットと同じです。必要に応じて、別の名前を指定できます。(RDS Custom for Oracle の場合、デフォルトは`ORCL`です。)

コンソール

DB スナップショットから RDS Custom DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. 復元の元にする DB スナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。
5. DB インスタンスの復元 ページで、DB インスタンス識別子 に、復元した RDS Custom DB インスタンスの名前を入力します。
6. DB インスタンスの復元 を選択します。

AWS CLI

RDS Custom DB スナップショットを復元するには、DBスナップショットからDBインスタンスを復元する AWS CLI コマンドを使用します。

復元元のスナップショットがプライベート DB インスタンスの場合、db-subnet-group-name とno-publicly-accessible の両方を正しく指定してください。そうでなければ、DB インスタンスはデフォルトでパブリックアクセスに設定されます。以下のオプションは必須です。

- db-snapshot-identifier - 復元元のスナップショットを識別します。
- db-instance-identifier - DB スナップショットから作成する RDS Custom DB インスタンスの名前を指定します。
- custom-iam-instance-profile — RDS Custom DB インスタンスの基盤となる Amazon EC2 インスタンスに関連付けられているインスタンスプロファイルを指定します。

次のコードは、my-custom-instance の my-custom-snapshot という名前のスナップショットを復元します。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot \
  --db-instance-identifier my-custom-instance \
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
  --no-publicly-accessible
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot ^
  --db-instance-identifier my-custom-instance ^
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
  --no-publicly-accessible
```

RDS Custom for Oracle インスタンスをポイントインタイムに復元する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

DB インスタンスを特定のポイントに復元し、新しい DB インスタンス を作成できます。PITR をサポートするには、DB インスタンスのバックアップ保持がゼロ以外の値に設定されている必要があります。

RDS Custom for Oracle DB インスタンスの最新の復元可能時間はいくつかの要因によって異なりますが、通常は現在時間から 5 分以内です。DB インスタンスの復元可能な直近の時間を確認するには、AWS CLI の describe-db-instances コマンドを使用し、DB インスタンスの [`LatestRestorableTime`] フィールドに返される値を確認します。Amazon RDS コンソールで各 DB インスタンスの復元可能な直近の時間を表示するには、[自動バックアップ] をクリックします。

バックアップ保持期間の任意の時点に復元できます。各 DB インスタンスの復元可能な最も早い時間を表示するには、Amazon RDS コンソールで [自動バックアップ] をクリックします。

PITR の一般情報については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

トピック

- RDS Custom for Oracle の PITR に関する考慮事項

RDS Custom for Oracle の PITR に関する考慮事項

RDS Custom for Oracle では、PITR は Amazon RDS の PITR と以下の重要な点で異なります。

- 復元されたデータベースの名前は、出典 DB インスタンスと同じです。必要に応じて、別の名前を指定できます。デフォルトは `ORCL` です。
- `AWSRDSCustomIamRolePolicy` は新しいアクセス許可を必要とします。詳細については、「ステップ 2: AWSRDSCustomInstanceRoleForRdsCustomInstance にアクセスポリシーを追加します。」を参照してください。

- すべての RDS Custom for Oracle DB インスタンスのバックアップ保持は、ゼロ以外の値に設定されている必要があります。
- OSまたは DB インスタンスのタイムゾーンを変更すると、PITR が機能しない可能性があります。タイムゾーンの変更については、「Oracle のタイムゾーン」を参照してください。
- オートメーションを ALL_PAUSED に設定すると、RDS Custom は最新の復元可能時間 (LRT) の前に作成されたログを含む、アーカイブされた REDO ログファイルのアップロードを一時停止します。オートメーションは短期間、一時停止することをお勧めします。

  説明のために、LRT が 10 分前だと仮定します。オートメーションを一時停止します。一時停止中は、RDS Custom はアーカイブされた REDO ログをアップロードしません。DB インスタンスがクラッシュした場合、一時停止時に存在していた LRT 以前の時間までしか回復できません。オートメーションを再開すると、RDS Custom はログのアップロードを再開します。LRT は進みます。通常の PITR のルールが適用されます。
- RDS Custom では、アップロード後に RDS Custom がアーカイブした REDO ログを削除する前に、アーカイブされた REDO ログを保持する時間数をマニュアルで指定できます。時間数を次のように指定します。

  1. /opt/aws/rdscustomagent/config/redo_logs_custom_configuration.json という名前のテキストファイルを作成します。
  2. 以下の形式の JSON オブジェクトを追加します: {"archivedLogRetentionHours" : "*num_of_hours*"}。数値は 1 ~ 840 の範囲の整数である必要があります。
- 非 CDB を PDB としてコンテナデータベース (CDB) に接続し、PITR を試みると仮定します。このオペレーションは、以前に PDB をバックアップしている場合のみ成功します。PDB を作成または変更した後は、常にバックアップすることをお勧めします。
- データベースの初期化パラメータはカスタマイズしないことをお勧めします。例えば、次のパラメータを変更すると PITR に影響します。
  - CONTROL_FILE_RECORD_KEEP_TIMEは、ログのアップロードと削除のルールに影響します。
  - LOG_ARCHIVE_DEST_*n*では、複数の復元先をサポートしていません。
  - ARCHIVE_LAG_TARGET は最新の復元可能時間に影響します。目標復旧時点 (RPO) が 5 分であるため、ARCHIVE_LAG_TARGET は 300 に設定されています。この目標を達成するために、RDS はオンラインの REDO ログを 5 分ごとに切り替えて、Amazon S3 バケットに保存します。ログの切り替え頻度が RDS Custom for Oracle データベースのパフォーマンス上の問題を引き起こす場合は、DB インスタンスとストレージを IOPS およびスループットの高いものにスケーリングできます。リカバリプランに必要な場合は、ARCHIVE_LAG_TARGET 初期化パラメータの設定を 60 ~ 7200 の値に調整できます。

- データベース初期化パラメータをカスタマイズする場合は、以下のカスタマイズのみを強くお勧めします。
  - COMPATIBLE
  - MAX_STRING_SIZE
  - DB_FILES
  - UNDO_TABLESPACE
  - ENABLE_PLUGGABLE_DATABASE
  - CONTROL_FILES
  - AUDIT_TRAIL
  - AUDIT_TRAIL_DEST

  その他の初期化パラメータについては、RDS Custom がデフォルト値を復元します。上記のリストにないパラメータを変更すると、PITR に悪影響を及ぼし、予期しない結果につながる可能性があります。例えば、CONTROL_FILE_RECORD_KEEP_TIMEはログのアップロードと削除のルールに影響します。

AWS マネジメントコンソール、AWS CLI、またはRDS API を使用して、 RDS Custom DB インスタンスをあるポイントに復元することができます。

コンソール

特定の時点に RDS Custom DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「自動バックアップ」を選択します。
3. 復元する RDS Custom DB インスタンスを選択します。
4. 「アクション」 で、「特定時点への復元」 を選択します。

   [特定時点への復元] ウィンドウが表示されます。
5. 「Latest restorable time」 を選択してできるだけ最新の時点に復元するか、「カスタム」 を選択して時刻を選択します。

   「カスタム」 を選択した場合、インスタンスクラスターを復元する日時を入力します。

時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。

6. 「DB インスタンス識別子」 に、ターゲットが復元された RDS Custom DB インスタンスの名前を入力します。名前は一意である必要があります。
7. 必要に応じて、DB インスタンスクラスなどの他のオプションを選択します。
8. [Restore to point in time] (特定時点への復元) を選択します。

AWS CLI

特定のポイントに DB インスタンスを復元するには、restore-db-instance-to-point-in-time AWS CLIコマンドを使用して、RDS Custom DB インスタンスを作成します。

次のいずれかのオプションを使用して、復元元のバックアップを指定します。

- `--source-db-instance-identifier` *`mysourcedbinstance`*
- `--source-dbi-resource-id` *`dbinstanceresourceID`*
- `--source-db-instance-automated-backups-arn` *`backupARN`*

`custom-iam-instance-profile` オプションは必須です。

次の例は、指定された時刻に `my-custom-db-instance` を `my-restored-custom-db-instance` という新しい DB インスタンスに復元します。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my-custom-db-instance\
    --target-db-instance-identifier my-restored-custom-db-instance \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
    --restore-time 2022-10-14T23:45:00.000Z
```

Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my-custom-db-instance ^
```

```
    --target-db-instance-identifier my-restored-custom-db-instance ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
    --restore-time 2022-10-14T23:45:00.000Z
```

RDS Custom for Oracle スナップショットの削除

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle で管理しているDBスナップショットは、不要になったら削除することができます。削除手順は、Amazon RDS および RDS Custom DB インスタンスのどちらでも同じです。

バイナリボリュームとルートボリュームの Amazon EBS スナップショットは、アカウントで実行されているいくつかのインスタンスや、他の RDS Custom for Oracle スナップショットにリンクされている可能性があるため、長期間アカウントに残ります。これらの EBS スナップショットは、既存の RDS Custom for Oracle リソース (DB インスタンスまたはバックアップ) に関連しなくなった後に自動的に削除されます。

コンソール

RDS Custom DB インスタンスのスナップショットを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. 削除する DB スナップショットを選択します。
4. 「アクション」 で、「スナップショットの削除」 を選択します。
5. 確認ページで、「削除」 を選択します。

AWS CLI

RDS Custom スナップショットを削除するには、AWS CLIコマンドdelete-db-snapshotを使用します。

以下のような必須オプションがあります。

- `--db-snapshot-identifier` - 削除するスナップショット

以下の例では、DB スナップショット `my-custom-snapshot` を削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot
```

Windows の場合:

```
aws rds delete-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for Oracle 自動バックアップの削除

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle の保持された自動バックアップは、不要になったら削除できます。手順は、Amazon RDS バックアップの削除と同じです。

コンソール

保持されている自動バックアップを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups] (自動バックアップ) を選択します。

3. [Retained] (保持) を選択します。
4. 削除する保持された自動バックアップを選択します。
5. 「アクション」 で、「削除」 を選択します。
6. 確認ページで、**`delete me`** を入力し、[Delete] (削除) を選択します。

AWS CLI

AWS CLI コマンド delete-db-instance-automated-backup を使用して、保持されている自動バックアップを削除できます。

保持されている自動バックアップを削除するには、以下のオプションを使用します。

- `--dbi-resource-id` - 出典 RDS Custom DB インスタンスのリソース識別子です。

  AWS CLI コマンド describe-db-instance-automated-backups を使用すると、保持された自動バックアップの出典 DB インスタンスのリソース識別子を見つけることができます。

次の例では、ソース DB インスタンスのリソース識別子 `custom-db-123ABCEXAMPLE` を持つ保持された自動バックアップを削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Windows の場合:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

RDS Custom for Oracle のオプショングループ使用する

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは

RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom は、オプショングループを使用して追加の機能の有効化と設定を行います。オプショングループには、RDS Custom for Oracle DB インスタンスに使用できるオプションという機能を指定できます。オプションの設定では、そのオプションの動作を指定できます。RDS Custom for Oracle DB インスタンスをオプショングループに関連付けると、指定したオプションとそれらの設定がそのインスタンスに対して有効になります。Amazon RDS のオプショングループに関する一般的な情報については、「オプショングループを使用する」を参照してください。

トピック

- RDS Custom for Oracle のオプショングループの概要
- Oracle のタイムゾーン

RDS Custom for Oracle のオプショングループの概要

Oracle Database のオプションを有効にするには、オプショングループにオプションを追加してから、そのオプショングループを DB インスタンスに関連付けます。詳細については、「オプショングループを使用する」を参照してください。

トピック

- RDS Custom for Oracle オプションの概要
- RDS Custom for Oracle DB インスタンスにオプションを追加する基本的な手順
- RDS Custom for Oracle でのオプショングループの作成
- オプショングループを RDS Custom for Oracle DB インスタンスに関連付ける

RDS Custom for Oracle オプションの概要

RDS Custom for Oracle は、DB インスタンスに対する以下のオプションをサポートします。

オプション	オプション ID	説明
Oracle のタイムゾーン	`Timezone`	RDS Custom for Oracle DB インスタンスで使用するタイムゾーン

RDS Custom for Oracle DB インスタンスにオプションを追加する基本的な手順

RDS Custom for Oracle DB インスタンスにオプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループにオプションを追加します。
3. DB インスタンスの作成または変更時に、オプショングループを DB インスタンスに関連付けます。

RDS Custom for Oracle でのオプショングループの作成

デフォルトオプショングループからその設定を引き継いで新しいオプショングループを作成することはできます。次に 1 つまたは複数のオプションを新しいオプショングループに追加します。あるいは、既存のオプショングループがある場合は、そのオプショングループをすべてのオプションと共に新しいオプショングループにコピーできます。オプショングループをコピーする方法については、「オプショングループをコピーする」を参照してください。

RDS Custom for Oracle のデフォルトのオプショングループは以下のとおりです。

- `default:custom-oracle-ee`
- `default:custom-oracle-se2`
- `default:custom-oracle-ee-cdb`
- `default:custom-oracle-se2-cdb`

オプショングループを作成すると、デフォルトのオプショングループから設定が引き継がれます。`TIME_ZONE` オプションを追加したら、DB インスタンスにオプショングループを関連付けることができます。

コンソール

オプショングループの作成方法の 1 つとして、AWS マネジメントコンソール を使用する方法があります。

新しいオプショングループをコンソールを使用して作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create option group] (オプショングループを作成) ウィンドウで次の操作を行います。
   a. [Name] (名前) には、AWS アカウント内で一意であるオプショングループの名前を入力します。名前には、英字、数字、ハイフンのみ使用できます。
   b. [Description] (説明) には、オプショングループの簡単な説明を入力します。説明は表示用に使用されます。
   c. [エンジン] で、次のいずれかの RDS Custom for Oracle DB エンジンを選択します。
      - custom-oracle-ee
      - custom-oracle-se2
      - custom-oracle-ee-cdb
      - custom-oracle-se2-cdb
   d. [メジャーエンジンバージョン] で、RDS Custom for Oracle でサポートされているメジャーエンジンバージョンを選択します。詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。
5. 続行するには、[Yes, Create(はい、作成する)] を選択します。逆に、操作をキャンセルするには、[Cancel] を選択します。

AWS CLI

オプショングループを作成するには、以下の必須パラメータを指定して AWS CLI の `create-option-group` コマンドを使用します。

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

以下の例では、testoptiongroup という名前のオプショングループを作成し、Oracle Enterprise Edition DB エンジンに関連付けています。説明は引用符で囲みます。

Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name testoptiongroup \
    --engine-name custom-oracle-ee-cdb \
    --major-engine-version 19 \
    --option-group-description "Test option group for a Custom Oracle CDB"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name testoptiongroup ^
    --engine-name custom-oracle-ee-cdb ^
    --major-engine-version 19 ^
    --option-group-description "Test option group for a Custom Oracle CDB"
```

RDS API

オプショングループを作成するには、Amazon RDS API の CreateOptionGroup オペレーションを呼び出します。

オプショングループを RDS Custom for Oracle DB インスタンスに関連付ける

オプショングループを新規または既存の DB インスタンスに関連付けることができます。

- 新しい DB インスタンスの場合は、インスタンスを作成するときにオプショングループを適用します。詳細については、「RDS Custom for Oracle DB インスタンスを作成する」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「RDS Custom for Oracle DB インスタンスを変更する」を参照してください。

Oracle のタイムゾーン

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

タイムゾーンオプションを使用して、RDS Custom for Oracle DB インスタンスで使用するシステムのタイムゾーンを変更することができます。例えば、オンプレミス環境またはレガシーアプリケーションとの互換性があるように、DB インスタンスのタイムゾーンで変更が必要になることがあります。タイムゾーンオプションでは、ホストレベルでタイムゾーンが変更されます。タイムゾーンを変更すると、`SYSDATE` や `SYSTIMESTAMP` など、すべての日付列および値に影響を与えます。

トピック

- RDS Custom for Oracle のタイムゾーンオプション設定
- RDS Custom for Oracle で利用可能なタイムゾーン
- RDS Custom for Oracle のタイムゾーンの設定に関する考慮事項
- RDS Custom for Oracle のタイムゾーン設定の制限事項
- オプショングループにタイムゾーンオプションを追加する
- タイムゾーンオプションの削除

RDS Custom for Oracle のタイムゾーンオプション設定

Amazon RDS は、タイムゾーンオプションの次の設定をサポートします。

オプション設定	有効な値	説明
`TIME_ZONE`	利用可能なタイムゾーンの例 利用できるタイムゾーンの一覧については、「RDS Custom for Oracle で利用可能なタイムゾーン」を参照してください。	DB インスタンスの新しいタイムゾーン。

RDS Custom for Oracle で利用可能なタイムゾーン

タイムゾーンオプションには、以下の値を使用できます。

ゾーン	タイムゾーン
アフリカ	Africa/Cairo、Africa/Casablanca、Africa/Harare、Africa/Lagos、Africa/Luanda、Africa/Monrovia、Africa/Nairobi、Africa/Tripoli、Africa/Windhoek
南北アメリカ大陸	America/Araguaina、America/Argentina/Buenos_Aires、America/Asuncion、America/Bogota、America/Caracas、America/Chicago、America/Chihuahua、America/Cuiaba、America/Denver、America/Detroit、America/Fortaleza、America/Godthab、America/Guatemala、America/Halifax、America/Lima、America/Los_Angeles、America/Manaus、America/Matamoros、America/Mexico_City、America/Monterrey、America/Montevideo、America/New_York、America/Phoenix、America/Santiago、America/Sao_Paulo、America/Tijuana、America/Toronto
アジア	Asia/Amman、Asia/Ashgabat、Asia/Baghdad、Asia/Baku、Asia/Bangkok、Asia/Beirut、Asia/Calcutta、Asia/Damascus、Asia/Dhaka、Asia/Hong_Kong、Asia/Irkutsk、Asia/Jakarta、Asia/Jerusalem、Asia/Kabul、Asia/Karachi、Asia/Kathmandu、Asia/Kolkata、Asia/Krasnoyarsk、Asia/Magadan、Asia/Manila、Asia/Muscat、Asia/Novosibirsk、Asia/Rangoon、Asia/Riyadh、Asia/Seoul、Asia/Shanghai、Asia/Singapore、Asia/Taipei、Asia/Tehran、Asia/Tokyo、Asia/Ulaanbaatar、Asia/Vladivostok、Asia/Yakutsk、Asia/Yerevan
大西洋	Atlantic/Azores、Atlantic/Cape_Verde
オーストラリア	Australia/Adelaide、Australia/Brisbane、Australia/Darwin、Australia/Eucla、Australia/Hobart、Australia/Lord_Howe、Australia/Perth、Australia/Sydney
ブラジル	Brazil/DeNoronha、Brazil/East
カナダ	Canada/Newfoundland、Canada/Saskatchewan
ETC	Etc/GMT-3

ゾーン	タイムゾーン
欧州	Europe/Amsterdam、Europe/Athens、Europe/Berlin、Europe/Dublin、Europe/Helsinki、Europe/Kaliningrad、Europe/London、Europe/Madrid、Europe/Moscow、Europe/Paris、Europe/Prague、Europe/Rome、Europe/Sarajevo
太平洋	Pacific/Apia、Pacific/Auckland、Pacific/Chatham、Pacific/Fiji、Pacific/Guam、Pacific/Honolulu、Pacific/Kiritimati、Pacific/Marquesas、Pacific/Samoa、Pacific/Tongatapu、Pacific/Wake
米国	US/Alaska、US/Central、US/East-Indiana、US/Eastern、US/Pacific
UTC	UTC

RDS Custom for Oracle のタイムゾーンの設定に関する考慮事項

DB インスタンスのタイムゾーンを設定する場合は、次の点を考慮してください。

- タイムゾーンオプションを追加すると、DB インスタンスが自動的に再起動する際に短い停止が発生します。
- 誤ってタイムゾーンを設定した場合、DB インスタンスを以前のタイムゾーン設定に戻す必要があります。このため、インスタンスにタイムゾーンオプションを追加する前に、次のいずれかの方法を使用することを強くお勧めします。
  - RDS Custom for Oracle DB インスタンスでデフォルトのオプショングループを使用している場合は、DB インスタンスのスナップショットを作成します。詳細については、「RDS Custom for Oracle スナップショットの作成」を参照してください。
  - DB インスタンスで現在デフォルト以外のオプショングループを使用している場合は、DB インスタンスのスナップショットを作成し、タイムゾーンオプションで新しいオプショングループを作成します。
- `Timezone` オプションを適用した後、DB インスタンスを手動でバックアップすることを強くお勧めします。
- 本稼働 DB インスタンスに追加する前に、テスト DB インスタンスでタイムゾーンオプションをテストすることを強くお勧めします。タイムゾーンオプションを追加すると、システムの日付を使用して日付または時刻を追加するテーブルに問題が発生することがあります。データやアプリケーションを分析して、タイムゾーンの変更による影響を評価することをお勧めします。

RDS Custom for Oracle のタイムゾーン設定の制限事項

以下の制限事項に留意してください。

- サポートペリメーター外にホストを移動しないと、ホスト上で直接タイムゾーンを変更することはできません。データベースのタイムゾーンを変更するには、オプショングループを作成する必要があります。
- タイムゾーンオプションは永続オプションであるため (固定オプションではありません)、次の操作を行うことはできません。
  - オプショングループに追加したオプションを削除する。
  - オプションのタイムゾーン設定を別のタイムゾーンに変更する。
- RDS Custom for Oracle DB インスタンスに複数のオプショングループを関連付けることはできません。
- CDB 内の個々の PDB にタイムゾーンを設定することはできません。

オプショングループにタイムゾーンオプションを追加する

RDS Custom for Oracle のデフォルトのオプショングループは以下のとおりです。

- `default:custom-oracle-ee`
- `default:custom-oracle-se2`
- `default:custom-oracle-ee-cdb`
- `default:custom-oracle-se2-cdb`

オプショングループを作成すると、デフォルトのオプショングループから設定が引き継がれます。Amazon RDS のオプショングループに関する一般的な情報については、「オプショングループを使用する」を参照してください。

コンソール

オプショングループにタイムゾーンオプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 変更するオプショングループを選択し、[オプションの追加] を選択します。

4. [Add option(オプションの追加)] ウィンドウで、以下の操作を行います。

   a. [タイムゾーン] を選択します。

   b. [オプション設定] で、タイムゾーンを選択します。

   c. オプションの追加後すぐに、関連付けられているすべての RDS Custom for Oracle DB インスタンスに対してオプションを有効にするには、[すぐに適用] で [はい] を選択します。[いいえ] を選択した場合 (デフォルト)、オプションは次のメンテナンスウィンドウ中に、関連付けられている各 DB インスタンスに対して有効になります。

   d.

   > **Important**
   >
   > すでに 1 つ以上の DB インスタンスにアタッチされている既存のオプショングループにタイムゾーンオプションを追加すると、すべての DB インスタンスが自動的に再起動される間に短い停止が発生します。

5. 設定が希望どおりになったら、[オプションの追加] を選択します。

6. タイムゾーンが更新された RDS Custom for Oracle DB インスタンスをバックアップします。詳細については、「RDS Custom for Oracle スナップショットの作成」を参照してください。

AWS CLI

以下の例では、AWS CLI の add-option-to-option-group コマンドを使用して、`Timezone` オプションと `TIME_ZONE` オプションの設定をオプショングループ `testoptiongroup` に追加しています。タイムゾーンは `America/Los_Angeles` に設定されています。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name "testoptiongroup" \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name "testoptiongroup" ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=America/
Los_Angeles}]" ^
```

```
    --apply-immediately
```

タイムゾーンオプションの削除

タイムゾーンオプションは、固定オプションではなく永続オプションです。オプショングループに追加したオプションを削除することはできません。DB インスタンスから古いオプショングループの関連付けを解除するには:

1. 更新された Timezone オプションを使用して新しいオプショングループを作成します。
2. DB インスタンスを変更した場合は、新しいオプショングループと関連付けます。

オンプレミスデータベースの RDS Custom for Oracle への移行

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

オンプレミスの Oracle データベースを RDS Custom for Oracle に移行する前に、以下の要素を検討してください。

- アプリケーションが許容できるダウンタイムの量
- ソースデータベースのサイズ
- ネットワーク接続
- フォールバックプランの要件
- ソースとターゲットの Oracle データベースバージョンと DB インスタンスの OS タイプ
- AWS Database Migration Service、Oracle GoldenGate、またはサードパーティーのレプリケーションツールなど、利用可能なレプリケーションツール

これらの要素に基づいて、物理移行、論理移行、またはその組み合わせを選択できます。物理移行を選択した場合は、次のテクニックを使用できます。

RMAN 複製

アクティブなデータベース複製では、ソースデータベースのバックアップは必要ありません。データベースファイルをネットワーク経由で補助インスタンスにコピーすることにより、ライブソースデータベースを送信先ホストに複製します。RMAN DUPLICATE コマンドは、必要なファイルをイメージコピーまたはバックアップセットとしてコピーします。このテクニックの詳細については、AWS ブログ記事「RMAN の duplication を用いて Oracle database を Amazon RDS Custom に物理コピーで移行する方法」を参照してください。

Oracle Data Guard

このテクニックでは、プライマリオンプレミスデータベースをバックアップして、バックアップを Amazon S3 バケットにコピーします。次に、バックアップを RDS Custom for Oracle のスタンバイ DB インスタンスにコピーします。必要な設定を実行したら、プライマリデータベースを RDS Custom for Oracle スタンバイデータベースに手動で切り替えます。このテクニックの詳細については、AWS ブログ記事「Physical migration of Oracle databases to Amazon RDS Custom using Data Guard」(Data Guard を使用した Oracle データベースの Amazon RDS Custom への物理移行) を参照してください。

RDS for Oracle へのデータの論理的なインポートに関する一般的な情報については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。

Amazon RDS Custom for Oracle DB インスタンスのアップグレード

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS Custom DB インスタンスは、新しいカスタムエンジンバージョン (CEV) を使用するように変更してアップグレードできます。アップグレードに関する一般的な情報については、DB インスタンスのエンジンバージョンのアップグレード を参照してください。

トピック

- RDS for Oracle のアップグレードの概要

- RDS Custom for Oracle のアップグレードの要件
- RDS Custom for Oracle のデータベースアップグレードに関する考慮事項
- RDS Custom for Oracle の OS アップグレードに関する考慮事項
- RDS Custom Oracle DB インスタンスの有効な CEV アップグレードターゲットの表示
- RDS Custom for Oracle DB インスタンスのアップグレード
- RDS Custom DB インスタンスの保留中データベースアップグレードの表示
- RDS Custom for Oracle DB インスタンスのアップグレードエラーのトラブルシューティング

RDS for Oracle のアップグレードの概要

RDS Custom for Oracle では、新しい CEV を作成し、新しい CEV を使用するようにインスタンスを変更することで、Oracle データベースまたは DB インスタンスのオペレーティングシステム (OS) にパッチを適用できます。

トピック

- CEV アップグレードのオプション
- CEV を使用しないパッチ適用
- DB インスタンスに CEV をパッチする一般的な手順

CEV アップグレードのオプション

アップグレード用 CEV を作成する場合、以下の相互排他的なオプションを選択できます。

データベースのみ

DB インスタンスで現在使用されている Amazon マシンイメージ (AMI) を再利用しますが、別のデータベースバイナリを指定します。RDS Custom は新しいバイナリボリュームを割り当て、それを既存の Amazon EC2 インスタンスにアタッチします。RDS カスタムは、データベースボリューム全体を、ターゲットデータベースバージョンを使用する新しいボリュームに置き換えます。

OS のみ

DB インスタンスで現在使用されているデータベースバイナリを再利用しますが、別の AMI を指定します。RDS Custom は新しい Amazon EC2 インスタンスを割り当て、既存のバイナリボリュームを新しいインスタンスにアタッチします。既存のデータベースボリュームは保持されます。

OS とデータベースの両方をアップグレードする場合は、CEV を 2 回アップグレードする必要があります。OS をアップグレードしてからデータベースをアップグレードすることも、データベースをアップグレードしてから OS をアップグレードすることもできます。

> **Warning**
>
> OS にパッチを適用すると、ルートボリュームデータや既存の OS カスタマイズは失われます。したがって、インストール、または永続的なデータやファイルの保存のためにはルートボリュームを使用しないことを強くお勧めします。また、アップグレード前にデータをバックアップすることをお勧めします。

CEV を使用しないパッチ適用

CEV を使用して RDS Custom for Oracle DB インスタンスをアップグレードすることを強くお勧めします。RDS Custom for Oracle オートメーションは、パッチメタデータを DB インスタンスのデータベースバイナリと同期します。

特別な状況では、RDS Custom は OPatch ユーティリティを使用して、基盤となる Amazon EC2 インスタンスに直接「1 回限りの」データベースパッチを適用することをサポートしています。例えば、すぐに適用したいデータベースパッチがあるが、RDS Custom チームが CEV 機能をアップグレードしているため、遅延が発生している場合などです。手動でデータベースパッチを適用するには、次のステップを実施します。

1. RDS Custom オートメーションを一時停止します。
2. Amazon EC2 インスタンスのデータベースバイナリにパッチを適用します。
3. RDS Custom オートメーションを再開します。

前述の方法の欠点は、アップグレードするすべてのインスタンスに手動でデータベースパッチを適用する必要があることです。一方、新しい CEV を作成すると、同じ CEV を使用して複数の DB インスタンスを作成またはアップグレードできます。

DB インスタンスに CEV をパッチする一般的な手順

OS またはデータベースのいずれにパッチを適用する場合でも、次の基本的な手順を実行します。

1. データベースまたは OS にパッチするかに応じて、次のいずれかを含む CEV を作成します。
   - DB インスタンスに適用する Oracle データベース RU

- 別の AMI (入手可能な最新のものまたはユーザーが指定したもの) と、ソースとして使用する既存の CEV

「CEV の作成」のステップを実行してください。

2. (データベースパッチのオプション) `describe-db-engine-versions` を実行して、利用可能なエンジンバージョンのアップグレードを確認します。
3. `modify-db-instance` を実行してパッチ適用プロセスを開始します。

   パッチが適用されているインスタンスのステータスは次のように異なります。

   - RDS がデータベースにパッチを適用している間、DB インスタンスのステータスは アップグレード中に変わります。
   - RDS が OS にパッチを適用している間、DB インスタンスのステータスは 変更中に変わります。

   DB インスタンスのステータスが利用可能 になると、パッチ適用は完了します。
4. `describe-db-instances` を実行して、DB インスタンスが新しい CEV を使用していることを確認します。

RDS Custom for Oracle のアップグレードの要件

RDS Custom for Oracle DB インスタンスをターゲット CEV にアップグレードする際に、必ず以下の前提条件を満たすようにしてください。

- アップグレード先のターゲット CEV が存在している必要があります。
- OS またはデータベースを 1 回の操作でアップグレードする必要があります。1 回の API 呼び出しで OS とデータベースの両方をアップグレードする操作はサポートされていません。
- ターゲット CEV は、現在の CEV のマニフェストにあるインストールパラメータ設定を使用する必要があります。例えば、デフォルトの Oracle ホームを使用するデータベースを、デフォルトではない Oracle ホーム を使用する CEV にアップグレードすることはできません。
- データベースをアップグレードする場合、ターゲット CEV は、新しいメジャーバージョンではなく、新しいマイナーデータベースバージョンを使用する必要があります。例えば、Oracle Database 12c CEV を Oracle Database 19c CEV にアップグレードすることはできません。ただし、バージョン 21.0.0.0.ru-2023-04.rur-2023-04.r1 からバージョン 21.0.0.0.ru-2023-07.rur-2023-07.r1 にはアップグレードできます。
- OS をアップグレードする場合、ターゲット CEV は異なる AMI を使用する必要がありますが、メジャーバージョンは同じです。

RDS Custom for Oracle のデータベースアップグレードに関する考慮事項

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

データベースをアップグレードする場合は、以下を考慮してください。

- 現在サポートされているオペレーティングシステム (OS) のバージョンは Oracle Linux 8 です。RDS Custom for Oracle から最新のセキュリティアップデートとパッチを引き続き受信するには、この OS に基づいて CEV を指定して DB インスタンスを Oracle Linux 8 にアップグレードします。Oracle Linux 8 をサポートするリリースは、Oracle Database 12c Release 1 (12.1)、Oracle Database Release 2 (12.2)、Oracle Database 19c のみです。最新の Oracle Linux 8 AMI に移行するには、OS を最新の AMI にアップグレードします。詳細については、「RDS Custom for Oracle DB インスタンスのアップグレード」を参照してください。

  Oracle Linux 7.9 は 2024 年 12 月 31 日にサポートを終了しました。サポート終了後も Oracle Linux 7 を引き続き実行するには、Oracle 延長サポートライセンスを購入してください。セキュリティ更新にはユーザー自身が責任を持ち、RDS Custom for Oracle インスタンスに手動でパッチを適用する必要があります。詳細については、「Lifetime Support Policy: Coverage for Oracle Open Source Service Offerings」を参照してください。
- プライマリ DB インスタンスでデータベースバイナリをアップグレードすると、RDS Custom for Oracle がリードレプリカを自動的にアップグレードします。ただし、OS をアップグレードする場合は、リードレプリカを手動でアップグレードする必要があります。
- コンテナデータベース (CDB) を新しいデータベースバージョンにアップグレードすると、RDS Custom for Oracle では、すべての PDB が開いているか、または開くことができるかを確認します。これらの条件が満たされない場合、RDS Custom はチェックを中止し、アップグレードを試行せずにデータベースを元の状態に戻します。条件が満たされると、RDS Custom は最初に CDB ルートにパッチを適用し、次に他のすべての PDB (`PDB$SEED` を含む) に並行してパッチを適用します。

  パッチ適用が完了すると、RDS Custom はすべての PDB を開こうとします。PDB を開くことができない場合は、`The following PDBs failed to open: list-of-PDBs` のイベントが表

示されます。RDS Custom が CDB ルートまたは PDB にパッチを適用できなかった場合、インスタンスは PATCH_DB_FAILED の状態になります。

- メジャーデータベースバージョンのアップグレードと非 CDB から CDB への変換を同時に行う場合があります。この場合は、以下の前提条件を満たしていることを確認します。
  1. Oracle マルチテナントアーキテクチャを使用する新しい RDS Custom for Oracle DB インスタンスを作成します。
  2. 非 CDB を CDB ルートに接続し、PDB として作成します。非 CDB が CDB と同じメジャーバージョンであることを確認してください。
  3. noncdb_to_pdb.sql Oracle SQL スクリプトを実行して、PDB を変換します。
  4. CDB インスタンスを検証します。
  5. インスタンスタイプをアップグレードします。

RDS Custom for Oracle の OS アップグレードに関する考慮事項

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

OS アップグレードを計画する際は、以下を考慮してください。

- RDS Custom for Oracle CEV で使用する独自の AMI を提供することはできません。指定できるのは、Oracle Linux 8 を使用するデフォルトの AMI、または RDS Custom for Oracle CEV で以前に使用したことのある AMI だけです。

  > **Note**
  >
  > 共通脆弱性とエクスポージャーが検出されると、RDS Custom for Oracle は新しいデフォルト AMI をリリースします。一定のスケジュールや保証はありません。RDS Custom for Oracle は、30 日ごとに新しいデフォルト AMI を公開する傾向があります。

- プライマリ DB インスタンスの OS をアップグレードするときは、関連するリードレプリカを手動でアップグレードする必要があります。

- OS にパッチ適用を開始する前に、AZ のインスタンスタイプに十分な Amazon EC2 コンピューティング容量を確保してください。

  キャパシティ予約を作成するときは、AZ、インスタンス数、インスタンス属性 (インスタンスタイプを含む) を指定します。例えば、DB インスタンスが基盤となる EC2 インスタンスタイプ r5.large を使用している場合は、AZ 内の r5.large の EC2 容量を予約することをお勧めします。OS のパッチ適用中、RDS Custom は db.r5.large タイプの新しいホストを 1 つ作成しますが、AZ にこのインスタンスタイプの EC2 キャパシティが不足していると、そのホストが停止する可能性があります。EC2 キャパシティを予約すれば、キャパシティの制約によってパッチ適用がブロックされるリスクが低くなります。詳細については、「Amazon EC2 ユーザーガイド」の「オンデマンドキャパシティーの予約」を参照してください。
- OS をアップグレードする前に、DB インスタンスをバックアップします。アップグレードすると、ルートボリュームデータと既存の OS カスタマイズがすべて削除されます。
- 責任共有モデルでは、OS を最新の状態に保つ責任はユーザーにあります。RDS Custom for Oracle では、OS にどのパッチを適用するかは義務付けていません。RDS Custom for Oracle が機能している場合は、この CEV に関連付けられた AMI を無期限に使用できます。

RDS Custom Oracle DB インスタンスの有効な CEV アップグレードターゲットの表示

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

既存の CEV は、AWS マネジメントコンソールのカスタムエンジンバージョンのページで見ることができます。

また、以下の例のように、describe-db-engine-versions AWS CLI コマンドを使用して、DB インスタンスをアップグレードする際に使用する有効な CEV を見つけることができます (例を参照)。この例では、エンジンバージョン `19.my_cev1` を使用して DB インスタンスを作成し、アップグレードバージョン `19.my_cev2` と `19.my_cev` が存在することを前提としています。

```
aws rds describe-db-engine-versions --engine custom-oracle-ee --engine-version
 19.my_cev1
```

出力は以下のようになります。ImageId フィールドには AMI ID が表示されます。

```
{
    "DBEngineVersions": [
        {
            "Engine": "custom-oracle-ee",
            "EngineVersion": "19.my_cev1",
            ...
            "Image": {
                "ImageId": "ami-2345",
                "Status": "active"
            },
            "DBEngineVersionArn": "arn:aws:rds:us-west-2:123456789012:cev:custom-
oracle-ee/19.my_cev1/12a34b5c-67d8-90e1-2f34-gh56ijk78lm9"
            "ValidUpgradeTarget": [
                {
                    "Engine": "custom-oracle-ee",
                    "EngineVersion": "19.my_cev2",
                    "Description": "19.my_cev2 description",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "custom-oracle-ee",
                    "EngineVersion": "19.my_cev3",
                    "Description": "19.my_cev3 description",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                }
            ]
            ...
```

RDS Custom for Oracle DB インスタンスのアップグレード

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは

RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle DB インスタンスをアップグレードするには、新しい CEV を使用するように変更します。この CEV には、新しいデータベースバイナリまたは新しい AMI を含めることができます。例えば、Oracle Linux 7.9 DB インスタンスを Oracle Linux 8 にアップグレードするには、Oracle Linux 8 を使用する最新の AMI を指定します。データベースと OS をアップグレードする場合は、2 つのアップグレードを別々に実行する必要があります。

> **Note**
>
> データベースをアップグレードする場合、RDS Custom はプライマリ DB インスタンスをアップグレードした後にリードレプリカを自動的にアップグレードします。OS をアップグレードする際は、レプリカを手動でアップグレードする必要があります。

始める前に、RDS Custom for Oracle のアップグレードの要件 と RDS Custom for Oracle のデータベースアップグレードに関する考慮事項 を確認してください。

コンソール

RDS Custom Oracle DB インスタンスをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、アップグレードする RDS Custom for SQL Server DB インスタンスを選択します。
3. [Modify] (変更) を選択します。Modify DB instance ページが表示されます。
4. DB エンジンバージョンの場合は、新しい CEV を選択してください。以下の操作を実行します。

   - データベースにパッチを適用する場合は、CEV が DB インスタンスで使用されているものとは異なるデータベースバイナリを指定し、指定した AMI が DB インスタンスが現在使用している AMI と違っていないことを確認してください。
   - OS にパッチを適用する場合は、CEV が DB インスタンスで使用されている AMI とは異なる AMI を指定し、異なるデータベースバイナリを指定していないことを確認してください。

> **Warning**
>
> OS にパッチを適用すると、ルートボリュームデータや既存の OS カスタマイズは失われます。

5. 継続する を選択して、変更の概要をチェックします。

   変更をすぐに反映させるには、すぐに申し込む を選択します。

6. 変更が正しい場合は、[Modify DB Instance] (DB インスタンスを変更) を選択します。または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

AWS CLI

以下の例は、考えられるアップグレードシナリオを示しています。この例では、RDS Custom for Oracle DB インスタンスを次の特性で作成したという前提です。

- `my-custom-instance` という DB インスタンス名
- `19.my_cev1` という名前の CEV
- CDB 以外のアーキテクチャを使用した Oracle Database 19c
- AMI `ami-1234` を使用する Oracle Linux 8

サービスが提供している最新の AMI は `ami-2345` です。CLI コマンド `describe-db-engine-versions` を実行して、エンドポイントを見つけることができます。

トピック

- OS のアップグレード
- データベースのアップグレード

OS のアップグレード

この例では、`ami-1234` を、サービスが提供している最新の AMIである `ami-2345` にアップグレードしたいと考えています。OS をアップグレードするため、データベースバイナリ `ami-1234` および `ami-2345` は同じである必要があります。`19.my_cev1` に基づいて `19.my_cev2` という名前の CEV を新規作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev2 \
    --description "Non-CDB CEV based on ami-2345" \
    --kms-key-id key-name \
    --source-custom-db-engine-version-identifer arn:aws:rds:us-
west-2:123456789012:cev:custom-oracle-ee/19.my_cev1/12345678-ab12-1234-cde1-
abcde123456789 \
    --image-id ami-2345
```

Windows の場合:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev2 ^
    --description "Non-CDB CEV based on ami-2345" ^
    --kms-key-id key-name ^
    --source-custom-db-engine-version-identifer arn:aws:rds:us-
west-2:123456789012:cev:custom-oracle-ee/19.my_cev1/12345678-ab12-1234-cde1-
abcde123456789 ^
    --image-id ami-2345
```

RDS Custom DB インスタンスをアップグレードするには、modify-db-instance AWS CLIコマンドを次のパラメータを指定します。

- `--db-instance-identifier` — アップグレードする RDS Custom for Oracle DB インスタンスを指定します。
- `--engine-version` — 新しい AMI が搭載されている CEV を指定します。
- `--no-apply-immediately|--apply-immediately` - アップグレードをすぐに実行するか、あるいはスケジュールされたメンテナンスウィンドウまで待つかを指定します。

次の例では、`my-custom-instance`バージョンを`19.my_cev2`にアップグレードします。OS のみがアップグレードされます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --engine-version 19.my_cev2 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --engine-version 19.my_cev2 ^
    --apply-immediately
```

データベースのアップグレード

この例では、RDS for Oracle DB インスタンスに Oracle パッチ p35042068 を適用します。OS のアップグレード で OS をアップグレードしたため、DB インスタンスでは現在 19.my_cev2 を使用していますが、これは ami-2345 に基づいています。また ami-2345 も使用する 19.my_cev3 という名前の新しい CEV を作成しますが、$MANIFEST 環境変数には新しい JSON マニフェストを指定します。そのため、新しい CEV とインスタンスが現在使用している CEV で異なるのは、データベースバイナリのみです。

Example

Linux、macOS、Unix の場合:

```
aws rds create-custom-db-engine-version \
    --engine custom-oracle-ee \
    --engine-version 19.my_cev3 \
    --description "Non-CDB CEV with p35042068 based on ami-2345" \
    --kms-key-id key-name \
    --image-id ami-2345 \
    --manifest $MANIFEST
```

Windows の場合:

```
aws rds create-custom-db-engine-version ^
    --engine custom-oracle-ee ^
    --engine-version 19.my_cev3 ^
    --description "Non-CDB CEV with p35042068 based on ami-2345" ^
    --kms-key-id key-name ^
    --image-id ami-2345 ^
```

```
    --manifest $MANIFEST
```

次の例では、my-custom-instance をエンジンバージョン19.my_cev3 にアップグレードします。データベースのみがアップグレードされます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --engine-version 19.my_cev3 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --engine-version 19.my_cev3 ^
    --apply-immediately
```

RDS Custom DB インスタンスの保留中データベースアップグレードの表示

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

Amazon RDS Custom DB インスタンスの保留中のデータベースアップグレードは、describe-db-instances または describe-pending-maintenance-actions AWS CLI コマンドを使用して見ることができます。

ただし、--apply-immediately オプションを使用したり、またはアップグレードが進行中である場合はこのアプローチは機能しません。

以下の describe-db-instances コマンドは、my-custom-instance の保留中のデータベースアップグレードを表示します。

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

出力は以下のようになります。

```
{
    "DBInstances": [
        {
           "DBInstanceIdentifier": "my-custom-instance",
            "EngineVersion": "19.my_cev1",
            ...
            "PendingModifiedValues": {
                "EngineVersion": "19.my_cev3"
            ...
            }
        }
    ]
}
```

RDS Custom for Oracle DB インスタンスのアップグレードエラーのトラブルシューティング

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom DB インスタンスのアップグレードに障害が発生すると、RDS イベントが生成され、DB インスタンスのステータスは `upgrade-failed` になります。

このステータスは、以下の例のように、describe-db-instances AWS CLIコマンドを使用して見ることができます。

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

出力は以下のようになります。

```
{
    "DBInstances": [
        {
           "DBInstanceIdentifier": "my-custom-instance",
            "EngineVersion": "19.my_cev1",
            ...
            "PendingModifiedValues": {
                "EngineVersion": "19.my_cev3"
            ...
            }
            "DBInstanceStatus": "upgrade-failed"
        }
    ]
}
```

アップグレードに障害が発生すると、次のタスクを実行するために DB インスタンスを変更する場合を除き、すべてのデータベースアクションがブロックされます。

- 同じアップグレードを再試行する
- RDS Custom オートメーションの一時停止と再開
- ポイントインタイムリカバリ (PITR)
- DB インスタンスを削除する

> **Note**
>
> RDS Custom DB インスタンスのオートメーションが一時停止されている場合、オートメーションを再開するまでアップグレードを再試行することはできません。
> RDS マネージドリードレプリカのアップグレードエラーには、プライマリと同じアクションが適用されます。

詳細については、「RDS Custom for Oracle のアップグレードに関するトラブルシューティング」を参照してください。

Amazon RDS Custom for Oracle の DB に関する問題のトラブルシューティング

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom の責任共有モデルは、OS シェルレベルのアクセスとデータベース管理者アクセスを提供します。RDS Custom は、システムアカウントでリソースを実行する Amazon RDS とは異なり、アカウント内でリソースを実行します。アクセスが増えるほど、責任も重くなります。以降のセクションで、Amazon RDS Custom DB インスタンスに関する問題のトラブルシューティング方法を学ぶことができます。

> **Note**
>
> このセクションでは RDS Custom for Oracle をトラブルシューティングする方法について説明します。RDS Custom for SQL Server のトラブルシューティングについては、「Amazon RDS Custom for SQL Server の DB に関する問題のトラブルシューティング」を参照してください。

トピック

- RDS Custom イベントの表示
- RDS Custom イベントへのサブスクライブ
- DB インスタンス作成に関する問題のトラブルシューティング
- RDS Custom for Oracle のカスタムエンジンバージョン作成のトラブルシューティング
- RDS Custom for Oracle でサポートされていない構成の修正
- RDS Custom for Oracle のアップグレードに関するトラブルシューティング
- RDS Custom for Oracle のレプリカプロモーションのトラブルシューティング

RDS Custom イベントの表示

イベントを表示する手順は、RDS Custom とAmazon RDS DB インスタンスでは同じです。詳細については、「Amazon RDS イベントの表示」を参照してください。

AWS CLIを使用してRDS Custom イベント通知を表示するには、describe-eventsコマンドを使用します。RDS Custom では、いくつかの新しいイベントを導入しています。イベントカテゴリは Amazon RDS の場合と同じです。イベントのリストについては、Amazon RDS イベントカテゴリとイベントメッセージを参照してください。

次の例では、指定した RDS Custom DB インスタンスで発生したイベントの詳細を取得します。

```
aws rds describe-events \
    --source-identifier my-custom-instance \
    --source-type db-instance
```

RDS Custom イベントへのサブスクライブ

イベント受信の手順は、RDS Custom とAmazon RDS DB インスタンスでは同じです。詳細については、「Amazon RDS イベント通知にサブスクライブする」を参照してください。

CLI を使用して RDS Custom イベント通知をサブスクライブするには、create-event-subscription コマンドを使用します。以下の必須パラメータを含めます。

- --subscription-name
- --sns-topic-arn

次の例では、現在のAWSアカウントのRDS Custom DB インスタンスのバックアップおよびリカバリイベントの受信を作成します。通知は、--sns-topic-arnで指定されたAmazon Simple Notification Service (Amazon SNS) のトピックに送信されます。

```
aws rds create-event-subscription \
    --subscription-name my-instance-events \
    --source-type db-instance \
    --event-categories '["backup","recovery"]' \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:interesting-events
```

DB インスタンス作成に関する問題のトラブルシューティング

環境が正しく設定されていないか、必要なアクセス許可がない場合、RDS Custom for Oracle DB インスタンスを作成または復元することはできません。DB インスタンスを作成または復元しようとすると、Amazon RDS は環境を検証し、問題を検出した場合に特定のエラーメッセージを返します。

すべての問題を解決したら、RDS Custom for Oracle DB インスタンスの作成または復元を再試行してください。

アクセス許可に関する一般的な問題

RDS Custom for Oracle インスタンスを作成または復元すると、Amazon RDS は環境に必要なアクセス許可があることを確認します。アクセス許可が欠落または拒否された場合、オペレーションは特定のエラーメッセージで失敗します。

問題タイプ	エラーメッセージ	アクション
IAM ロールアクセスポリシー	You can't create the DB instance because of incompatible resources. The host environment validation failed for the following permissions: <permission> on resource: <resource> due to permissions issue with message: User: <user> is not authorized to perform: <permission> on resource: <resource> because no identity-based policy allows the <permission> action.	リストされている必要なアクセス許可が存在し、適切なリソースが含まれているアクセスポリシーで `Allow` に設定されていることを確認します。
アクセス許可の境界	You can't create the DB instance because of incompatible resources. The host environment validation failed for the following permissions: <permission> on resource: <resource> due to permissions issue with message: User: <user> is not authorized to perform: <permission> on	インスタンスロールにアタッチされたアクセス許可の境界が、リストされている必要なアクセス許可とリソースを制限していないことを確認します。

問題タイプ	エラーメッセージ	アクション
	resource: <resource> with an explicit deny in a permissions boundary.	
サービスコントロールポリシー	You can't create the DB instance because of incompatible resources. The host environment validation failed for the following permissions: <permission> on resource: <resource> due to permissions issue with message: User: <user> is not authorized to perform: <permission> on resource: <resource> with an explicit deny in a service control policy.	AWS Organizations 管理者に連絡し、アカウントにアタッチされたサービスコントロールポリシーが、リストされている必要なアクセス許可とリソースを制限していないことを確認します。
リソースコントロールポリシー	You can't create the DB instance because of incompatible resources. The host environment validation failed for the following permissions: <permission> on resource: <resource> due to permissions issue with message: User: <user> is not authorized to perform: <permission> on resource: <resource> with an explicit deny in a resource control policy.	AWS Organizations 管理者に連絡し、アカウントにアタッチされたリソースコントロールポリシーが、リストされている必要なアクセス許可とリソースを制限していないことを確認します。
VPC エンドポイントポリシー	You can't create the DB instance because of incompatible resources. The host environment validation failed for the following permissions: <permission> on resource: <resource> due to permissions issue with message: User: <user> is not authorized to perform: <permission> on resource: <resource> with an explicit deny in a VPC endpoint policy.	必要な VPC エンドポイントが存在し、それらにアタッチされたポリシーが、リストされている必要なアクセス許可とリソースを制限していないことを確認します。

ネットワーキングの問題

ステップ 6: RDS Custom for Oracle 用に VPC を設定する の確認に加えて、以下が正しく設定され、必要な AWS のサービスへのアクセスが制限されていないことを確認します。

Amazon EC2 インスタンスにアタッチされているセキュリティグループ

セキュリティグループが RDS Custom オペレーションに必要なすべてのインバウンドトラフィックとアウトバウンドトラフィックを許可していることを確認します。

VPC にアタッチされたセキュリティグループ

VPC セキュリティグループが、必要な AWS のサービスとの間のトラフィックを許可していることを確認します。

VPC エンドポイント

必要なすべての VPC エンドポイントが正しく設定され、アクセス可能であることを確認します。

ネットワークアクセスコントロールリスト

ネットワーク ACL が RDS Custom 機能に必要なトラフィックをブロックしていないことを確認します。

RDS Custom for Oracle のカスタムエンジンバージョン作成のトラブルシューティング

CEV の作成に失敗すると、RDS Custom はメッセージ`Creation failed for custom engine version` *`major-engine-version.cev_name`* とともに`RDS-EVENT-0198`を発行し、これは障害に関する詳細を含んでいます。例えば、イベントは不足ファイルを出力します。

次の問題が原因で、CEV の作成に失敗することがあります。

- インストールファイルの入った Amazon S3 バケットが、CEV と同じ AWS リージョンにありません。
- AWS リージョンで初めて CEV 作成をリクエストすると、RDS Custom は RDS Custom リソース (CEV アーティファクト、AWS CloudTrailログ、トランザクションログなど) を保存するための S3 バケットを作成します。

RDS Custom が S3 バケットを作成できないと、CEV の作成は失敗します。「ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する」で説明されているように、発信者に S3 権限がないか、あるいは S3 バケットの数が制限に達しています。

- 発信者には、インストールメディアファイルが含まれている S3 バケットからファイルを取得する権限がありません。これらの権限については、ステップ 7: 必要な IAM アクセス許可を追加するを参照してください。
- IAM ポリシーには `aws:SourceIp` 条件があります。「AWS Identity and Access Management ユーザーガイド」の「AWS: 送信元 IP に基づいて AWS へのアクセスを拒否する」に記載されている推奨事項に、必ず従ってください。また呼び出し元に、「ステップ 5: IAM ユーザーまたはロールに必要なアクセス許可を付与する」で説明した S3 へのアクセス許可があることを確認します。
- CEV マニフェストに記載されているインストールメディアファイルが S3 バケットにありません。
- インストールファイルの SHA-256 チェックサムは RDS Custom で不明です。

  提供されたファイルの SHA-256 チェックサムが、Oracle Web サイトの SHA-256 チェックサムと一致していることを確認します。チェックサム が一致する場合は、失敗した CEV 名、ファイル名、およびチェックサムをAWSサポートに連絡してください。
- Opatch バージョンはパッチファイルとの互換性がありません。次のメッセージが表示される場合があります。`OPatch is lower than minimum required version. Check that the version meets the requirements for all patches, and try again`。Oracle パッチを適用するには、互換性のあるバージョンの OPatch ユーティリティを使用する必要があります。Opatch ユーティリティの必要なバージョンは、パッチの readme ファイルにあります。My Oracle Support から最新の OPatch ユーティリティをダウンロードし、CEV を再度作成してください。
- CEV マニフェストで指定されたパッチの順序が間違っています。

RDS イベントは、RDS コンソール (ナビゲーションペインでイベント) を選択) で、またはdescribe-events AWS CLIコマンドを使用して表示できます。デフォルトの期間は 60 分間です。イベントが返されない場合は、次の例のようにより長い期間を指定します。

```
aws rds describe-events --duration 360
```

現在、Amazon S3 からファイルをインポートして CEV を作成する MediaImport サービスは、AWS CloudTrailと統合されていません。そのため、CloudTrail で Amazon RDS のデータログを有効にする

と、CreateCustomDbEngineVersionイベントのようなMediaImport サービスの呼び出しはログに記録されません。

ただし、Amazon S3 バケットにアクセスする API Gateway からの呼び出しが表示される場合があります。これらの呼び出しは、CreateCustomDbEngineVersionイベントの MediaImport サービスから出されます。

RDS Custom for Oracle でサポートされていない構成の修正

共有責任モデルでは、RDS Custom for Oracle DB インスタンスを unsupported-configuration 状態にする設定上の問題は、お客様の責任で解決していただく必要があります。問題が AWS インフラストラクチャにある場合は、コンソールや AWS CLI を使用して修正できます。OS またはデータベースの設定に問題がある場合は、ホストにログインして修正できます。

> **Note**
>
> このセクションでは RDS Custom for Oracle でサポートされていない構成を修正する方法について説明します。RDS Custom for SQL Server の詳細については、「RDS Custom for SQL Server DB でサポートされていない構成の修正」を参照してください。

次の表では、サポートペリメーターが送信する通知とイベント、その修正方法について説明します。これらの通知とサポートペリメーターは変更されることがあります。サポート周辺の背景については、「RDS Custom サポート範囲」を参照してください。イベントの説明については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-00000	サポートされていない手動設定	The RDS Custom DB instance status is set to [Unsupported configuration] because of: *reason*.	この問題を解決するには、サポートケースを作成します。

AWS リソース (インフラストラクチャ)

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O1001	Amazon Elastic Block Store (Amazon EBS) ボリューム	The following EBS volumes were added to EC2 instance *ec2_id*: *volume_id* . To resolve the issue, detach the specified volumes from the instance.	RDS Custom は、Amazon マシンイメージ (AMI) から作成したルートボリュームのほか、2 種類の EBS ボリュームを作成して EC2 インスタンスに関連付けます。 • データベースソフトウェアバイナリが置かれているバイナリボリューム • データベースファイルが置かれているデータボリューム DB インスタンスを作成する際に指定したストレージ設定により、データボリュームが設定されます。 サポートペリメーターは、次のことをモニタリングします。 • DB インスタンスと共に作成された初期 EBS ボリュームは引き続き関連付けられています。 • 初期 EBS ボリュームは、初期に設定したストレージタイプ、サイズ、プロビジョンド IOPS、ストレージスループットと同じ設定です。 • DB インスタンスに追加の EBS ボリュームは添付されません。 次の CLI コマンドを使用して、EBS ボリュームの詳細と RDS Custom for Oracle インスタンスの詳細のボリュームタイプを比較します。 `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O1002	Amazon Elastic Block Store (Amazon EBS) ボリューム	EBS volume *volume_id* has been detached from EC2 instance [*ec2_id*]. You can't detach the original volume from this instance. To resolve the issue, re-attach *volume_id* to *ec2_id*.	RDS Custom は、Amazon マシンイメージ (AMI) から作成したルートボリュームのほか、2 種類の EBS ボリュームを作成して EC2 インスタンスに関連付けます。 • データベースソフトウェアバイナリが置かれているバイナリボリューム • データベースファイルが置かれているデータボリューム DB インスタンスを作成する際に指定したストレージ設定により、データボリュームが設定されます。 サポートペリメーターは、次のことをモニタリングします。 • DB インスタンスと共に作成された初期 EBS ボリュームは引き続き関連付けられています。 • 初期 EBS ボリュームは、初期に設定したストレージタイプ、サイズ、プロビジョンド IOPS、ストレージスループットと同じ設定です。 • DB インスタンスに追加の EBS ボリュームは添付されません。 次の CLI コマンドを使用して、EBS ボリュームの詳細と RDS Custom for Oracle インスタンスの詳細のボリュームタイプを比較します。 `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O1003	Amazon Elastic Block Store (Amazon EBS) ボリューム	The original EBS volume *volume_id* attached to EC2 instance *ec2_id* has been modified as follows: size [*X*] to [*Y*], type [*N*] to [*M*], or IOPS [*J*] to [*K*]. To resolve the issue, revert the modification.	RDS Custom は、Amazon マシンイメージ (AMI) から作成したルートボリュームのほか、2 種類の EBS ボリュームを作成して EC2 インスタンスに関連付けます。 • データベースソフトウェアバイナリが置かれているバイナリボリューム • データベースファイルが置かれているデータボリューム DB インスタンスを作成する際に指定したストレージ設定により、データボリュームが設定されます。 サポートペリメーターは、次のことをモニタリングします。 • DB インスタンスと共に作成された初期 EBS ボリュームは引き続き関連付けられています。 • 初期 EBS ボリュームは、初期に設定したストレージタイプ、サイズ、プロビジョンド IOPS、ストレージスループットと同じ設定です。 • DB インスタンスに追加の EBS ボリュームは添付されません。 次の CLI コマンドを使用して、EBS ボリュームの詳細と RDS Custom for Oracle インスタンスの詳細のボリュームタイプを比較します。 `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O1004	Amazon EC2 インスタンスの状態	Automated recovery left EC2 instance [*ec2_id*] in an impaired state. To resolve the issue, see Troubleshooting instance recovery failures.	DB インスタンスのステータスを確認するには、コンソールを使用するか、次の AWS CLI コマンドを実行します。 `aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* |grep DBInstanceStatus`
SP-O1005	Amazon EC2 インスタンス属性	EC2 instance [*ec2_id*] was modified as follows: attribute [*att1*] changed from [*val-old*] to [*val-new*], attribute [*att2*] changed from [*val-old*] to [*val-new*]. To resolve the issue, revert to the original value.	

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O1006	Amazon EC2 インスタンスの状態	EC2 instance [*ec2_id*] was terminated or can't be found. To resolve the issue, delete the RDS Custom DB instance.	サポートペリメーターは、 EC2 インスタンスの状態変更通知をモニタリングします。EC2 インスタンスは常に実行されている必要があります。 DB インスタンスを削除するには 1. DB インスタンスのステータスを確認するには、コンソールを使用するか、次の AWS CLI コマンドを実行します。 `aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* |grep DBInstanceStatus` 2. RDS Custom for Oracle DB インスタンスを削除します。
SP-O1007	Amazon EC2 インスタンスの状態	EC2 instance [*ec2_id*] was stopped. To resolve the issue, start the instance.	サポートペリメーターは、 EC2 インスタンスの状態変更通知をモニタリングします。EC2 インスタンスは常に実行されている必要があります。 DB インスタンスを再起動するには 1. DB インスタンスのステータスを確認するには、コンソールを使用するか、次の AWS CLI コマンドを実行します。 `aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* |grep DBInstanceStatus` 2. DB インスタンスを起動します。 3. バイナリおよびデータボリュームを再マウントします。

イベント ID	設定	RDS イベントメッセージ	アクション

SP-10	Amazon SQS のアクセス許可	Permissions are missing for Amazon SQS. Check the permissions for the IAM instance profile, VPC endpoint policy, and dependent service connections, and then try again.	これは、ホストに関連付けられた IAM プロファイルに次のアクセス許可があることを確認することで解決できます。
			`"SQS:SendMessage"`
			`"SQS:ReceiveMessage"`
			`"SQS:DeleteMessage"`
			`"SQS:GetQueueUrl"`

SP-10	Amazon Simple Queue Service (Amazon SQS)	The SQS queue [%s] was deleted and couldn't be recovered. この問題を解決するには、キューを再作成します。	Amazon SQS キュー を再作成します。

オペレーティングシステム

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O2001	RDS カスタムエージェントステータス	The RDS Custom agent isn't running on EC2 instance [*ec2_id*]. Make sure the agent	RDS Custom for Oracle では、RDS カスタムエージェントが停止すると、DB インスタンスはサポートペリメーターを外れます。エージェントは、30 秒ごとに`IamAlive`メトリクスを Amazon CloudWatch に公開します。メトリクスが 30 秒間公開されないと、アラームがトリガーされます。サポート境界は、1 分ご

イベントID	設定	RDS イベントメッセージ	アクション

		is running on [*ec2_id*].	とにホストの RDS Custom エージェントプロセスの状態をモニタリングします。
			
			RDS Custom エージェントを再起動するには
			
			1. ホストにログインし、RDS Custom エージェントが実行されていることを確認します。
			2. 次のコマンドを実行して、エージェントのステータスを確認します。
			
			`service rdscustomagent status`
			
			3. エージェントを開始するには、次のコマンドを実行します。
			
			`service rdscustomagent start`
			
			RDS Custom エージェントが再び実行されると、`IamAlive`メトリクスが Amazon CloudWatch に公開され、アラームが0K状態に切り替わります。この切り替えは、エージェントが実行中であることをサポートペリメーターに通知します。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-02002	AWS Systems Manager エージェント (SSM エージェント) ステータス	The Systems Manager agent on EC2 instance [*ec2_id*] is unreachable. Make sure that you that have correctly configured the network, agent, and IAM permissions.	SSM Agent は常に実行されている必要があります。RDS Custom エージェントは、Systems Manager Agent が実行されていることを確認するロールがあります。SSM Agent が終了して再起動した場合、RDS Custom エージェントはメトリクス `SSM_Agent_Restarted_Or_NotFound` を CloudWatch に発行します。RDS Custom エージェントには、過去 3 分ごとに再起動があった場合にトリガーするように設定されたメトリクス `do-not-delete-rds-custom-ssm-agent-restarted-or-notfound-` *ec2-id* のアラームがあります。サポート境界は、ホスト上の SSM Agent のプロセス状態も 30 分ごとにモニタリングします。 詳細については、「SSM Agent のトラブルシューティング」を参照してください。
SP-02003	AWS Systems Manager エージェント (SSM エージェント) ステータス	The Systems Manager agent on EC2 instance [*ec2_id*] crashed multiple times. For more information, see the SSM Agent troubleshooting documentation.	詳細については、「SSM Agent のトラブルシューティング」を参照してください。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O2004	OS のタイムゾーン	The time zone on EC2 instance [*ec2_id*] was changed. To resolve this issue, revert the timezone to the previous setting of [*previous-time-zone*]. Then use an RDS options group to change the time zone.	RDS オートメーションは、ホストのタイムゾーンがオプショングループを使用せずに変更されたことを検出しました。このホストレベルの変更により RDS オートメーションが失敗する可能性があるため、EC2 インスタンスは `unsupported-configuration` 状態になります。 タイムゾーン設定を修正するには 1. EC2 ホストにログインし、OS タイムゾーンを次のように確認します。 `timedatectl` 2. RDS Custom オートメーションを一時停止します。詳細については、「RDS Custom DB インスタンスの一時停止と再開」を参照してください。 3. DB インスタンスを停止します。 4. オペレーティングシステムのタイムゾーンの変更を元に戻します。 5. DB インスタンスを起動します。 6. RDS Custom オートメーションを再開します。 DB インスタンスは 30 分以内に利用可能になります。将来ペリメーターから外れないようにするため、オプショングループを使用してタイムゾーンを変更します。詳細については、「Oracle のタイムゾーン」を参照してください。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O2005	sudo 設定	The sudo configurations on EC2 instance [*ec2_id*] lack necessary permissions. To resolve this issue, revert the recent changes to the sudo configurations.	サポートペリメーターは、特定の OS ユーザーがホスト上で特定のコマンドを実行できることを検証します。sudo 設定をモニタリングし、サポートされている状態と比較します。 sudo の設定がサポートされていない場合、RDS Custom は以前のサポートされた状態に上書きして戻そうとします。上書きが成功すると、RDS Custom は次の通知を送信します。 RDS Custom が設定を上書きしました。 上書きが失敗した場合、DB インスタンスの設定はサポートされていない状態のままになります。この問題を解決するには、sudoers.d/ ファイル内の変更を元に戻すか、アクセス許可を修正します。 **sudo 設定の変更を調査するには** 1. ホストにログインします。 2. 以下のコマンドを実行してください。 `visudo -c -f /etc/sudoers.d/ individual_sudo_files` 3. 必要に応じて sudo 設定を変更します。 サポート境界が sudo 設定に対応していると判断すると、30 分以内に RDS Custom for Oracle DB インスタンスが利用できるようになります。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O2006	S3 バケットのアクセシビリティ	RDS Custom automation can't download files from the S3 bucket on EC2 instance [*ec2_id*]. Check your networking configuration and make sure the instance allows connections to and from S3.	
SP-20	高可用性ソフトウェアソリューションバージョン	The HA solution of your instance differs from the expected version. この問題を解決するには、AWS サポート ケースを作成します。	AWS サポート ケースを作成する。

データベース

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O3001	データベースアーカイブ遅延	The ARCHIVE_LAG_TARGET parameter on EC2 instance	サポート境界は、DB インスタンスの最新の復元可能時間が妥当な範囲内であることを確認するため、`ARCHIVE_LAG_TARGET` データベースパラメータをモニタリングします。

イベント ID	設定	RDS イベントメッセージ	アクション
	ターゲット	[*ec2_id*] is out of the recommended range *value_range* . To resolve the issue, set the parameter to a value within value_range.	アーカイブ REDO ログの遅延ターゲットを変更するには 1. EC2 ホストにログインします。 2. RDS Custom for Oracle DB インスタンスに接続します。 3. ARCHIVE_LAG_TARGET パラメータを 60～7200 の値に変更します。例えば、次の SQL ステートメントを使用します。 `ALTER SYSTEM SET ARCHIVE_LAG_TARGET=300 SCOPE=BOTH;` DB インスタンスは 30 分以内に利用可能になります。

イベント ID	設定	RDS イベントメッセージ	アクション

SP-O3002	Oracle Data Guard ロール	The database role [*role_name*] isn't supported for Oracle Data Guard on EC2 instance [*ec2_id*]. To resolve the issue, set the DATABASE_ROLE parameter to either PRIMARY or PHYSICAL STANDBY.	サポートペリメーターは、 15 秒ごとに現在のデータベースロールをモニタリングし、データベースロールが変更されると CloudWatch 通知を送信します。Oracle Data GuardDATABASE_ROLE パラメータはPRIMARYまたはPHYSICAL STANDBYのいずれかである必要があります。
			
			Oracle Data Guard データベースロールをサポートされる値に復元するには
			
			1. 次のステートメントを実行して、Oracle Data Guard ロールを確認します。
			
			`SELECT DATABASE_ROLE FROM V$DATABASE;`
			
			2. DB インスタンスがスタンドアロンの場合は、次のステートメントのいずれかを使用してインスタンスを PRIMARY ロールに戻します。
			
			`ALTER DATABASE COMMIT TO SWITCHOVER PRIMARY;`
			`ALTER DATABASE ACTIVATE STANDBY DATABASE;`
			
			DB インスタンスがレプリカの場合、次のステートメントを使用して PHYSICAL STANDBY ロールに戻します。
			
			`ALTER DATABASE CONVERT TO PHYSICAL STANDBY;`
			
			サポートペリメーターがデータベースロールがサポートされると判断すると、RDS Custom for Oracle DB インスタンスは 15 秒以内に利用可能になります。

イベント ID	設定	RDS イベントメッセージ	アクション
SP-O3003	データベースのヘルス	The SMON process of the Oracle database is in a zombie state. To resolve the issue, manually recover the database on EC2 instance [*ec2_id*], open the database, and then immediately back it up. For more help, contact サポート.	サポートペリメーターは DB インスタンスの状態をモニタリングします。また、前の 1 時間と前日に、何回再起動が発生したかをモニタリングします。インスタンスがまだ存在する状態にあるときに通知されますが、インスタンスと対話することはできません。 サポート境界にインスタンスの状態を評価させるには 1. ホストにログインし、データベースの状態を判断します。 ``` ps -eo pid,state,command | grep smon ``` 2. 必要に応じて、DB インスタンスを再起動します。再起動が失敗した場合は、次のステップに進みます。 3. 必要に応じて、EC2 ホストを再起動します。 DB インスタンスの再起動後、RDS Custom エージェントは DB インスタンスが無応答の状態ではなくなったことを検出します。次に、DB インスタンスの状態を再評価するようサポートペリメーターに通知します。

イベント ID	設定	RDS イベントメッセージ	アクション

SP-O3004	データベースログモード	The database log mode on EC2 instance [*ec2_id*] was changed to [*value_b*]. To resolve the issue, set the log mode to [*value_a*].	DB インスタンスのログモードを **ARCHIVELOG** に変更するには
			1. EC2 ホストにログインします。
			2. データベースに接続し、次のステートメントを実行します。
			`SELECT LOG_MODE FROM V$DATABASE;`
			または、以下のコマンドを SQL*Plus で実行します。
			`ARCHIVE LOG LIST`
			3. 次の SQL*Plus コマンドを実行して、一貫したシャットダウンを開始します。
			`SHUTDOWN IMMEDIATE`
			RDS Custom エージェントは DB インスタンスを再起動し、ログモードを ARCHIVELOG に設定します。DB インスタンスは 30 分以内に利用可能になります。

SP-O3005	Oracle ホームパス	The Oracle home on EC2 instance [*ec2_id*] was changed to *new_path*. To resolve the issue, revert the setting to *old_path*.	

イベント ID	設定	RDS イベントメッセージ	アクション

SP-O3006	データベースの一意の名前	The database unique name on EC2 instance [*ec2_id*] was changed to *new_value* . To resolve the issue, revert the name to *old_value* .	DB インスタンスのデータベースの一意の名前を変更するには
			1. EC2 ホストにログインします。
			2. データベースに接続し、次のステートメントを実行します。
			`SELECT DB_UNIQUE_NAME FROM V$DATABASE;`
			3. コマンド ALTER SYSTEM SET DB_UNIQUE_NAME を使用して、元のデータベースの一意の名前を指定します。
			4. 次の SQL ステートメントを実行して、一貫性のあるシャットダウンを開始します。
			`SHUTDOWN IMMEDIATE;`
			RDS Custom エージェントは DB インスタンスを再起動し、ログモードを ARCHIVELOG に設定します。DB インスタンスは 30 分以内に利用可能になります。

RDS Custom for Oracle のアップグレードに関するトラブルシューティング

RDS Custom for Oracle インスタンスのアップグレードに失敗する場合があります。以下では、RDS Custom DB for Oracle DB インスタンスのアップグレード時に使用できるテクニックを紹介します。

- DB インスタンスの `/tmp` ディレクトリにあるアップグレード出力ログファイルを確認します。ログの名前は、DB エンジンのバージョンによって異なります。例えば、`catupgrd` または `catup` という文字列を含むログが表示される場合があります。
- `/rdsdbdata/log/trace` ディレクトリにある `alert.log` ファイルを確認します。

- grep ディレクトリで以下の root コマンドを実行し、アップグレード OS のプロセスを追跡します。このコマンドは、ログファイルが書き込まれる場所を示し、アップグレードプロセスの状態を判断します。

```
ps -aux | grep upg
```

出力例を次に示します。

```
root     18884  0.0  0.0 235428  8172 ?        S<   17:03   0:00 /usr/bin/
sudo -u rdsdb /rdsdbbin/scripts/oracle-control ORCL op_apply_upgrade_sh RDS-
UPGRADE/2.upgrade.sh
rdsdb    18886  0.0  0.0 153968 12164 ?        S<   17:03   0:00 /usr/bin/perl -T -
w /rdsdbbin/scripts/oracle-control ORCL op_apply_upgrade_sh RDS-UPGRADE/2.upgrade.sh
rdsdb    18887  0.0  0.0 113196  3032 ?        S<   17:03   0:00 /bin/sh /rdsdbbin/
oracle/rdbms/admin/RDS-UPGRADE/2.upgrade.sh
rdsdb    18900  0.0  0.0 113196  1812 ?        S<   17:03   0:00 /bin/sh /rdsdbbin/
oracle/rdbms/admin/RDS-UPGRADE/2.upgrade.sh
rdsdb    18901  0.1  0.0 167652 20620 ?        S<   17:03   0:07 /rdsdbbin/oracle/
perl/bin/perl catctl.pl -n 4 -d /rdsdbbin/oracle/rdbms/admin -l /tmp catupgrd.sql
root     29944  0.0  0.0 112724  2316 pts/0    S+   18:43   0:00 grep --color=auto
 upg
```

- 以下の SQL クエリを実行してコンポーネントの現在の状態を確認し、DB インスタンスにインストールされているデータベースのバージョンとオプションを見つけます。

```
SET LINESIZE 180
COLUMN COMP_ID FORMAT A15
COLUMN COMP_NAME FORMAT A40 TRUNC
COLUMN STATUS FORMAT A15 TRUNC
SELECT COMP_ID, COMP_NAME, VERSION, STATUS FROM DBA_REGISTRY ORDER BY 1;
```

出力は以下のようになります。

```
COMP_NAME                                STATUS               PROCEDURE
---------------------------------------- --------------------
 ------------------------------------------------
Oracle Database Catalog Views            VALID
 DBMS_REGISTRY_SYS.VALIDATE_CATALOG
Oracle Database Packages and Types       VALID
 DBMS_REGISTRY_SYS.VALIDATE_CATPROC
Oracle Text                              VALID                VALIDATE_CONTEXT
```

```
Oracle XML Database                        VALID                DBMS_REGXDB.VALIDATEXDB

4 rows selected.
```

- 以下の SQL クエリを実行して、アップグレードプロセスに干渉する可能性のある無効なオブジェクトがないかチェックします。

```
SET PAGES 1000 LINES 2000
COL OBJECT FOR A40
SELECT SUBSTR(OWNER,1,12) OWNER,
       SUBSTR(OBJECT_NAME,1,30) OBJECT,
       SUBSTR(OBJECT_TYPE,1,30) TYPE, STATUS,
       CREATED
FROM   DBA_OBJECTS
WHERE  STATUS <>'VALID'
AND    OWNER IN ('SYS','SYSTEM','RDSADMIN','XDB');
```

RDS Custom for Oracle のレプリカプロモーションのトラブルシューティング

コンソール、`promote-read-replica` AWS CLI コマンド、または `PromoteReadReplica` API を使用して、RDS Custom for Oracle のマネージド Oracle レプリカを昇格させることができます。プライマリ DB インスタンスを削除し、すべてのレプリカが正常であれば、RDS Custom for Oracle はマネージドレプリカをスタンドアロンインスタンスに自動的に昇格させます。レプリカが自動化を一時停止しているか、サポートペリメーター外にある場合は、RDS Custom が自動的にレプリカを昇格させる前にレプリカを修正する必要があります。詳細については、「RDS Custom for Oracle レプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

以下の状況では、レプリカのプロモーションワークフローが停止することがあります。

- プライマリ DB インスタンスは、`STORAGE_FULL` 状態です。
- プライマリデータベースは、すべてのオンライン REDO ログをアーカイブできるわけではありません。
- Oracle レプリカとプライマリデータベースのアーカイブ REDO ログファイルの間にギャップが存在します。

停止したワークフローに対応するには

1. Oracle レプリカ DB インスタンスの REDO ログギャップを同期します。

2. 適用された最新の REDO ログにリードレプリカを強制的に昇格させます。SQL*Plus の次のコマンドを実行します。

```
ALTER DATABASE ACTIVATE STANDBY DATABASE;
SHUTDOWN IMMEDIATE
STARTUP
```

3. サポートに連絡し、DB インスタンスを `available` ステータスに移行してもらいます。

Amazon RDS Custom for Oracle の既知の問題

> **Note**
>
> サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。2027 年 3 月 31 日以降、RDS Custom for Oracle コンソールまたは RDS Custom for Oracle リソースにアクセスできなくなります。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。

RDS Custom for Oracle を使用する場合は、DB インスタンスの次の問題に注意してください。

- ルートボリュームまたは dbbin ボリュームのサイズ変更はサポートされていません。

  > **Warning**
  >
  > ルートボリュームまたは dbbin ボリュームのサイズを手動で変更しないことを強くお勧めします。すべての構成をデータボリュームに保存すればパッチ適用後も保持されます。また、ボリュームのサイズを変更する場合は必ず RDS スケールストレージ API 使用してください。

- データベースインスタンスが、Oracle Linux 7 を使用する AMI など、古い AMI にある場合、RDS API がブロックされる可能性があります。この問題を解決するには、OS パッチを使用して DB インスタンスに最新の AMI にパッチを適用します。詳細については、「CEV アップグレードのオプション」を参照してください。
- RDS オペレーションを実行する前に、AWS アカウント にコンピューティングとストレージに十分なクォータがあることを確認してください。
- データベースを作成している間に、データベースまたは Amazon EC2 のホストにアクティブにログインしてコマンドを実行すると、データベースの作成が完了しない可能性があります。

- リードレプリカの問題のため、コントロールファイルの多重化は現在サポートされていません。リードレプリカを作成する前に、ソースデータベースの CONTROL_FILES 初期化パラメータでファイル名を 1 つだけ指定してください。
- データベースモードを PHYSICAL STANDBY (マウントまたは読み取り専用) から SNAPSHOT STANDBY (読み取り/書き込みに変換) に変更することはできません。
- AWS アカウント がサービスコントロールポリシー (SCP) を持つ AWS Organization の一部であり、SCP に条件キーが含まれている場合、RDS Custom for Oracle DB インスタンスが次のエラーで作成されない場合があります。

```
You can't create the DB instance because of incompatible resources.
The IAM instance profile role [AWSRDSCustomInstanceRole1-us-east-1] is missing the
 following permissions:
EFFECT [Allow] on ACTION(S) [ssm:DescribeAssociation, ssm:DescribeDocument,
 ssm:GetConnectionStatus,
  ssm:GetDeployablePatchSnapshotForInstance, ssmmessages:OpenControlChannel,
 ssm:GetParameters,
  ssm:ListInstanceAssociations, ssm:PutConfigurePackageResult,
 ssmmessages:CreateControlChannel,
  ssm:GetParameter, ssm:UpdateAssociationStatus, ssm:GetManifest,
 ssmmessages:CreateDataChannel,
  ssm:PutInventory, ssm:UpdateInstanceInformation, ssm:DescribeInstanceInformation,
  ssmmessages:OpenDataChannel, ssm:GetDocument, ssm:ListAssociations,
 ssm:PutComplianceItems,
  ssm:UpdateInstanceAssociationStatus] for RESOURCE(S) [], EFFECT [Allow] on
 ACTION(S) [ec2messages:DeleteMessage,
  ec2messages:FailMessage, ec2messages:GetEndpoint, ec2messages:AcknowledgeMessage,
 ec2messages:GetMessages,
  ec2messages:SendReply] for RESOURCE(S) [], EFFECT [Allow] on ACTION(S)
 [logs:CreateLogStream,
  logs:DescribeLogStreams, logs:PutRetentionPolicy, logs:PutLogEvents]
```

この問題を解決するには、サポート でチケットを作成します。

データベースユーザーアカウントの既知の問題

次の問題に注意してください。

- RDSADMIN や RDS_DATAGUARD などの、文字列 RDS で始まるデータベースユーザーアカウントを削除しないでください。RDS Custom for Oracle は、自動化のために RDS アカウントを使用しま

す。このユーザーアカウントを削除すると、RDS Custom はインスタンスをサポートされていない設定状態に移行します。

- `ModifyDBInstance` API を使用して RDS Custom for Oracle DB インスタンスのマスターユーザー名を変更することはできません。
- RDS Custom for Oracle は、すべての DB インスタンスでユーザーアカウントの認証情報をローテーションします。詳細については、「コンプライアンスプログラムのための RDS Custom for Oracle の認証情報のローテーション」を参照してください。オンプレミスのプライマリ/スタンバイ設定を使用すると、認証情報のローテーションが次のリソースに影響する可能性があります。
  - 手動で作成された RDS Custom for Oracle のスタンバイインスタンス

    この問題を解決するには、手動のスタンバイデータベースを削除し、API コールを使用して Oracle リードレプリカを作成します。手動のスタンバイデータベースのシークレットを手動で管理し、ソースの DB インスタンスと一致させてください。
  - 手動で作成されたクロスリージョンリードレプリカ

    この問題を解決するには、シークレットを手動で管理して、プライマリ DB インスタンスと一致させてください。

パラメータファイルと設定ファイルに関する既知の問題

- コンピューティングのスケール、OS アップグレード、および RDS Custom がルートボリュームを置き換えるその他の処理の後に、`crontab` ファイルを設定する必要があります。`crontab` のバックアップを保持しておくことを強くお勧めします。
- `listener.ora` ファイルの設定は、次のガイドラインに従って行ってください。
  - ファイル内のすべてのエントリが同じ行にあることを確認します。これにより、インスタンス作成時にインデントの問題を回避できます。
  - `GLOBAL_DBNAME` の値が `SID_NAME` と等しいことを確認します。
  - `LISTENER` の値が命名規則 `L_`*`dbname`*`_001` に従っていることを確認します。
  - `listener.ora` ファイルにある接続先のデータベース名が変更されていないことを確認します。RDS Custom は、この接続を使用してデータベースの起動を検証します。このファイルを誤って変更すると、スケーリングコンピューティングやパッチ適用などの操作が失敗する可能性があります。

    次に、正しく設定された `listener.ora` の例を示します。

```
ADR_BASE_L_ORCL_001=/rdsdbdata/log/
```

```
USE_SID_AS_SERVICE_L_ORCL_001=ON
SID_LIST_L_ORCL_001=(SID_LIST = (SID_DESC = (SID_NAME = ORCL)(GLOBAL_DBNAME = ORCL)
 (ORACLE_HOME = /rdsdbbin/oracle.19.custom.r1.EE.1)))
SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_ORCL_001=OFF
L_ORCL_001=(DESCRIPTION_LIST = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT =
 XXXX)(HOST = x.x.x.x))) (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT = XXXX)
(HOST = 127.0.0.1))))
```

- コメントは、サーバーパラメータファイルまたは初期化パラメータファイルではサポートされていません。
- サーバーパラメータファイル (/rdsdbdata/config/oracle_pfile) で次の初期化パラメータを宣言する必要があります。
  - MEMORY_MAX_TARGET
  - MEMORY_TARGET
  - PGA_AGGREGATE_TARGET
  - PROCESSES
  - SGA_TARGET
  - USE_LARGE_PAGES

  前述のパラメータが /rdsdbdata/config/oracle_pfile で宣言されていない場合、リードレプリカの作成とコンピューティングのスケールが失敗する可能性があります。
- サーバーパラメータファイル、監査ファイル、listener.ora、tnsnames.ora、sqlnet.ora などの設定ファイルのシンボリックリンクは削除できません。また、これらのファイルのディレクトリ構造を変更することはできません。RDS Custom オートメーションは、これらのファイルが特定のディレクトリ構造に存在することを想定しています。

  初期化パラメータファイルからサーバーパラメータファイルを作成するには、次の構文を使用します。

```
CREATE SPFILE='/rdsdbdata/admin/$ORACLE_SID/pfile/spfile$ORACLE_SID.ora'
    FROM PFILE='/rdsdbdata/config/oracle_pfile';
```

RDS Custom for SQL Server の使用

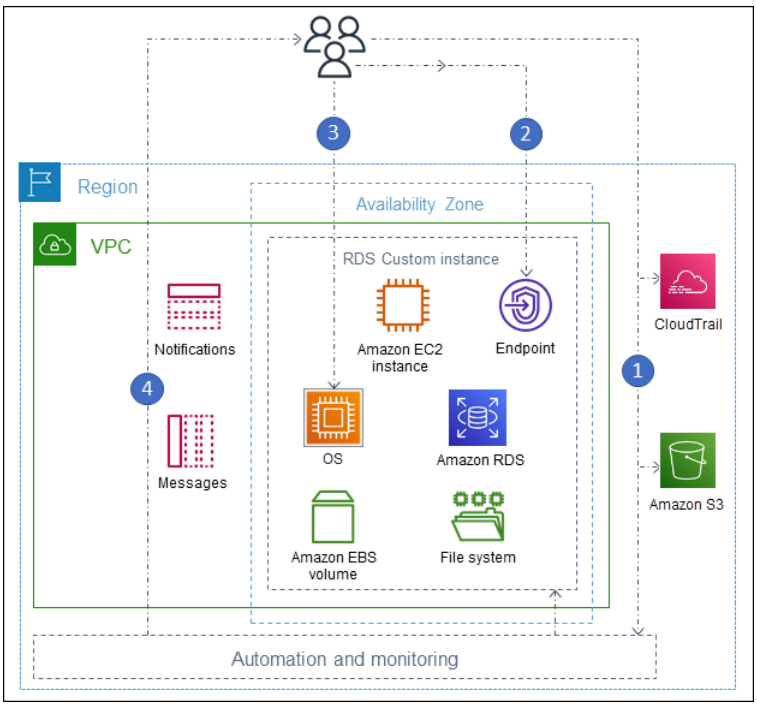
RDS Custom for SQL Server DB インスタンスの作成、管理、保守の手順を以下に示します。

トピック

- RDS for SQL Server のワークフロー
- Amazon RDS Custom for SQL Server の要件と制限
- Amazon RDS Custom for SQL Server の環境設定
- RDS Custom for SQL Server での Bring Your Own Media
- RDS Custom for SQL Server のカスタムエンジンバージョンの使用
- Amazon RDS Custom SQL Server の DB インスタンスの作成と接続
- Amazon RDS Custom for SQL Server DB インスタンスの管理
- RDS Custom for SQL Server による Microsoft Active Directory の操作
- RDS Custom for SQL Server のマルチ AZ 配置の管理
- Amazon RDS Custom for SQL Server DB インスタンスのバックアップと復元
- Amazon RDS Custom for SQL Server DB スナップショットのコピー
- オンプレミスデータベースを Amazon RDS Custom for SQL Server に移行する
- RDS Custom for SQL Server オペレーティングシステムの更新
- Amazon RDS Custom for SQL Server の DB インスタンスのアップグレード
- Amazon RDS Custom for SQL Server の DB に関する問題のトラブルシューティング

RDS for SQL Server のワークフロー

次の図表は、RDS Custom for SQL Server の一般的なワークフローを示しています。



ステップは次のとおりです。

1. RDS Custom が提供するエンジンバージョンから、RDS Custom for SQL Serverの DB インスタンスを作成します。

   詳細については、「RDS Custom for SQL Server DB インスタンスの作成」を参照してください。
2. アプリケーションを RDS Custom DB インスタンスエンドポイントに接続します。

   詳細については、「AWS Systems Managerを使用して RDS カスタム DB インスタンスに接続する」および「RDP を使用した RDS Custom DB インスタンスへの接続」を参照してください。
3. (オプション) ホストにアクセスしてソフトウェアをカスタマイズします。
4. RDS Custom オートメーションによって生成された通知とメッセージをモニタリングします。

RDS Custom の DB インスタンスを作成する

RDS Custom DB インスタンスは、`create-db-instance`コマンドを使用して作成します。この手順は、Amazon RDS インスタンスの作成と似ています。ただし、一部のパラメータは異なります。詳細については、「Amazon RDS Custom SQL Server の DB インスタンスの作成と接続」を参照してください。

データベース接続

Amazon RDS DB インスタンスと同様に、RDS Custom for SQL Serverの DB インスタンスは VPC 内に存在します。アプリケーションは、RDS for SQL Server と同様に、SQL Server 管理スイート (SSMS) などのクライアントを使用して RDS Custom インスタンスに接続します。

RDS Custom のカスタマイズ

RDS Custom ホストにアクセスして、ソフトウェアをインストールまたはカスタマイズできます。変更と RDS カスタムオートメーションの間の競合を回避するために、指定した期間オートメーションを一時停止できます。この期間中、RDS Custom はモニタリングまたはインスタンスのリカバリを実行しません。RDS Custom は期間終了時にフルオートメーションを再開します。詳細については、「RDS Custom オートメーションの一時停止と再開」を参照してください。

Amazon RDS Custom for SQL Server の要件と制限

Amazon RDS Custom for SQL Server 要件と制限事項の概要は、以下から簡単に参照できます。要件と制限事項は、関連するセクションにも記載されています。

トピック

- 利用可能なリージョンとバージョン
- RDS Custom for SQL Server の一般的な要件
- RDS Custom for SQL Server の DB インスタンスクラスでのサポート
- RDS Custom for SQL Server の制限
- RDS Custom for SQL Server DB の文字セットと照合順序の設定
- RDS Custom for SQL Server DB インスタンスのローカルタイムゾーン
- RDS Custom for SQL Server でのサービスマスターキーの使用
- RDS Custom for SQL Server による変更データキャプチャ (CDC) サポート

利用可能なリージョンとバージョン

機能の利用可能性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョン によって異なります。Amazon RDS Custom for SQL Server での Amazon RDS の利用可能なバージョンとリージョンの詳細については、「RDS Custom for SQL Server でサポートされているリージョンと DB エンジン」を参照してください。

RDS Custom for SQL Server の一般的な要件

Amazon RDS Custom for SQL Server ついては、次の要件に従ってください。

- RDS Custom for SQL Server の DB インスタンスクラスでのサポートに表示されるインスタンスクラスを使用します。サポートされているストレージタイプは、gp2、gp3、io1、および io2 Block Express のソリッドステートドライブ (SSD) タイプのみです。io1、gp2、gp3 の最大ストレージ制限は 16 TiB で、io2 は 64 TiB をサポートしています。
- RDS Custom DB インスタンスを作成するために、対称暗号化 AWS KMS キーがあることを確認してください。詳細については、「対称暗号化 AWS KMS キーであることを確認します。」を参照してください。
- AWS Identity and Access Management(IAM) ロールとインスタンスプロファイルを作成していることを確認します。詳細については、「IAM ロールおよびインスタンスプロファイルをマニュア

ルで作成する」および「AWS マネジメントコンソール を使用したインスタンスプロファイルの自動作成」を参照してください。

- RDS Custom が他の AWS のサービス サービスにアクセスするために使用できるネットワーク構成を必ず指定してください。特定の要件については、「ステップ 2: ネットワーキング、インスタンスプロファイル、および暗号化を構成する」を参照してください。
- RDS Custom と Amazon RDS DB インスタンスの合計数は、クォータ制限を超えてはいけません。例えば、クォータが 40 DB インスタンスの場合、20 のRDS Custom for SQL Server DB インスタンスと20 の Amazon RDS DB インスタンスを持つことができます。
- RDS Custom は、名前が `do-not-delete-rds-custom-` で始まる AWS CloudTrail 追跡を自動的に作成します。RDS Custom サポートの周辺構成は CloudTrail からのイベントに依存して、アクションが RDS Custom オートメーションに影響を与えるかどうかを判断します。RDS Custom は、最初の DB インスタンスを作成するときに追跡を作成します。既存の CloudTrail を使用するには、AWS サポートにお問い合わせください。詳細については、「AWS CloudTrail」を参照してください。

RDS Custom for SQL Server の DB インスタンスクラスでのサポート

describe-orderable-db-instance-options コマンドを使用して、使用中のリージョンで DB インスタンスクラスがサポートされているかどうかを確認します。

RDS Custom for SQL Server では、次の表に示す DB インスタンスクラスをサポートしています。

SQL Server エディション	RDS Custom サポート
Enterprise Edition	db.r5.xlarge-db.r5.24xlarge db.r5b.xlarge–db.r5b.24xlarge db.m5.xlarge-db.m5.24xlarge db.r6i.xlarge–db.r6i.32xlarge db.m6i.xlarge–db.m6i.32xlarge db.x2iedn.xlarge–db.x2iedn.32xlarge

SQL Server エディション	RDS Custom サポート
Standard Edition	db.r5.large-db.r5.24xlarge db.r5b.large–db.r5b.8xlarge db.m5.large-db.m5.24xlarge db.r6i.large–db.r6i.8xlarge db.m6i.large–db.m6i.8xlarge db.x2iedn.xlarge–db.x2iedn.8xlarge
Developer Edition	db.r5.xlarge-db.r5.24xlarge db.r5b.xlarge–db.r5b.24xlarge db.m5.xlarge-db.m5.24xlarge db.r6i.xlarge–db.r6i.32xlarge db.m6i.xlarge–db.m6i.32xlarge db.x2iedn.xlarge–db.x2iedn.32xlarge
Web Edition	db.r5.large-db.r5.4xlarge db.m5.large-db.m5.4xlarge db.r6i.large–db.r6i.4xlarge db.m6i.large–db.m6i.4xlarge db.r5b.large–db.r5b.4xlarge

db.x2iedn クラスタイプには、次の推奨事項が適用されます。

- 作成時は、ローカルストレージは未処理の未割り当てデバイスです。このインスタンスクラスで DB インスタンスを使用する前に、ローカルストレージをマウントしてフォーマットする必要があります。その後、tempdb を設定して最適なパフォーマンスを確保します。詳細については、「Optimize tempdb performance in Amazon RDS Custom for SQL Server using local instance storage」を参照してください。
- スケールコンピューティング、インスタンスの交換、スナップショットの復元、ポイントインタイムリカバリ (PITR) などの DB インスタンスオペレーションを実行すると、ローカルストレージは未処理の未割り当ての状態に戻ります。このような状況では、ドライブと tempdb を再マウント、再フォーマット、再構成して、機能を復元する必要があります。
- マルチ AZ インスタンスの場合は、スタンバイ DB インスタンスで構成を実行することをお勧めします。これにより、フェイルオーバーが発生しても、スタンバイインスタンスにすでに構成が適用されているため、システムは問題なく動作し続けます。

RDS Custom for SQL Server の制限

RDS Custom for SQL Server では、次の制限が適用されます。

- RDS Custom SQL Server DB インスタンスの Amazon RDS リードレプリカを作成することはできません。ただし、マルチ AZ 配置を使用して、高可用性を自動的に設定できます。詳細については、「RDS Custom for SQL Server のマルチ AZ 配置の管理」を参照してください。
- 既存の RDS Custom for SQL Server DB インスタンスの DB インスタンス識別子を変更することはできません。
- RDS Custom for SQL Server DB インスタンスが、カスタムエンジンバージョン (CEV) で作成されたものでない場合、Microsoft Windows オペレーティングシステムへの変更が保持される保証はありません。例えば、スナップショットを開始したり、ポイントインタイム復元オペレーションを開始したりすると、これらの変更は失われます。RDS Custom for SQL Server DB インスタンスが CEV で作成された場合、その変更は保持されます。
- 一部のオプションがサポートされていません。例えば、 RDS Custom for SQL Server DBインスタンスを作成する場合、次の操作を実行することはできません。
  - DB インスタンスクラスの CPU コア数とコアごとのスレッド数を変更します。
  - ストレージのオートスケーリングを有効にします。
  - 独自の DB パラメータグループ、オプショングループ、または文字セットを指定します。
  - Database Insights の Performance Insights を有効にします。
  - 自動マイナーバージョンアップグレードを実行します。
- DB インスタンスの最大ストレージは 64 TiB です。

- RDS Custom for SQL Server で RDS Proxy を使用することはできません。
- RDS Custom for SQL Server DB インスタンスに `describe-reserved-db-instances` API を使用することはできません。

RDS Custom for SQL Server DB の文字セットと照合順序の設定

概要

RDS Custom for SQL Server DB インスタンスでは、データの保存とソート方法を決定する文字セットと照合順序を設定できます。文字セットは許可される文字を定義し、照合順序ではデータのソートと比較のルールを指定します。多言語データを使用するアプリケーションや特定のソート要件があるアプリケーションに適した文字セットと照合順序を設定することは重要です。例えば、アクセント付き文字を処理し、言語固有のソートルールを定義したり、さまざまなロケールでデータの整合性を維持したりする必要がある場合があります。以下のセクションでは、RDS Custom for SQL Server DB インスタンスの文字セットと照合順序サポートについて説明します。

RDS Custom for SQL Server は、SQL_Latin、日本語、ドイツ語、アラビア語のロケールで、従来のエンコーディングと UTF-8 エンコーディングの両方で、さまざまなサーバー照合順序をサポートしています。デフォルトのサーバー照合順序は `SQL_Latin1_General_CP1_CI_AS` ですが、サポートされている別の照合順序を選択して使用することもできます。RDS for SQL Server が使用するのと同じ手順を使用して照合順序を選択できます。詳細については、「Amazon RDS for Microsoft SQL Server の照合と文字セットの管理」を参照してください。

考慮事項

RDS Custom for SQL Server でサーバー照合順序を使用する場合は、次の要件と制限が適用されます。

- RDS Custom for SQL Server DB インスタンスを作成するときに、サーバー照合順序を設定できます。DB インスタンスの作成後にサーバーレベルの照合順序を変更することはできません。
- DB スナップショットからの復元時やポイントインタイムリカバリ (PITR) 時は、サーバーレベルの照合順序を変更できません。
- RDS Custom for SQL Server CEV から DB インスタンスを作成しても、DB インスタンスは CEV のサーバー照合順序を継承しません。代わりに、デフォルトのサーバー照合順序である `SQL_Latin1_General_CP1_CI_AS` が使用されます。RDS Custom for SQL Server CEV でデフォルト以外のサーバー照合順序を設定していて、新しい DB インスタンスでも同じサーバー照合順序を使用したい場合は、CEV から DB インスタンスを作成するときに必ず同じ照合順序を選択してください。

> **Note**
>
> DB インスタンスの作成時に選択した照合順序が CEV の照合順序と異なる場合、新しい RDS Custom for SQL Server DB インスタンス上の Microsoft SQL Server システムデータベースは、更新された照合順序を使用するように再構築されます。再構築プロセスは、新しい RDS Custom for SQL Server DB インスタンスでのみ実行され、CEV 自体には影響しません。CEV 上のシステムデータベースに加えた以前の変更は、システムデータベースが再構築されると、新しい RDS Custom for SQL Server DB インスタンスでは保持されません。変更の例としては、`master` データベース内のユーザー定義オブジェクト、`msdb` データベースのスケジュールされたジョブ、CEV 上の `model` データベース内のデフォルトデータベース設定の変更などがあります。新しい RDS Custom for SQL Server DB インスタンスを作成した後は、変更を手動で再作成できます。

- RDS Custom for SQL Server カスタムエンジンバージョン (CEV) から DB インスタンスを作成し、CEV とは異なる照合順序を選択する場合は、新しい DB インスタンス上の Microsoft SQL Server システムデータベースを再構築できるように、CEV の作成に使用するゴールデンイメージ (AMI) が以下の要件を満たしていることを確認してください。
  - SQL Server 2022 の場合、`setup.exe` ファイルが次のパスにあることを確認します: `C:\Program Files\Microsoft SQL Server\160\Setup Bootstrap\SQL2022\setup.exe`
  - SQL Server 2019 の場合、`setup.exe` ファイルが次のパスにあることを確認します: `C:\Program Files\Microsoft SQL Server\150\Setup Bootstrap\SQL2019\setup.exe`
  - `master`、`model`、および `msdb` データベースのデータとログテンプレートのコピーは、デフォルトの場所に存在している必要があります。詳細については、Microsoft の公開ドキュメントの「システムデータベースを再構築する」を参照してください。
  - SQL Server データベースエンジンがサービスアカウントとして `NT Service\MSSQLSERVER` または `NT AUTHORITY\NETWORK SERVICE` を使用していることを確認します。DB インスタンスにデフォルト以外のサーバー照合順序を設定すると、他のアカウントには `C:\` ドライブに対する必要なアクセス許可が付与されません。
- 新しい DB インスタンス用に選択されたサーバー照合順序が CEV で設定されているものと同じ場合、新しい RDS Custom for SQL Server DB インスタンス上の Microsoft SQL Server システムデータベースは再構築プロセスを実行しません。CEV 上のシステムデータベースに加えた以前のすべての変更は、新しい RDS Custom for SQL Server DB インスタンスに自動的に保持されます。

サポートされている照合順序

照合順序は以下の表に示すいずれかの値に設定できます。

照合	説明
Arabic_100_BIN	Arabic-100、バイナリソート
Arabic_100_BIN2	Arabic-100、バイナリコードポイント比較ソート
Arabic_100_CI_AI	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_100_CI_AI_KS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_100_CI_AI_KS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Arabic_100_CI_AI_KS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Arabic_100_CI_AI_KS_WS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_100_CI_AI_KS_WS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Arabic_100_CI_AI_KS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片

照合	説明
	仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Arabic_100_CI_AI_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Arabic_100_CI_AI_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Arabic_100_CI_AI_WS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_100_CI_AI_WS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Arabic_100_CI_AI_WS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Arabic_100_CI_AS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_100_CI_AS_KS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
Arabic_100_CI_AS_KS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Arabic_100_CI_AS_KS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Arabic_100_CI_AS_KS_WS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_100_CI_AS_KS_WS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Arabic_100_CI_AS_KS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Arabic_100_CI_AS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Arabic_100_CI_AS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8

照合	説明
Arabic_100_CI_AS_WS	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_100_CI_AS_WS_SC	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Arabic_100_CI_AS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Arabic_100_CS_AI	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_100_CS_AI_KS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_100_CS_AI_KS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Arabic_100_CS_AI_KS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Arabic_100_CS_AI_KS_WS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Arabic_100_CS_AI_KS_WS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Arabic_100_CS_AI_KS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Arabic_100_CS_AI_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Arabic_100_CS_AI_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Arabic_100_CS_AI_WS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_100_CS_AI_WS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Arabic_100_CS_AI_WS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8

照合	説明
Arabic_100_CS_AS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_100_CS_AS_KS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_100_CS_AS_KS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Arabic_100_CS_AS_KS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Arabic_100_CS_AS_KS_WS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_100_CS_AS_KS_WS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Arabic_100_CS_AS_KS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Arabic_100_CS_AS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字

照合	説明
Arabic_100_CS_AS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Arabic_100_CS_AS_WS	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_100_CS_AS_WS_SC	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Arabic_100_CS_AS_WS_SC_UTF8	Arabic-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Arabic_BIN	アラビア語、バイナリソート
Arabic_BIN2	アラビア語、バイナリコードポイント比較ソート
Arabic_CI_AI	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_CI_AI_KS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_CI_AI_KS_WS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Arabic_CI_AI_WS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_CI_AS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_CI_AS_KS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_CI_AS_KS_WS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_CI_AS_WS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Arabic_CS_AI	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_CS_AI_KS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_CS_AI_KS_WS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_CS_AI_WS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり

照合	説明
Arabic_CS_AS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Arabic_CS_AS_KS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Arabic_CS_AS_KS_WS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Arabic_CS_AS_WS	アラビア語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Chinese_PRC_BIN2	Chinese-PRC、バイナリコードポイントの比較ソート
Chinese_PRC_CI_AS	中国語 - 中華人民共和国、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Chinese_Taiwan_Stroke_CI_AS	繁体字中国語 (台湾)、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Danish_Norwegian_CI_AS	デンマーク-ノルウェー語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Finnish_Swedish_CI_AS	フィンランド-スウェーデン語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
French_CI_AS	フランス語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_100_BIN	German-PhoneBook-100、バイナリソート
German_PhoneBook_100_BIN2	German-PhoneBook-100、バイナリコードポイント比較ソート
German_PhoneBook_100_CI_AI	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook _100_CI_AI_KS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_100_CI_AI_KS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
German_PhoneBook_100_CI_AI_KS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
German_PhoneBook_100_CI_AI_KS_WS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
German_PhoneBook_100_CI_AI_KS_WS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
German_PhoneBook_100_CI_AI_KS_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
German_PhoneBook_100_CI_AI_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
German_PhoneBook_100_CI_AI_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
German_PhoneBook_100_CI_AI_WS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
German_PhoneBook_100_CI_AI_WS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
German_PhoneBook_100_CI_AI_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8

照合	説明
German_PhoneBook_100_CI_AS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_100_CI_AS_KS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_100_CI_AS_KS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
German_PhoneBook_100_CI_AS_KS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
German_PhoneBook_100_CI_AS_KS_WS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_100_CI_AS_KS_WS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
German_PhoneBook_100_CI_AS_KS_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8

照合	説明
German_PhoneBook_100_CI_AS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
German_PhoneBook_100_CI_AS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
German_PhoneBook_100_CI_AS_WS	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
German_PhoneBook_100_CI_AS_WS_SC	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
German_PhoneBook_100_CI_AS_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
German_PhoneBook_100_CS_AI	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_100_CS_AI_KS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
German_PhoneBook_100_CS_AI_KS_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
German_PhoneBook_100_CS_AI_KS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
German_PhoneBook_100_CS_AI_KS_WS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
German_PhoneBook_100_CS_AI_KS_WS_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
German_PhoneBook_100_CS_AI_KS_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
German_PhoneBook_100_CS_AI_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
German_PhoneBook_100_CS_AI_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8

照合	説明
German_PhoneBook_100_CS_AI_WS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
German_PhoneBook_100_CS_AI_WS_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
German_PhoneBook_100_CS_AI_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
German_PhoneBook_100_CS_AS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_100_CS_AS_KS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_100_CS_AS_KS_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
German_PhoneBook_100_CS_AS_KS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8

照合	説明
German_PhoneBook_100_CS_AS_KS_WS	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
German_PhoneBook_100_CS_AS_KS_WS_SC	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
German_PhoneBook_100_CS_AS_KS_WS_SC_UTF8	German-PhoneBook-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
German_PhoneBook_BIN	German-PhoneBook、バイナリソート
German_PhoneBook_BIN2	German-PhoneBook、バイナリコードポイント比較ソート
German_PhoneBook_CI_AI	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_CI_AI_KS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_CI_AI_KS_WS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
German_PhoneBook_CI_AI_WS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり

照合	説明
German_PhoneBook_CI_AS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_CI_AS_KS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_CI_AS_KS_WS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_CI_AS_WS	German-PhoneBook、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
German_PhoneBook_CS_AI	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
German_PhoneBook_CS_AI_KS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_CS_AI_KS_WS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
German_PhoneBook_CS_AI_WS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
German_PhoneBook_CS_AS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
German_PhoneBook_CS_AS_KS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
German_PhoneBook_CS_AS_KS_WS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
German_PhoneBook_CS_AS_WS	German-PhoneBook、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Hebrew_BIN	ヘブライ語、バイナリソート
Hebrew_CI_AS	ヘブライ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_90_BIN	Japanese-90、バイナリソート
Japanese_90_BIN2	Japanese-90、バイナリコードポイント比較ソート
Japanese_90_CI_AI	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_90_CI_AI_KS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_90_CI_AI_KS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字

照合	説明
Japanese_90_CI_AI_KS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_90_CI_AI_KS_WS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_90_CI_AI_KS_WS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_90_CI_AI_KS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_90_CI_AI_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_90_CI_AI_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_90_CI_AI_WS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり

照合	説明
Japanese_90_CI_AI_WS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_90_CI_AI_WS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_90_CI_AS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_90_CI_AS_KS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_90_CI_AS_KS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_90_CI_AS_KS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_90_CI_AS_KS_WS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_90_CI_AS_KS_WS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字

照合	説明
Japanese_90_CI_AS_KS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_90_CI_AS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_90_CI_AS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_90_CI_AS_WS	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_90_CI_AS_WS_SC	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_90_CI_AS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_90_CS_AI	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_90_CS_AI_KS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
Japanese_90_CS_AI_KS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_90_CS_AI_KS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_90_CS_AI_KS_WS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_90_CS_AI_KS_WS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_90_CS_AI_KS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_90_CS_AI_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_90_CS_AI_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8

照合	説明
Japanese_90_CS_AI_WS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_90_CS_AI_WS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_90_CS_AI_WS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_90_CS_AS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_90_CS_AS_KS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_90_CS_AS_KS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_90_CS_AS_KS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_90_CS_AS_KS_WS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Japanese_90_CS_AS_KS_WS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_90_CS_AS_KS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_90_CS_AS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_90_CS_AS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_90_CS_AS_WS	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_90_CS_AS_WS_SC	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_90_CS_AS_WS_SC_UTF8	Japanese-90、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_BIN	日本語、バイナリソート
Japanese_BIN2	日本語、バイナリコードポイント比較ソート

照合	説明
Japanese_Bushu_Kakusu_100_BIN	Japanese-Bushu-Kakusu-100、バイナリソート
Japanese_Bushu_Kakusu_100_BIN2	Japanese-Bushu-Kakusu-100、バイナリコードポイント比較ソート
Japanese_Bushu_Kakusu_100_CI_AI	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CI_AI_KS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CI_AI_KS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CI_AI_KS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字

照合	説明
Japanese_Bushu_Kakusu_100_CI_AI_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AI_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CI_AI_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AI_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Bushu_Kakusu_100_CI_AI_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CI_AI_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
Japanese_Bushu_Kakusu_100_CI_AS_KS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CI_AS_KS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CI_AS_KS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CI_AS_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字

照合	説明
Japanese_Bushu_Kakusu_100_CI_AS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CI_AS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Bushu_Kakusu_100_CI_AS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CI_AS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AI	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CS_AI_KS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CS_AI_KS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字

照合	説明
Japanese_Bushu_Kakusu_100_CS_AI_KS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CS_AI_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AI_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CS_AI_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AI_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり

照合	説明
Japanese_Bushu_Kakusu_100_CS_AI_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CS_AI_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CS_AS_KS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Bushu_Kakusu_100_CS_AS_KS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CS_AS_KS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CS_AS_KS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_Bushu_Kakusu_100_CS_AS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_Bushu_Kakusu_100_CS_AS_WS	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Bushu_Kakusu_100_CS_AS_WS_SC	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_Bushu_Kakusu_100_CS_AS_WS_SC_UTF8	Japanese-Bushu-Kakusu-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_Bushu_Kakusu_140_BIN	Japanese-Bushu-Kakusu-140、バイナリソート

照合	説明
Japanese_Bushu_Kakusu_140_BIN2	Japanese-Bushu-Kakusu-140、バイナリコードポイント比較ソート
Japanese_Bushu_Kakusu_140_CI_AI	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AI_KS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AI_KS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_KS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AI_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8

照合	説明
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AI_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あリ、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり

照合	説明
Japanese_Bushu_Kakusu_140_CI_AI_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AI_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AI_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AI_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_Bushu_Kakusu_140_CI_AS_KS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AS_KS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_KS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AS_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8

照合	説明
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AS_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_Bushu_Kakusu_140_CI_AS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CI_AS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CI_AS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AI	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AI_KS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AI_KS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8

照合	説明
Japanese_Bushu_Kakusu_140_CS_AI_KS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AI_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AI_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8

照合	説明
Japanese_Bushu_Kakusu_140_CS_AI_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AI_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AI_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AI_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AI_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AI_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり

照合	説明
Japanese_Bushu_Kakusu_140_CS_AI_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AS_KS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AS_KS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_KS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AS_KS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8

照合	説明
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AS_KS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり

照合	説明
Japanese_Bushu_Kakusu_140_CS_AS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_WS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_Bushu_Kakusu_140_CS_AS_WS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_Bushu_Kakusu_140_CS_AS_WS_VSS	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_Bushu_Kakusu_140_CS_AS_WS_VSS_UTF8	Japanese-Bushu-Kakusu-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_CI_AI	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
Japanese_CI_AI_KS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_CI_AI_KS_WS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_CI_AI_WS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_CI_AS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_CI_AS_KS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_CI_AS_KS_WS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_CI_AS_WS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_CS_AI	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_CS_AI_KS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
Japanese_CS_AI_KS_WS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_CS_AI_WS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_CS_AS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_CS_AS_KS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_CS_AS_KS_WS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_CS_AS_WS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Unicode_BIN	Japanese-Unicode、バイナリソート
Japanese_Unicode_BIN2	Japanese-Unicode、バイナリコードポイント比較ソート
Japanese_Unicode_CI_AI	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Unicode_CI_AI_KS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
Japanese_Unicode_CI_AI_KS_WS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_Unicode_CI_AI_WS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Unicode_CI_AS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Unicode_CI_AS_KS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Unicode_CI_AS_KS_WS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_Unicode_CI_AS_WS	Japanese-Unicode、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Unicode_CS_AI	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Unicode_CS_AI_KS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Unicode_CS_AI_KS_WS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Japanese_Unicode_CS_AI_WS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_Unicode_CS_AS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_Unicode_CS_AS_KS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_Unicode_CS_AS_KS_WS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_Unicode_CS_AS_WS	Japanese-Unicode、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_XJIS_100_BIN	Japanese-XJIS-100、バイナリソート
Japanese_XJIS_100_BIN2	Japanese-XJIS-100、バイナリコードポイント比較ソート
Japanese_XJIS_100_CI_AI	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_XJIS_100_CI_AI_KS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_XJIS_100_CI_AI_KS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字

照合	説明
Japanese_XJIS_100_CI_AI_KS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CI_AI_KS_WS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_XJIS_100_CI_AI_KS_WS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_XJIS_100_CI_AI_KS_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CI_AI_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_XJIS_100_CI_AI_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CI_AI_WS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり

照合	説明
Japanese_XJIS_100_CI_AI_WS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_XJIS_100_CI_AI_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CI_AS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_XJIS_100_CI_AS_KS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_XJIS_100_CI_AS_KS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_XJIS_100_CI_AS_KS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CI_AS_KS_WS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_XJIS_100_CI_AS_KS_WS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字

照合	説明
Japanese_XJIS_100_CI_AS_KS_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CI_AS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_XJIS_100_CI_AS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CI_AS_WS	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_XJIS_100_CI_AS_WS_SC	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_XJIS_100_CI_AS_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CS_AI	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_XJIS_100_CS_AI_KS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし

照合	説明
Japanese_XJIS_100_CS_AI_KS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_XJIS_100_CS_AI_KS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CS_AI_KS_WS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり
Japanese_XJIS_100_CS_AI_KS_WS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_XJIS_100_CS_AI_KS_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CS_AI_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_XJIS_100_CS_AI_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8

照合	説明
Japanese_XJIS_100_CS_AI_WS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_XJIS_100_CS_AI_WS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字
Japanese_XJIS_100_CS_AI_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CS_AS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_XJIS_100_CS_AS_KS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Japanese_XJIS_100_CS_AS_KS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字
Japanese_XJIS_100_CS_AS_KS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CS_AS_KS_WS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり

照合	説明
Japanese_XJIS_100_CS_AS_KS_WS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字
Japanese_XJIS_100_CS_AS_KS_WS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、UTF8
Japanese_XJIS_100_CS_AS_SC	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字
Japanese_XJIS_100_CS_AS_SC_UTF8	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Japanese_XJIS_100_CS_AS_WS	Japanese-XJIS-100、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Japanese_XJIS_140_BIN	Japanese-XJIS-140、バイナリソート
Japanese_XJIS_140_BIN2	Japanese-XJIS-140、バイナリコードポイント比較ソート
Japanese_XJIS_140_CI_AI	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_XJIS_140_CI_AI_KS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AI_KS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AI_KS_VSS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AI_KS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AI_KS_WS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AI_KS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8

照合	説明
Japanese_XJIS_140_CI_AI_KS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AI_KS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AI_UTF8	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AI_VSS	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AI_VSS_UTF8	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AI_WS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_XJIS_140_CI_AI_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AI_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AI_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AS	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AS_KS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AS_KS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8

照合	説明
Japanese_XJIS_140_CI_AS_KS_VSS	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AS_KS_VSS_UTF8	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AS_KS_WS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AS_KS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AS_KS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AS_KS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8

照合	説明
Japanese_XJIS_140_CI_AS_UTF8	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AS_VSS	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AS_VSS_UTF8	Japane-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CI_AS_WS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CI_AS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり

照合	説明
Japanese_XJIS_140_CI_AS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AI	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AI_KS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AI_KS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AI_KS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AI_KS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8

照合	説明
Japanese_XJIS_140_CS_AI_KS_WS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AI_KS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AI_KS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AI_KS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AI_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AI_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり

照合	説明
Japanese_XJIS_140_CS_AI_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AI_WS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AI_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AI_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AI_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_XJIS_140_CS_AS_KS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AS_KS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AS_KS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AS_KS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AS_KS_WS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CS_AS_KS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8

照合	説明
Japanese_XJIS_140_CS_AS_KS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AS_KS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり、UTF8
Japanese_XJIS_140_CS_AS_WS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし

照合	説明
Japanese_XJIS_140_CS_AS_WS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし、UTF8
Japanese_XJIS_140_CS_AS_WS_VSS	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AS_WS_VSS_UTF8	Japanese-XJIS-140、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり、補足文字、バリエーションセレクタの区別あり、UTF8
Korean_Wansung_CI_AS	韓国語 (Wansung)、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_100_BIN	Latin1-General-100、バイナリソート
Latin1_General_100_BIN2	Latin1-General-100、バイナリコードポイント比較ソート
Latin1_General_100_BIN2_UTF8	Latin1-General-100、バイナリコードポイント比較ソート、UTF8
Latin1_General_100_CI_AS	Latin1-General-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
Latin1_General_100_CI_AS_SC_UTF8	Latin1-General-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、UTF8
Latin1_General_BIN	Latin1-General、バイナリソート
Latin1_General_BIN2	Latin1-General、バイナリ、コードポイント比較ソート
Latin1_General_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_CI_AS_KS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Latin1_General_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Modern_Spanish_CI_AS	現代スペイン語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
SQL_1xCompat_CP850_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 49 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP1_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 54 (非 Unicode データの場合)
SQL_Latin1_General_CP1_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 52 (非 Unicode データの場合)
SQL_Latin1_General_CP1_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 51 (非 Unicode データの場合)
SQL_Latin1_General_CP1250_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1250 の SQL Server ソート順 82 (非 Unicode データの場合)
SQL_Latin1_General_CP1250_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1250 の SQL Server ソート順 81 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP1251_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1251 の SQL Server ソート順 106 (非 Unicode データの場合)
SQL_Latin1_General_CP1251_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1251 の SQL Server ソート順 105 (非 Unicode データの場合)
SQL_Latin1_General_CP1253_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1253 の SQL Server ソート順 124 (非 Unicode データの場合)
SQL_Latin1_General_CP1253_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1253 の SQL Server ソート順 114 (非 Unicode データの場合)
SQL_Latin1_General_CP1253_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1253 の SQL Server ソート順 113 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP1254_CI_AS	トルコ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1254 の SQL Server ソート順 130 (非 Unicode データの場合)
SQL_Latin1_General_CP1254_CS_AS	トルコ語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1254 の SQL Server ソート順 129 (非 Unicode データの場合)
SQL_Latin1_General_CP1255_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1255 の SQL Server ソート順 138 (非 Unicode データの場合)
SQL_Latin1_General_CP1255_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1255 の SQL Server ソート順 137 (非 Unicode データの場合)
SQL_Latin1_General_CP1256_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1256 の SQL Server ソート順 146 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP1256_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1256 の SQL Server ソート順 145 (非 Unicode データの場合)
SQL_Latin1_General_CP1257_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1257 の SQL Server ソート順 154 (非 Unicode データの場合)
SQL_Latin1_General_CP1257_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1257 の SQL Server ソート順 153 (非 Unicode データの場合)
SQL_Latin1_General_CP437_BIN	Latin1-General、バイナリソート (Unicode データの場合)、コードページ 437 の SQL Server ソート順 30 (非 Unicode データの場合)
SQL_Latin1_General_CP437_BIN2	Latin1-General、バイナリコードポイント比較ソート (Unicode データの場合)、コードページ 437 の SQL Server ソート順 30 (非 Unicode データの場合)
SQL_Latin1_General_CP437_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 437 の SQL Server ソート順 34 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP437_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 437 の SQL Server ソート順 32 (非 Unicode データの場合)
SQL_Latin1_General_CP437_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 437 の SQL Server ソート順 31 (非 Unicode データの場合)
SQL_Latin1_General_CP850_BIN	Latin1-General、バイナリソート (Unicode データの場合)、コードページ 850 の SQL Server ソート順 40 (非 Unicode データの場合)
SQL_Latin1_General_CP850_BIN2	Latin1-General、バイナリコードポイント比較ソート (Unicode データの場合)、コードページ 850 の SQL Server ソート順 40 (非 Unicode データの場合)
SQL_Latin1_General_CP850_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 44 (非 Unicode データの場合)
SQL_Latin1_General_CP850_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 42 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP850_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 41 (非 Unicode データの場合)
SQL_Latin1_General_Pref_CP1_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 53 (非 Unicode データの場合)
SQL_Latin1_General_Pref_CP437_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 437 の SQL Server ソート順 33 (非 Unicode データの場合)
SQL_Latin1_General_Pref_CP850_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 43 (非 Unicode データの場合)
Thai_CI_AS	タイ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

RDS Custom for SQL Server DB インスタンスのローカルタイムゾーン

RDS Custom for SQL Server DB インスタンスのタイムゾーンは、デフォルトで設定されます。現在のデフォルトは協定世界時 (UTC) です。DB インスタンスのタイムゾーンをローカルタイムゾーンに設定して、アプリケーションのタイムゾーンと一致させることも可能です。

DB インスタンスを最初に作成するときにタイムゾーンを設定します。AWS マネジメントコンソール、Amazon RDS API の CreateDBInstance アクション、または AWS CLI の create-db-instance コマンドを使用して、DB インスタンスを作成できます。

DB インスタンスがマルチ AZ 配置の一部である場合、フェイルオーバーが行われても、タイムゾーンは設定したローカルタイムゾーンを維持します。

特定の時点への復元をリクエストする場合には、復元を行う時間を指定します。時間は、ローカルタイムゾーンで表示されます。詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

DB インスタンスにローカルタイムゾーンを設定する際の制限事項を以下に示します。

- インスタンスの作成時に DB インスタンスのタイムゾーンを設定できますが、既存の RDS Custom for SQL Server DB インスタンスのタイムゾーンを変更することはできません。
- 既存の RDS Custom for SQL Server DB インスタンスのタイムゾーンが変更された場合、RDS Custom は DB インスタンスのステータスを `unsupported-configuration` に変更し、イベント通知を送信します。
- あるタイムゾーンの DB インスタンスのスナップショットを、タイムゾーンの異なる DB インスタンスに復元することはできません。
- あるタイムゾーンのバックアップファイルを、別のタイムゾーンに復元しないことを強くお勧めします。バックアップファイルを別のタイムゾーンに復元した場合は、タイムゾーンの変更によるクエリとアプリケーションへの影響を精査する必要があります。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

サポートされているタイムゾーン

ローカルタイムゾーンは以下の表に示されているいずれかの値に設定できます。

タイムゾーン	標準の時間オフセット	説明	コメント
アフガニスタン標準時	(UTC+04:30)	カブール	このタイムゾーンは夏時間を考慮しません。
アラスカ標準時	(UTC–09:00)	アラスカ	
アリューシャン標準時	(UTC–10:00)	アリューシャン列島	
アルタイ標準時	(UTC+07:00)	バルナウル (ゴルノ＝アルタイスク)	
アラブ標準時	(UTC+03:00)	クウェート (リヤド)	このタイムゾーンは夏時間を考慮しません。
アラビア標準時	(UTC+04:00)	アブダビ、マスカット	
アラビア標準時	(UTC+03:00)	バグダッド	このタイムゾーンは夏時間を考慮しません。
アルゼンチン標準時	(UTC–03:00)	ブエノスアイレス市	このタイムゾーンは夏時間を考慮しません。
アストラハン標準時	(UTC+04:00)	アストラハン (ウリヤノフスク州)	
大西洋標準時	(UTC–04:00)	大西洋標準時 (カナダ)	
中部オーストラリア標準時	(UTC+09:30)	ダーウィン	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
オーストラリア中西部標準時	(UTC+08:45)	ユークラ	
東部オーストラリア標準時	(UTC+10:00)	キャンベラ、メルボルン、シドニー	
アゼルバイジャン標準時	(UTC+04:00)	バクー	
アゾレス諸島標準時	(UTC–01:00)	アゾレス諸島	
バイア州標準時	(UTC–03:00)	サルバドル	
バングラデシュ標準時	(UTC+06:00)	ダッカ	このタイムゾーンは夏時間を考慮しません。
ベラルーシ標準時	(UTC+03:00)	ミンスク	このタイムゾーンは夏時間を考慮しません。
ブーゲンビル標準時	(UTC+11:00)	ブーゲンビル島	
中部カナダ標準時	(UTC–06:00)	サスカチュワン	このタイムゾーンは夏時間を考慮しません。
カーボベルデ標準時	(UTC–01:00)	カーボベルデ島	このタイムゾーンは夏時間を考慮しません。
コーカサス標準時	(UTC+04:00)	エレバン	
中部 オーストラリア標準時	(UTC+09:30)	アデレード	
アメリカ中部標準時	(UTC–06:00)	中米	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
中央アジア標準時	(UTC+06:00)	アスターナ	このタイムゾーンは夏時間を考慮しません。
中部ブラジル標準時	(UTC–04:00)	クヤバ	
中央ヨーロッパ標準時	(UTC+01:00)	ベオグラード、ブラティスラヴァ、ブダペスト、リュブヤナ、プラハ	
中央ヨーロッパ標準時	(UTC+01:00)	サラエボ、スコピエ、ワルシャワ、ザグレブ	
中央太平洋標準時	(UTC+11:00)	ソロモン諸島、ニューカレドニア	このタイムゾーンは夏時間を考慮しません。
中部標準時	(UTC–06:00)	中部標準時 (米国およびカナダ)	
中部標準時 (メキシコ)	(UTC–06:00)	グアダラハラ、メキシコシティ、モンテレイ	
チャタム諸島標準時	(UTC+12:45)	チャタム諸島	
中国標準時	(UTC+08:00)	北京、重慶、香港特別行政区、ウルムチ	このタイムゾーンは夏時間を考慮しません。
キューバ標準時	(UTC–05:00)	ハバナ	

タイムゾーン	標準の時間オフセット	説明	コメント
日付変更線標準時	(UTC–12:00)	国際日付変更線西	このタイムゾーンは夏時間を考慮しません。
アフリカ東部標準時	(UTC+03:00)	ナイロビ	このタイムゾーンは夏時間を考慮しません。
オーストラリア東部標準時	(UTC+10:00)	ブリスベン	このタイムゾーンは夏時間を考慮しません。
欧州東部標準時	(UTC+02:00)	キシナウ	
南米東部標準時	(UTC–03:00)	ブラジリア	
イースター島標準時	(UTC–06:00)	イースター島	
東部標準時	(UTC–05:00)	東部標準時 (米国およびカナダ)	
東部標準時 (メキシコ)	(UTC–05:00)	チェトゥマル	
エジプト標準時	(UTC+02:00)	カイロ	
エカテリンブルク標準時	(UTC+05:00)	エカテリンブルク	
フィジー標準時	(UTC+12:00)	フィジー	
FLE 標準時	(UTC+02:00)	ヘルシンキ、キエフ、リガ、ソフィア、タリン、ビリニュス	
ジョージア標準時	(UTC+04:00)	トビリシ	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
GMT 標準時	(UTC)	ダブリン エジンバラ、リスボン、ロンドン	このタイムゾーンはグリニッジ標準時と同じではありません。このタイムゾーンは夏時間を考慮します。
グリーンランド標準時	(UTC–03:00)	グリーンランド	
グリニッジ標準時	(UTC)	モンロビア、レイキャビク	このタイムゾーンは夏時間を考慮しません。
GTB 標準時	(UTC+02:00)	アテネ、ブカレスト	
ハイチ標準時	(UTC–05:00)	ハイチ	
ハワイ標準時	(UTC–10:00)	ハワイ	
インド標準時	(UTC+05:30)	チェンナイ、カルカッタ、ムンバイ、ニューデリー	このタイムゾーンは夏時間を考慮しません。
イラン標準時	(UTC+03:30)	テヘラン	
イスラエル標準時	(UTC+02:00)	エルサレム	
ヨルダン標準時	(UTC+02:00)	アンマン	
カリーニングラード標準時	(UTC+02:00)	カリーニングラード	
カムチャツカ標準時	(UTC+12:00)	ペトロパブロフスク・カムチャツキー – オールド	

タイムゾーン	標準の時間オフセット	説明	コメント
韓国標準時	(UTC+09:00)	ソウル	このタイムゾーンは夏時間を考慮しません。
リビア標準時	(UTC+02:00)	トリポリ	
ライン諸島標準時	(UTC+14:00)	キリティマティ島	
ロード・ハウ標準時	(UTC+10:30)	ロード・ハウ島	
マガダン標準時	(UTC+11:00)	マガダン	このタイムゾーンは夏時間を考慮しません。
マガジャネス標準時	(UTC–03:00)	プンタ・アレーナス	
マルケサス標準時	(UTC–09:30)	マルケサス諸島	
モーリシャス標準時	(UTC+04:00)	ポートルイス	このタイムゾーンは夏時間を考慮しません。
中東標準時	(UTC+02:00)	ベイルート	
モンテビデオ標準時	(UTC–03:00)	モンテビデオ	
モロッコ標準時	(UTC+01:00)	カサブランカ	
山岳部標準時	(UTC–07:00)	山地標準時 (米国およびカナダ)	
山岳部標準時 (メキシコ)	(UTC–07:00)	チワワ、ラパス、マサトラン	
ミャンマー標準時	(UTC+06:30)	ヤンゴン (ラングーン)	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
中央アジア北部標準時	(UTC+07:00)	ノヴォシビルスク	
ナミビア標準時	(UTC+02:00)	ウィントフック	
ネパール標準時	(UTC+05:45)	カトマンズ	このタイムゾーンは夏時間を考慮しません。
ニュージーランド標準時	(UTC+12:00)	オークランド、ウェリントン	
ニューファンドランド標準時	(UTC–03:30)	ニューファンドランド	
ノーフォーク標準時	(UTC+11:00)	ノーフォーク島	
北アジア東部標準時	(UTC+08:00)	イルクーツク	
北アジア標準時	(UTC+07:00)	クラスノヤルスク	
北朝鮮標準時	(UTC+09:00)	平壌	
オムスク標準時	(UTC+06:00)	オムスク	
太平洋南アメリカ標準時	(UTC–03:00)	サンティアゴ	
太平洋標準時	(UTC–08:00)	太平洋標準時 (米国およびカナダ)	
太平洋標準時 (メキシコ)	(UTC–08:00)	バハ・カリフォルニア	
パキスタン標準時	(UTC+05:00)	イスラマバード (カラチ)	このタイムゾーンは夏時間を考慮しません。
パラグアイ標準時	(UTC–04:00)	アスンシオン	

タイムゾーン	標準の時間オフセット	説明	コメント
ロマンス標準時	(UTC+01:00)	ブリュッセル、コペンハーゲン、マドリード、パリ	
ロシアタイムゾーン 10	(UTC+11:00)	チョクルダフ	
ロシアタイムゾーン 11	(UTC+12:00)	ペトロパブロフスク・カムチャツキー・アナディル	
ロシアタイムゾーン 3	(UTC+04:00)	イジェフスク	
ロシア標準時	(UTC+03:00)	モスクワ、サンクト・ペテルブルク、ヴォルゴグラード	このタイムゾーンは夏時間を考慮しません。
SA 東部標準時	(UTC–03:00)	フォルタレザカイエンヌ	このタイムゾーンは夏時間を考慮しません。
南アメリカ太平洋標準時	(UTC–05:00)	ボゴタ、リマ、キト、リオブランコ	このタイムゾーンは夏時間を考慮しません。
SA 西部標準時	(UTC–04:00)	ジョージタウン、ラパス、マナウス、サンフアン	このタイムゾーンは夏時間を考慮しません。
サンピエール標準時	(UTC–03:00)	サンピエール・ミクロン	
サハリン標準時	(UTC+11:00)	サハリン	
サモア標準時	(UTC+13:00)	サモア	
サントメ標準時	(UTC+01:00)	サントメ	

タイムゾーン	標準の時間オフセット	説明	コメント
サラトフ標準時	(UTC+04:00)	サラトフ	
東南アジア標準時	(UTC+07:00)	バンコク、ハノイ、ジャカルタ	このタイムゾーンは夏時間を考慮しません。
シンガポール標準時	(UTC+08:00)	クアラルンプール、シンガポール	このタイムゾーンは夏時間を考慮しません。
南アフリカ標準時	(UTC+02:00)	ハラレ (プレトリア)	このタイムゾーンは夏時間を考慮しません。
スリランカ標準時	(UTC+05:30)	スリ・ジャヤワルデネプラ	このタイムゾーンは夏時間を考慮しません。
スーダン標準時	(UTC+02:00)	ハルツーム	
シリア標準時	(UTC+02:00)	ダマスカス	
台北標準時	(UTC+08:00)	台北	このタイムゾーンは夏時間を考慮しません。
タスマニア標準時	(UTC+10:00)	ホバート	
トカンティンス標準時	(UTC–03:00)	アラグアイナ	
日本標準時	(UTC+09:00)	大阪、札幌、東京	このタイムゾーンは夏時間を考慮しません。
トムスク標準時	(UTC+07:00)	トムスク	

タイムゾーン	標準の時間オフセット	説明	コメント
トンガ標準時	(UTC+13:00)	ヌクアロファ	このタイムゾーンは夏時間を考慮しません。
トランスバイカル標準時	(UTC+09:00)	チタ	
トルコ標準時	(UTC+03:00)	イスタンブール	
タークス・カイコス標準時	(UTC–05:00)	タークス・カイコス島	
ウランバートル標準時	(UTC+08:00)	ウランバートル	このタイムゾーンは夏時間を考慮しません。
米国東部標準時	(UTC–05:00)	インディアナ東部	
山地標準時 (米国)	(UTC–07:00)	アリゾナ	このタイムゾーンは夏時間を考慮しません。
UTC	UTC	協定世界時	このタイムゾーンは夏時間を考慮しません。
UTC–02	(UTC–02:00)	協定世界時–02	このタイムゾーンは夏時間を考慮しません。
UTC–08	(UTC–08:00)	協定世界時–08	
UTC–09	(UTC–09:00)	協定世界時–09	
UTC–11	(UTC–11:00)	協定世界時–11	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
UTC+12	(UTC+12:00)	協定世界時+12	このタイムゾーンは夏時間を考慮しません。
UTC+13	(UTC+13:00)	協定世界時+13	
ベネズエラ標準時	(UTC–04:00)	カラカス	このタイムゾーンは夏時間を考慮しません。
ウラジオストク標準時	(UTC+10:00)	ウラジオストク	
ヴォルゴグラード標準時	(UTC+04:00)	ヴォルゴグラード	
オーストラリア西部標準時	(UTC+08:00)	パース	このタイムゾーンは夏時間を考慮しません。
中部アフリカ西部標準時	(UTC+01:00)	西部・中部アフリカ	このタイムゾーンは夏時間を考慮しません。
ヨーロッパ西部標準時	(UTC+01:00)	アムステルダム、ベルリン、ベルン、ローマ、ストックホルム、ウィーン	
モンゴル西部標準時	(UTC+07:00)	ホヴド	
西アジア標準時	(UTC+05:00)	アシガバート (タシュケント)	このタイムゾーンは夏時間を考慮しません。
西岸標準時	(UTC+02:00)	ガザ (ヘブロン)	

タイムゾーン	標準の時間オフセット	説明	コメント
西太平洋標準時	(UTC+10:00)	グアム、ポートモレスビー	このタイムゾーンは夏時間を考慮しません。
ヤクーツク標準時	(UTC+09:00)	ヤクーツク	

RDS Custom for SQL Server でのサービスマスターキーの使用

RDS Custom for SQL Server は、サービスマスターキー (SMK) の使用をサポートしています。RDS Custom は、RDS Custom for SQL Server DB インスタンスの存続期間中、同じ SMK を保持します。同じ SMK を保持することで、DB インスタンスは、リンクされたサーバーのパスワードや認証情報など、その SMK で暗号化されたオブジェクトを使用できます。マルチ AZ 配置を使用する場合も、RDS Custom はプライマリ DB インスタンスとセカンダリ DB インスタンス間で SMK を同期して維持します。

トピック

- 利用可能なリージョンとバージョン
- サポートされている機能
- TDE の使用
- 機能の設定
- 要件と制限

利用可能なリージョンとバージョン

SMK の使用は、RDS Custom for SQL Server が利用可能なすべてのリージョンで、RDS Custom で利用可能なすべての SQL Server バージョンでサポートされています。RDS Custom for SQL Server での Amazon RDS の利用可能なバージョンとリージョンの詳細については、「RDS Custom for SQL Server でサポートされているリージョンと DB エンジン」を参照してください。

サポートされている機能

RDS Custom for SQL Server で SMK を使用する場合、次の機能がサポートされています。

- Transparent Data Encryption (TDE)

- 列レベルの暗号化
- データベースメール
- リンクサーバー
- SQL Server Integration Services (SSIS)

TDE の使用

SMK を使用すると、透過的なデータ暗号化 (TDE) を設定して、データをストレージに書き込む前に暗号化し、データをストレージから読み取るときに自動的に復号できます。RDS for SQL Server とは異なり、RDS Custom for SQL Server DB インスタンスで TDE を設定する場合、オプショングループを使用する必要はありません。代わりに、証明書とデータベース暗号化キーを作成したら、次のコマンドを実行してデータベースレベルで TDE を有効にすることができます。

```
ALTER DATABASE [myDatabase] SET ENCRYPTION ON;
```

RDS for SQL Server での TDE の使用の詳細については、「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。

Microsoft SQL Server での TDE の詳細については、Microsoft ドキュメントの「透過的なデータ暗号化」を参照してください。

機能の設定

RDS Custom for SQL Server で SMK を使用する機能の詳細な設定手順については、Amazon RDS データベースブログの以下の投稿を参照してください。

- リンクサーバー: Configuring linked servers on RDS Custom for SQL Server
- SSIS: Migrate SSIS packages to RDS Custom for SQL Server
- TDE: Secure your data using TDE on RDS Custom for SQL Server

要件と制限

RDS Custom for SQL Server DB インスタンスで SMK を使用する場合は、次の要件と制限に注意してください。

- DB インスタンスで SMK を再生成する場合は、すぐに手動 DB スナップショットを実行する必要があります。可能であれば、SMK を再生成しないことをお勧めします。

- サーバー証明書とデータベースマスターキーパスワードのバックアップを維持する必要があります。これらのバックアップを維持しないと、データが失われる可能性があります。
- SSIS を設定する場合は、SSM ドキュメントを使用して、スケールコンピューティングの実行時またはホスト交換時に RDS Custom for SQL Server DB インスタンスをドメインに結合する必要があります。
- TDE または列暗号化が有効になっている場合、データベースバックアップは自動的に暗号化されます。スナップショットの復元またはポイントインタイムリカバリを実行すると、ソース DB インスタンスの SMK が復元用のデータを復号するために復元され、復元されたインスタンスのデータを再暗号化するための新しい SMK が生成されます。

Microsoft SQL Server のサービスマスターキーの詳細については、Microsoft ドキュメントの「SQL Server とデータベースの暗号化キー」を参照してください。

RDS Custom for SQL Server による変更データキャプチャ (CDC) サポート

概要

RDS Custom for SQL Server は、変更データキャプチャ (CDC) をネイティブにサポートしているため、SQL Server テーブルのデータ変更を追跡およびキャプチャできます。CDC は、後続の取得と分析のために、これらの変更に関する詳細なメタデータを保存します。CDC の動作の詳細については、Microsoft ドキュメントの「変更データキャプチャ」を参照してください。

SQL Server の CDC オペレーションでは、`sys.servers` の ローカルサーバー (`server_id = 0`) および `SERVERPROPERTY('ServerName')` 識別子の間の値が一致する必要があります。RDS Custom for SQL Server は、インスタンスのライフサイクル全体でこの同期を自動的に維持し、メンテナンスまたは復旧オペレーション中にホストが置き換えられた場合でも、CDC が継続的に機能するようにします。

> **Important**
>
> マルチ AZ インスタンスのフェイルオーバー後、`SERVERPROPERTY('Servername')` 関数はネットワーク/コンピュータ名の変更を自動的に反映します。ただし、この `@@SERVERNAME` 関数は、`MSSQLSERVER` サービスが再起動されるまで古いサーバー名を保持します。@@SERVERNAME をクエリすると、フェイルオーバー後に以前のサーバー名が返されます。フェイルオーバー後に正確なサーバー名を取得するには、次の SQL クエリを使用します。
>
> ```
> SELECT name FROM sys.servers WHERE server_id=0
> ```

このクエリは、サービスの再起動を必要とせずに最新のサーバー名情報を提供します。

利用可能なリージョンとバージョン

CDC 機能は、RDS Custom for SQL Server が利用可能なすべての AWS リージョンにおいて、RDS Custom でサポートされているすべての SQL Server バージョンでサポートされています。RDS Custom for SQL Server のサポートされるバージョンと利用可能なリージョンの詳細については、「RDS Custom for SQL Server でサポートされているリージョンと DB エンジン」を参照してください。

要件と制限

RDS Custom for SQL Server に CDC を実装するときは、次の重要な考慮事項に注意してください。

- MS Replication などの機能を使用するように `sys.servers` で `@@SERVERNAME` および/またはローカルサーバーを手動で設定する場合で、`sys.servers` のローカルサーバー (`server_id = 0` を持つ) の値が `*.rds.amazonaws.com` または `*.awsrds.*.com` に一致する形式に設定されている場合、RDS Custom for SQL Server は `SERVERPROPERTY('ServerName')` と一致するように変更しようとしません。
- リモートログインまたはリンクされたサーバーが古いホスト名をアクティブに使用している間、RDS は `sys.servers` のローカルサーバー (`server_id = 0` を持つ) を新しいホスト名に変更することはできません。この制限は、次の 2 つのシナリオに適用されます。
  - リンクされたサーバーが、古いホスト名に関連付けられたリモートログインを使用してローカルサーバーへの接続を確立する場合
  - RDS Custom for SQL Server インスタンスがパブリッシャーまたはディストリビューターとして機能し、古いホスト名に関連付けられたログインをサブスクライバーインスタンスにリンクしている場合。

トラブルシューティング

古いサーバー名に関連付けられたリモートログインまたはリンクされたログインを特定するには、次のクエリを使用します。結果を検証し、これらのログインを削除して、CDC が適切に機能するようにします。

```
SELECT * FROM sys.remote_logins WHERE server_id=0
```

or

```
select sss.srvname,ssp.name,srl.remote_name  from sys.server_principals ssp
inner join sys.remote_logins srl on srl.local_principal_id=ssp.principal_id
inner join sys.sysservers sss  on srl.server_id = sss.srvid
where sss.srvname = @@SERVERNAME
```

Amazon RDS Custom for SQL Server の環境設定

Amazon RDS Custom for SQL Server DB インスタンスの DB インスタンスを作成および管理する前に、次のタスクを実行してください。

目次

- RDS Custom for SQL Server を設定するための前提条件
  - AWS マネジメントコンソール を使用したインスタンスプロファイルの自動作成
- ステップ 1: IAM プリンシパルに必要なアクセス許可を付与する
- ステップ 2: ネットワーキング、インスタンスプロファイル、および暗号化を構成する
  - による設定CloudFormation
    - CloudFormation に必要なパラメータ
    - CloudFormation テンプレートファイルをダウンロードする
    - CloudFormation を使用したリソースの設定
  - 手動設定
    - 対称暗号化 AWS KMS キーであることを確認します。
    - IAM ロールおよびインスタンスプロファイルをマニュアルで作成する
      - AWSRDSCustomSQLServerInstanceRole の IAM ロールを作成します。
      - AWSRDSCustomsqlServerInstanceRole にアクセスポリシーを追加します。
      - RDS Custom for SQL Server インスタンスプロファイルを作成する
      - RDS Custom の SQL Server インスタンスプロファイルに AWSRDSCustomSQLServerInstanceRole を追加します。
    - VPC をマニュアルで設定する
      - 次のように VPC セキュリティグループを設定します。
      - 非独立型 AWS のサービス のエンドポイント設定
      - インスタンスメタデータサービスの設定
- クロスインスタンスに関する制限

> **Note**
>
> 前提条件を設定し Amazon RDS Custom for SQL Server を起動する方法のステップバイステップのチュートリアルについては、ブログ投稿の「Get started with Amazon RDS Custom for SQL Server using an CloudFormation template (Network setup)」および「Explore the

prerequisites required to create an Amazon RDS Custom for SQL Server instance」を参照してください。

RDS Custom for SQL Server を設定するための前提条件

RDS Custom for SQL Server DB インスタンスを作成する前に、ご使用の環境がこのトピックで説明する要件を満たしていることを確認してください。CloudFormation テンプレートを使用して、AWS アカウント に前提条件を設定することもできます。詳細については、による設定CloudFormationを参照してください。

RDS Custom for SQL Server では、以下の前提条件を設定する必要があります。

- インスタンス作成に必要な AWS Identity and Access Management (IAM) アクセス許可を設定します。これは、RDS に `create-db-instance` リクエストを行うために必要な AWS Identity and Access Management (IAM) ユーザーまたはロールです。
- RDS Custom for SQL Server DB インスタンスに必要な前提条件リソースを設定します。
  - RDS Custom インスタンスの暗号化に必要な AWS KMS キーを設定します。RDS Custom では、インスタンスの作成時に、暗号化にカスタマーマネージドキーが必要です。KMS キー ARN、ID、エイリアス ARN、またはエイリアス名は、RDS Custom DB インスタンスを作成するリクエストで `kms-key-id` パラメータとして渡されます。
  - RDS Custom for SQL Server DB インスタンスに必要なアクセス許可を設定します。RDS Custom は、作成時にインスタンスプロファイルを DB インスタンスにアタッチし、DB インスタンスでのオートメーションに使用します。インスタンスプロファイル名は、RDS Custom 作成リクエストで `custom-iam-instance-profile` に設定されます。AWS マネジメントコンソール からインスタンスプロファイルを作成するか、インスタンスプロファイルを手動で作成できます。詳細については、「AWS マネジメントコンソール を使用したインスタンスプロファイルの自動作成」および「IAM ロールおよびインスタンスプロファイルをマニュアルで作成する」を参照してください。
  - RDS Custom for SQL Server の前提条件に従って、ネットワーク設定を構成します。RDS Custom インスタンスは、インスタンスの作成時に指定したサブネット (DB サブネットグループで設定) にあります。これらのサブネットは、RDS Custom インスタンスが RDS オートメーションに必要なサービスと通信できるように設定する必要があります。

> **Note**
>
> 上記の要件については、アカウントレベルのアクセス許可を制限するサービスコントロールポリシー (SCP) がないことを確認してください。
> 使用しているアカウントが AWS 組織の一部の場合、アカウントレベルのアクセス許可を制限するサービスコントロールポリシー (SCP) がある場合があります。次の手順を使用して作成するユーザーおよびロールのアクセス許可が SCP によって制限されないことを確認します。
> SCP の詳細については、AWS Organizations ユーザーガイド の「サービスコントロールポリシー (SCP)」を参照してください。describe-organization AWS CLI コマンドを使用して、アカウントが AWS 組織の一部かどうかをチェックします。
> AWS Organizations の詳細については、AWS Organizations ユーザーガイドの「AWS Organizations の概要」を参照してください。

RDS Custom for SQL Server に適用される一般的な要件については、RDS Custom for SQL Server の一般的な要件を参照してください。

AWS マネジメントコンソール を使用したインスタンスプロファイルの自動作成

RDS Custom では、RDS Custom for SQL Server DB インスタンスを起動するためのインスタンスプロファイルを作成し設定する必要があります。AWS マネジメントコンソール を使用して、1 つのステップで新しいインスタンスプロファイルを作成しアタッチします。このオプションは、[データベースの作成]、[スナップショットの復元]、[特定時点への復元] コンソールページの「RDS Custom セキュリティ」セクションにあります。[新しいインスタンスプロファイルを作成] を選択して、インスタンスプロファイル名のサフィックスを指定します。AWS マネジメントコンソール は、RDS Custom オートメーションタスクに必要なアクセス許可を持つ新しいインスタンスプロファイルを作成します。新しいインスタンスプロファイルを自動的に作成するには、ログインしている AWS マネジメントコンソール ユーザーに `iam:CreateInstanceProfile`、`iam:AddRoleToInstanceProfile`、`iam:CreateRole`、`iam:Atta` のアクセス許可が必要です。

> **Note**
>
> このオプションは、AWS マネジメントコンソール でのみ使用できます。CLI または SDK を使用している場合は、RDS Custom が提供する CloudFormation テンプレートを使用する

か、インスタンスプロファイルを手動で作成します。詳細については、「IAM ロールおよびインスタンスプロファイルをマニュアルで作成する」を参照してください。

ステップ 1: IAM プリンシパルに必要なアクセス許可を付与する

RDS Custom インスタンスを作成するために必要なキーがあることを確認してください。コンソールまたは CLI を使用して RDS Custom for SQL Server DB インスタンスを作成するための IAM ロールまたは IAM ユーザー (IAM プリンシパル) には、DB インスタンスを正常に作成するために、次のいずれかのポリシーが必要です。

- `AdministratorAccess` ポリシー
- 追加のアクセス許可を持つ`AmazonRDSFullAccess`ポリシーです。

```
iam:SimulatePrincipalPolicy
cloudtrail:CreateTrail
cloudtrail:StartLogging
s3:CreateBucket
s3:PutBucketPolicy
s3:PutBucketObjectLockConfiguration
s3:PutBucketVersioning
kms:CreateGrant
kms:DescribeKey
kms:Decrypt
kms:ReEncryptFrom
kms:ReEncryptTo
kms:GenerateDataKeyWithoutPlaintext
kms:GenerateDataKey
ec2:DescribeImages
ec2:RunInstances
ec2:CreateTags
```

RDS Custom は、インスタンスの作成時にこれらのアクセス許可を使用します。これらのアクセス許可は、RDS Custom オペレーションに必要なアカウントのリソースを設定します。

`kms:CreateGrant` アクセス許可の詳細については、「AWS KMS key 管理」を参照してください。

次のサンプル JSON ポリシーにより、必要なアクセス許可が付与されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ValidateIamRole",
            "Effect": "Allow",
            "Action": "iam:SimulatePrincipalPolicy",
            "Resource": "*"
        },
        {
            "Sid": "CreateCloudTrail",
            "Effect": "Allow",
            "Action": [
                "cloudtrail:CreateTrail",
                "cloudtrail:StartLogging"
            ],
            "Resource": "arn:aws:cloudtrail:*:*:trail/do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:PutBucketObjectLockConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

カスタムエンジンバージョン (CEV) を使用するには、IAM プリンシパルに次の追加のアクセス許可が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureKmsKeyEncryptionPermission",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key_id"
        },
        {
            "Sid": "CreateEc2Instance",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:RunInstances",
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

111122223333 をインスタンスの作成に使用しているアカウント ID に置き換えます。*us-east-1* をインスタンスの作成に使用している AWS リージョンに置き換えます。*key_id* を、カスタマーマネージド型キー ID に置き換えます。必要に応じて複数のキーを追加できます。

EC2 インスタンスの起動に必要なリソースレベルのアクセス許可の詳細については、「インスタンスの起動 (RunInstances)」を参照してください。

また、IAM プリンシパルにはIAM ロールに対する `iam:PassRole` アクセス許可が必要です。Custom DB インスタンスを作成するには、これをリクエストの `custom-iam-instance-profile` パラメータで渡されたインスタンスプロファイルにアタッチする必要があります。インスタンスプロファイルとそのアタッチされたロールは、後に ステップ 2: ネットワーキング、インスタンスプロファイル、および暗号化を構成する で作成されます。

> **Note**
>
> 以前にリストされたアクセス許可が、IAM プリンシパルに関連付けられたサービスコントロールポリシー (SCP)、アクセス許可の境界、またはセッションポリシーによって制限されていないことを確認します。

ステップ 2: ネットワーキング、インスタンスプロファイル、および暗号化を構成する

IAM インスタンスプロファイルのロール、仮想プライベートクラウド (VPC)、および AWS KMS 対称暗号化キーは、次のプロセスのいずれかを使用して設定できます。

- による設定CloudFormation (推奨)
- 手動設定

> **Note**
>
> アカウントが AWS Organizations の一部の場合は、インスタンスプロファイルロールに必要なアクセス許可がサービスコントロールポリシー (SCP) によって制限されていないことを確認します。
> このトピックのネットワーク構成は、パブリックアクセスが可能でない DB インスタンスで最適に動作するように設計されています。VPC の外部から DB インスタンスに直接接続することはできません。

による設定CloudFormation

セットアップを簡素化するために、CloudFormation スタックの作成に CloudFormation テンプレートファイルを使用できます。CloudFormation テンプレートは、RDS Custom の要件に従って、すべてのネットワーク、インスタンスプロファイル、および暗号化リソースを作成します。

スタックの作成方法については、CloudFormationユーザーガイドの「CloudFormation コンソールでのスタックの作成」を参照してください。

CloudFormation テンプレートを使用して Amazon RDS Custom for SQL Server を起動する方法のチュートリアルについては、AWS データベースブログの「CloudFormation テンプレートを使用して Amazon RDS Custom for SQL Server を開始する」を参照してください。

トピック

- CloudFormation に必要なパラメータ
- CloudFormation テンプレートファイルをダウンロードする
- CloudFormation を使用したリソースの設定

CloudFormation に必要なパラメータ

CloudFormation で RDS Custom の前提条件となるリソースを設定するには、次のパラメータが必要です。

パラメータグループ	パラメータ名	デフォルト値	説明
可用性の設定	前提条件設定の可用性設定を選択する	マルチ AZ	RDS Custom インスタンスのシングル AZ 構成またはマルチ AZ 構成で前提条件を設定するかどうかを指定します。1 つ以上のマルチ AZ DB インスタンスが必要な場合は、マルチ AZ 構成を使用する必要があります。
ネットワーク構成	VPC の IPv4 CIDR ブロック	10.0.0.0/16	VPC の IPv4 CIDR ブロック (または IP アドレス範囲) を指定します。この VPC は、RDS Custom DB インスタンスを作成

パラメータグループ	パラメータ名	デフォルト値	説明
			し使用するように設定されています。
	プライベートサブネット 1 (2 つのサブネットのうちの 1) の IPv4 CIDR ブロック	10.0.128.0/20	1 つ目のプライベートサブネットの IPv4 CIDR ブロック (または IP アドレス範囲) を指定します。これは、RDS Custom DB インスタンスを作成できる 2 つのサブネットのうちの 1 つです。このプライベートサブネットは、インターネットにアクセスすることはできません。
	プライベートサブネット 2 (2 つのサブネットのうちの 2) の IPv4 CIDR ブロック	10.0.144.0/20	2 つ目のプライベートサブネットの IPv4 CIDR ブロック (または IP アドレス範囲) を指定します。これは、RDS Custom DB インスタンスを作成できる 2 つのサブネットのうちの 1 つです。このプライベートサブネットは、インターネットにアクセスすることはできません。

パラメータグループ	パラメータ名	デフォルト値	説明
	パブリックサブネットの IPv4 CIDR ブロック	10.0.0.0/20	パブリックサブネットの IPv4 CIDR ブロック (または IP アドレス範囲) を指定します。これは、EC2 インスタンスが接続できる RDS Custom DB インスタンスを作成できるサブネットのうちの 1 つです。このパブリックサブネットは、インターネットにアクセスできます。
RDP アクセス設定	ソースの IPv4 CIDR ブロック	-	ソースの IPv4 CIDR ブロック (または IP アドレス範囲) を指定します。これは、パブリックサブネットで EC2 インスタンスへの RDP 接続を行う IP 範囲です。設定されていない場合、EC2 インスタンスへの RDP 接続は設定されません。

パラメータグループ	パラメータ名	デフォルト値	説明
	RDS Custom for SQL Server インスタンスへの RDP アクセスをセットアップする	いいえ	EC2 インスタンスから RDS Custom for SQL Server インスタンスへの RDP 接続を有効にするかどうかを指定します。デフォルトでは、EC2 インスタンスから DB インスタンスへの RDP 接続は設定されていません。

CloudFormation で作成されるリソース

デフォルトの設定で CloudFormation スタックを正常に作成すると、スタックは AWS アカウント で以下のリソースを作成します。

- RDS Custom で管理されるデータを暗号化するための対称暗号化 KMS キー。
- インスタンスプロファイルは、RDS Custom に必要なアクセス許可を提供するために `AmazonRDSCustomInstanceProfileRolePolicy` を持つ IAM ロールに関連付けられています。詳細については、「AWS Managed Policy リファレンスガイド」の「AmazonRDSCustomServiceRolePolicy」を参照してください。
- CloudFormation パラメータとして CIDR 範囲が指定された VPC。デフォルト値は `10.0.0.0/16` です。
- パラメータで指定された CIDR 範囲を持つ 2 つのプライベートサブネット、および AWS リージョン の異なる 2 つのアベイラビリティーゾーン。サブネット CIDR のデフォルト値は `10.0.128.0/20` と `10.0.144.0/20` です。
- パラメータで指定された CIDR 範囲を持つ 1 つのパブリックサブネット。サブネット CIDR のデフォルト値は 10.0.0.0/20 です。EC2 インスタンスはこのサブネットに存在し、RDS Custom インスタンスへの接続に使用できます。
- Amazon ドメインネームシステム (DNS) サーバーへドメイン名が解決された VPC の DHCP オプション設定。

- 2 つのプライベートサブネットに関連付けられており、インターネットアクセスがないルートテーブル。
- パブリックサブネットに関連付けられており、インターネットアクセスがあるルートテーブル。
- パブリックサブネットへのインターネットアクセスを許可する VPC に関連付けられたインターネットゲートウェイ。
- 2 つのプライベートサブネットに関連付けるネットワークアクセスコントロールリスト (ACL) と HTTPS および VPC の DB ポートに制限されたアクセス。
- RDS Custom インスタンスに関連付けられた VPC セキュリティグループ。アクセスは、RDS Custom で必要な AWS のサービス エンドポイントへのアウントバウンド HTTPS、および EC2 インスタンスセキュリティグループからのインバウンド DB ポートに制限されます。
- パブリックサブネット内の EC2 インスタンスは、VPC セキュリティグループに関連付けられます。RDS Custom インスタンスセキュリティグループへのアウトバウンド DB ポートのアクセスが制限されます。
- RDS Custom で必要とされる AWS のサービス エンドポイント用に作成された VPC エンドポイントに関連付けらる VPC セキュリティグループ。
- RDS カスタムインスタンスが作成される DB サブネットグループ。このテンプレートによって作成された 2 つのプライベートサブネットが DB サブネットグループに追加されます。
- RDS Custom で必要とされる各 AWS のサービス エンドポイントの VPC エンドポイント。

可用性構成を multi-az に設定すると、上記に加えて次のリソースが作成されます。

- プライベートサブネット間の通信を許可するネットワーク ACL ルール。
- RDS Custom インスタンスに関連付けられた VPC セキュリティグループ内のマルチ AZ ポートへのインバウンドおよびアウトバウンドアクセス。
- マルチ AZ 通信に必要な AWS サービスエンドポイントへの VPC エンドポイント。

さらに、RDP アクセス構成を設定すると、次のリソースが作成されます。

- ソース IP アドレスからパブリックサブネットへの RDP アクセス構成：
  - ソース IP からパブリックサブネットへの RDP 接続を許可するネットワーク ACL ルール。
  - EC2 インスタンスに関連付けられた VPC セキュリティグループへのソース IP から RDP ポートへの入力アクセス。
- パブリックサブネットの EC2 インスタンスからプライベートサブネットの RDS Custom インスタンスへの RDP アクセス構成：

- パブリックサブネットからプライベートサブネットへの RDP 接続を許可するネットワーク ACL ルール。
- EC2 インスタンスに関連付けられた VPC セキュリティグループから RDS Custom インスタンスに関連付けられた VPC セキュリティグループへの RDP ポートへのインバウンドアクセス。

RDS Custom for SQL Server の CloudFormation スタックを作成するには、以下の手順に従います。

CloudFormation テンプレートファイルをダウンロードする

テンプレートファイルをダウンロードするには

1. リンク custom-sqlserver-onboard.zip のコンテキスト (右クリック) メニューを開き、[Save Link As] (名前を付けてリンク先を保存) を選択します。
2. ファイルをパソコンに保存し、解凍します。

CloudFormation を使用したリソースの設定

CloudFormation を使用してリソースを設定するには

1. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. 「スタックの作成」ウィザードを起動し、[Create Stack] (スタックの作成) を選択します。

   [Create stack] (スタックの作成) が表示されます。
3. 前提条件 – テンプレートの準備で、テンプレートの準備完了を選択します。
4. [Specify template] (テンプレートの指定) ページで、以下を実行します。

   a. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。
   b. [ファイルを選択] に移動して、正しいファイルを選択します。
5. [次へ] を選択します。

   [Specify stack details] (DB 詳細の指定) ページが表示されます。
6. [スタック名] に「**`rds-custom-sqlserver`**」と入力します。
7. [Parameters] (パラメータ) では、以下を行います。

   a. デフォルトのオプションを保持するには、[Next] (次へ) を選択します。
   b. オプションを変更するには、適切な可用性構成を選択して、ネットワーク設定、RDP アクセス設定を選択し、[次へ] を選択します。

パラメータを変更する前に、各パラメータの説明をよくお読みください。

> **Note**
>
> この CloudFormation スタックに 1 つ以上のマルチ AZ インスタンスを作成する場合は、CloudFormation スタックパラメータ、[前提条件設定の可用性構成を選択する] が `Multi-AZ` に設定されていることを確認してください。CloudFormation スタックをシングル AZ として作成する場合は、最初のマルチ AZ インスタンスを作成する前に、CloudFormation スタックをマルチ AZ 設定に更新します。

8. [スタックオプションの設定] ページで、[次へ] を選択します。
9. 「rds-custom-sqlserver のレビュー」ページで、以下を実行します。
   a. [機能] で、[CloudFormation が IAM リソースを作成する可能性があることに同意する] チェックボックスをオンにします。
   b. [スタックの作成] を選択してください。

> **Note**
>
> この CloudFormation スタックから作成されたリソースは、リソースページから直接更新しないでください。CloudFormation テンプレートを使用したこれらのリソースの今後の更新ができなくなります。

CloudFormation は、RDS Custom for SQL Server が必要とするリソースを作成します。スタックの作成に失敗した場合は、[Events] (イベント) タブを表示して、失敗したリソースの作成とそのステータスの理由を確認します。

コンソール内のこの CloudFormation スタックの[Outputs] (出力) タブには、RDS Custom for SQL Server DB インスタンスを作成するためのパラメータとして渡されるすべてのリソースに関する情報が含まれている必要があります。CloudFormation によって RDS カスタム DB インスタンス用に作成された VPC セキュリティグループと DB サブネットグループを必ず使用してください。デフォルトでは、RDS はデフォルトの VPC セキュリティグループをアタッチしようと試みます。これには、必要なアクセス許可がない可能性があります。

CloudFormation を使用してリソースを作成した場合は、手動設定 はスキップできます。

CloudFormation スタックの更新

作成後に CloudFormation スタックの設定の一部を更新することもできます。更新できる設定は次のとおりです。

- RDS Custom for SQL Server の可用性構成
  - 前提条件設定の可用性構成を選択する: このパラメータを更新して、シングル AZ 構成とマルチ AZ 構成を切り替えます。この CloudFormation スタックを 1 つ以上のマルチ AZ インスタンスで使用している場合は、スタックを更新してマルチ AZ 構成を選択する必要があります。
- RDS Custom for SQL Server の RDP アクセス設定
  - ソースの IPv4 CIDR ブロック: このパラメータを更新することで、ソースの IPv4 CIDR ブロック (または IP アドレス範囲) を更新できます。このパラメータを空のままにすると、ソース CIDR ブロックからパブリックサブネットへの RDP アクセス設定が削除されます。
  - RDS Custom for SQL Server への RDP アクセスを設定する: EC2 インスタンスから RDS Custom for SQL Server インスタンスへの RDP 接続を有効または無効にします。

CloudFormation スタックの削除

リソースを使用するすべての RDS Custom インスタンスをスタックから削除した後、CloudFormation スタックを削除できます。RDS Custom は CloudFormation スタックを追跡しないため、スタックリソースを使用する DB インスタンスがある場合、スタックの削除をブロックしません。スタックを削除する際に、スタックリソースを使用する RDS Custom DB インスタンスがないことを確認してください。

> **Note**
>
> CloudFormation スタックを消去すると、KMS キーを除き、そのスタックで作成されたすべてのリソースが消去されます。KMS キーは pending-deletion の状態になり、30 日後に削除されます。KMS キーを保持するには、CancelKeyDeletion オペレーションを 30 日間の猶予期間中に実行します。

手動設定

リソースを手動で設定する場合は、以下のタスクを実行します。

> **Note**
>
> セットアップを簡素化するために、CloudFormation スタックの作成に、手動設定ではなく CloudFormation テンプレートファイルを使用できます。詳細については、「による設定 CloudFormation」を参照してください。
> AWS CLI を使用してこのセクションを完了することもできます。その場合は、最新の CLI をダウンロードしてインストールします。

トピック

- 対称暗号化 AWS KMS キーであることを確認します。
- IAM ロールおよびインスタンスプロファイルをマニュアルで作成する
- VPC をマニュアルで設定する

対称暗号化 AWS KMS キーであることを確認します。

RDS Custom には対称暗号化 AWS KMS key が必要です。SQL Server DB インスタンスの RDS Custom DB インスタンスを作成する際、KMS キー識別子を `kms-key-id` パラメータとして指定する必要があります。詳細については、「Amazon RDS Custom SQL Server の DB インスタンスの作成と接続」を参照してください。

次のオプションがあります。

- AWS アカウント に既存のカスタマーマネージド KMS キーがある場合は、RDS Custom で使用できます。これ以上の操作は不要です。
- RDS Custom エンジンのカスタマーマネージド対称暗号化 KMS キーを既に作成している場合は、同じ KMS キーを再利用できます。これ以上の操作は不要です。
- アカウントに既存のカスタマーマネージド対称暗号化 KMS キーがない場合は、AWS Key Management Service デベロッパーガイドの「キーの作成」の手順に従って KMS キーを作成します。
- CEV または RDS Custom DB インスタンスを作成していて、KMS キーが別の AWS アカウント にある場合は、必ず AWS CLI を使用してください。クロスアカウントの KMS キーでは AWS コンソールを使用できません。

> **Important**
>
> RDS Custom はAWSマネージド KMS キーに対応していません。

対称暗号化キーが、IAM インスタンスプロファイルの AWS Identity and Access Management (IAM) ロールに `kms:Decrypt` および `kms:GenerateDataKey` オペレーションへのアクセスを許可していることを確認してください。アカウントに新しい対称暗号化キーがある場合、変更は必要ありません。それ以外の場合は、対称暗号化キーのポリシーがこれらのオペレーションへのアクセスを許可していることを確認してください。

詳細については、「ステップ 4: RDS Custom for Oracle 用に IAM を設定する」を参照してください。

IAM ロールおよびインスタンスプロファイルをマニュアルで作成する

手動で作成したインスタンスプロファイルを使用して、RDS Custom インスタンスを起動できます。AWS マネジメントコンソール でインスタンスを作成する場合は、このセクションをスキップします。AWS マネジメントコンソール を使用すると、インスタンスプロファイルを作成して RDS Custom DB インスタンスにアタッチできます。詳細については、「AWS マネジメントコンソールを使用したインスタンスプロファイルの自動作成」を参照してください。

インスタンスプロファイルを手動で作成する際は、インスタンスプロファイル名を `custom-iam-instance-profile` パラメータとして `create-db-instance` CLI コマンドに渡します。RDS Custom は、インスタンスプロファイルに関連付けられたロールを使用してオートメーションを実行し、インスタンスを管理します。

RDS Custom for SQL Server の IAM インスタンスプロファイルと IAM ロールを作成するには

1. `AWSRDSCustomSQLServerInstanceRole`という名前の IAM ロールを作成し、Amazon EC2 にこのロールを引き受けるように許可する信頼ポリシーを使用します。
2. AWS 管理ポリシー `AmazonRDSCustomInstanceProfileRolePolicy` を `AWSRDSCustomSQLServerInstanceRole` に追加します。
3. `AWSRDSCustomSQLServerInstanceProfile` という名前のRDS Custom for SQL Server IAM インスタンスプロファイルを作成します。
4. `AWSRDSCustomSQLServerInstanceRole`をインスタンスプロファイルに追加します。

AWSRDSCustomSQLServerInstanceRole の IAM ロールを作成します。

以下の例では、AWSRDSCustomSQLServerInstanceRoleロールを作成します。信頼ポリシーにより、Amazon EC2 はロールを引き受けます。

```
aws iam create-role \
    --role-name AWSRDSCustomSQLServerInstanceRole \
    --assume-role-policy-document '{
        "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "sts:AssumeRole",
              "Effect": "Allow",
              "Principal": {
                  "Service": "ec2.amazonaws.com"
              }
            }
          ]
        }'
```

AWSRDSCustomsqlServerInstanceRole にアクセスポリシーを追加します。

必要なアクセス許可を付与するには、AWS マネージドポリシー AmazonRDSCustomInstanceProfileRolePolicy を AWSRDSCustomSQLServerInstanceRole にアタッチします。AmazonRDSCustomInstanceProfileRolePolicy は、RDS Custom インスタンスにメッセージを送受信し、さまざまなオートメーションアクションを実行することを許可します。

> **Note**
>
> アクセスポリシーのアクセス許可が、インスタンスプロファイルロールに関連付けられた SCP またはアクセス許可の境界によって制限されていないことを確認します。

次の例では、AWS マネージドポリシー AmazonRDSCustomInstanceProfileRolePolicy を AWSRDSCustomSQLServerInstanceRole ロールにアタッチします。

```
aws iam attach-role-policy \
    --role-name AWSRDSCustomSQLServerInstanceRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonRDSCustomInstanceProfileRolePolicy
```

RDS Custom for SQL Server インスタンスプロファイルを作成する

インスタンスプロファイルは、1 つの IAM ロールを含むコンテナです。RDS Custom はインスタンスプロファイルを使用してインスタンスにロールを渡します。

AWS マネジメントコンソール を使用して Amazon EC2 のロールを作成する場合、ロールが作成サれる際、コンソールはインスタンスプロファイルを自動的に作成し、そのインスタンスプロファイルにロールと同じ名前を付けます。インスタンスプロファイルを作成し、`AWSRDSCustomSQLServerInstanceProfile`と名前を付けます。

```
aws iam create-instance-profile \
    --instance-profile-name AWSRDSCustomSQLServerInstanceProfile
```

RDS Custom の SQL Server インスタンスプロファイルに AWSRDSCustomSQLServerInstanceRole を追加します。

以前に作成した `AWSRDSCustomSQLServerInstanceProfile` プロファイルに `AWSRDSCustomInstanceRoleForRdsCustomInstance` ロールを追加します。

```
aws iam add-role-to-instance-profile \
    --instance-profile-name AWSRDSCustomSQLServerInstanceProfile \
    --role-name AWSRDSCustomSQLServerInstanceRole
```

VPC をマニュアルで設定する

RDS Custom DB インスタンスは、Amazon EC2 インスタンスまたは Amazon RDS インスタンスと同様に、Amazon VPC サービスに基づく Virtual Private Cloud (VPC) にあります。独自の VPC を提供して設定します。したがって、インスタンスのネットワーク設定を完全に制御できます。

RDS Custom は DB インスタンスから他の AWS のサービス に通信を送信します。RDS Custom DB インスタンスを作成するサブネットから以下のサービスにアクセスできることを確認します。

- Amazon CloudWatch (`com.amazonaws.`*`region`*`.monitoring`)
- Amazon CloudWatch Logs (`com.amazonaws.`*`region`*`.logs`)
- Amazon CloudWatch Events (`com.amazonaws.`*`region`*`.events`)
- Amazon EC2 (`com.amazonaws.`*`region`*`.ec2` および `com.amazonaws.`*`region`*`.ec2messages`)
- Amazon S3 (`com.amazonaws.`*`region`*`.s3`)
- AWS Secrets Manager (`com.amazonaws.`*`region`*`.secretsmanager`)

- AWS Systems Manager (com.amazonaws.*region*.ssm および com.amazonaws.*region*.ssmmessages)

マルチ AZ 配置を作成する場合

- Amazon Simple Queue Service (com.amazonaws.*region*.sqs)

RDS Custom は必要なサービスと通信できない場合、次のイベントを公開します。

```
Database instance in incompatible-network. SSM Agent connection not available. Amazon
 RDS can't connect to the dependent AWS services.
```

```
Database instance in incompatible-network. Amazon RDS can't connect to dependent AWS
 services. Make sure port 443 (HTTPS) allows outbound connections, and try again.
 "Failed to connect to the following services: s3 events"
```

incompatible-network エラーを避けるため、RDS Custom DB インスタンスと AWS のサービス間の通信に関与している VPC コンポーネントが次の要件を満たしていることを確認します。

- DB インスタンスは、ポート 443 で他の AWS のサービス へのアウトバウンド接続を行うことができます。
- VPC は、RDS Custom DB インスタンスから発信されるリクエストへの受信レスポンスを許可します。
- RDS Custom は、各 AWS のサービス に対してエンドポイントのドメイン名を正しく解決できます。

異なる RDS Custom DB エンジン用に VPC を既に設定している場合は、その VPC を再利用してこのプロセスをスキップできます。

トピック

- 次のように VPC セキュリティグループを設定します。
- 非独立型 AWS のサービス のエンドポイント設定
- インスタンスメタデータサービスの設定

次のように VPC セキュリティグループを設定します。

セキュリティグループは、VPC インスタンスの仮想ファイアウォールとして機能し、インバウンドトラフィックとアウトバウンドトラフィックを制御します。RDS Custom DB インスタンスには、インスタンスを保護するネットワークインターフェースにアタッチされたセキュリティグループがあります。セキュリティグループが RDS Custom と他の AWS のサービス との間の HTTPS トラフィックを許可していることを確認します。このセキュリティグループをインスタンス作成リクエストの `vpc-security-group-ids` パラメータとして渡します。

RDS Custom のセキュリティグループを設定するには

1. AWS マネジメントコンソール にサインインして、Amazon VPC コンソール (https://console.aws.amazon.com/vpc) を開きます。
2. RDS Custom でデフォルトのセキュリティグループを使用するか、独自のセキュリティグループを作成することを許可します。

   詳細な手順については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。
3. セキュリティグループがポート 443 でアウトバウンド接続を許可していることを確認します。RDS Custom は、非独立型の AWS のサービス と通信するためにこのポートが必要です。
4. プライベート VPC があり VPC エンドポイントを使用する場合は、DB インスタンスに関連付けられたセキュリティグループが VPC エンドポイントにポート 443 でアウトバウンド接続を許可していることを確認してください。また、VPC エンドポイントに関連付けられたセキュリティグループが、DB インスタンスからポート 443 でインバウンド接続を許可していることを確認します。

   着信接続が許可されていない場合、RDS Custom インスタンスは AWS Systems Manager および Amazon EC2 エンドポイントに接続できません。詳細については、AWS Systems Manager ユーザーガイドの「仮想プライベートクラウドエンドポイントの作成」を参照してください。
5. RDS Custom for SQL Server マルチ AZ インスタンスの場合、DB インスタンスに関連付けられたセキュリティグループで、このセキュリティグループ自体へのインバウンド接続とアウトバウンド接続がポート 1120 で許可されていることを確認します。これは、マルチ AZ RDS Custom for SQL Server DB インスタンスでのピアホスト接続に必要です。

VPC セキュリティグループの詳細については、Amazon VPC デベロッパーガイドの「VPC のセキュリティグループ」を参照してください。

非独立型 AWS のサービス のエンドポイント設定

次の手順を使用して、すべてのサービスのエンドポイントを VPC に追加することをお勧めします。ただし、VPC がAWSサービスエンドポイントと通信する任意のソリューションを使用できます。例えば、ネットワークアドレス変換 (NAT) を使用したり、AWS Direct Connect。

RDS Custom が動作する AWS のサービス のエンドポイントを設定するには

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. ナビゲーションバーで、リージョンセレクターを使用して、AWS リージョンを選択します。
3. ナビゲーションペインで、[エンドポイント] を選択します。メインペインで、[Create Endpoint (エンドポイントの作成)] を選択します。
4. [Service category] (サービスカテゴリ) で、AWS のサービス を選択します。
5. サービス名で、表に表示されたエンドポイントを選択します。
6. [VPC] で、ユーザーの VPC を選択します。
7. [Subnets (サブネット)] で、追加するサブネットを各アベイラビリティーゾーンから選択します。

   VPC エンドポイントは、複数のアベイラビリティーゾーンにまたがることができます。AWS は、VPC エンドポイントの Elastic Network Interface を、選択された各サブネットで作成します。各ネットワークインターフェースには ドメインネームシステム (DNS) ホスト名とプライベート IP アドレスがあります。
8. 「セキュリティグループ」 で、セキュリティグループを選択または作成します。

   セキュリティグループを使用すると、エンドポイントへのアクセスを制御できます。これは、ファイアウォールを使用するのに似ています。セキュリティグループで、DB インスタンスからのインバウンド接続がポート 443 で許可されていることを確認してください。VPC セキュリティグループの詳細については、Amazon VPC ユーザーガイドの「VPC のセキュリティグループ」を参照してください。
9. オプションで、VPC エンドポイントにポリシーをアタッチできます。エンドポイントポリシーは、接続している AWS のサービス へのアクセスを制御します。デフォルトのポリシーでは、すべてのリクエストがエンドポイントを通過することを許可します。カスタムポリシーを使用している場合は、DB インスタンスからのリクエストがポリシーで許可されていることを確認してください。
10. エンドポイントの作成 を選択します。

次の表では、VPC がアウトバウンド通信に必要なエンドポイントのリストを検索する方法について説明します。

サービス	エンドポイントフォーマット	メモとリンク
AWS Systems Manager	以下のエンドポイント形式を使用します。 • ssm.*region*.amazonaws.com • ssmmessages. *region*.amazonaws.com	各リージョンのエンドポイントのリストについては、Amazon Web Services 全般のリファレンス の「AWS Systems Manager エンドポイントとクォータ」を参照してください。
AWS Secrets Manager	エンドポイント形式 secretsmanager. *region*.amazonaws.com を使用します。	各リージョンのエンドポイントのリストについては、Amazon Web Services 全般のリファレンス の「AWS Secrets Manager エンドポイントとクォータ」を参照してください。
Amazon CloudWatch	以下のエンドポイント形式を使用します。 • CloudWatch メトリクスの場合は、monitoring. *region*.amazonaws.com を使用します • CloudWatch Eventsの場合は、events.*region*.amazonaws.com を使用します • CloudWatch Logs の場合は、logs.*region*.amazonaws.com を使用します	各リージョンのエンドポイントのリストについては、以下を参照してください。 • Amazon Web Services 全般のリファレンス の「Amazon CloudWatch エンドポイントとクォータ」 • Amazon Web Services 全般のリファレンス の「Amazon CloudWatch Logs エンドポイントとクォータ」 • Amazon Web Services 全般のリファレンス の「Amazon CloudWatch Events エンドポイントとクォータ」

サービス	エンドポイントフォーマット	メモとリンク
Amazon EC2	以下のエンドポイント形式を使用します。 • ec2.*region*.amazonaws.com • ec2messages. *region*.amazonaws.com	各リージョンのエンドポイントのリストについては、Amazon Web Services 全般のリファレンスの「Amazon EC2 エンドポイントとクォータ」を参照してください。
Amazon S3	エンドポイントの形式s3.*region*.amazonaws.com を使用します。	各リージョンのエンドポイントのリストについては、Amazon Web Services 全般のリファレンスの「Amazon Simple Storage Service エンドポイントとクォータ」を参照してください。 Amazon S3 のゲートウェイエンドポイントの詳細については、Amazon VPC デベロッパーガイドのAmazon S3 におけるエンドポイントを参照してください。 アクセスポイントの作成方法については、「Amazon VPC デベロッパーガイド」の「Creating access points」(アクセスポイントの作成)を参照してください。 Amazon S3 のゲートウェイエンドポイントを作成する方法については、「ゲートウェイ VPC エンドポイント」を参照してください。
Amazon Simple Queue Service	エンドポイント形式 sqs.*region*.amazonaws.com を使用します。	各リージョンのエンドポイントのリストについては、「Amazon Simple Queue Service endpoints and quotas」を参照してください。

インスタンスメタデータサービスの設定

以下を実行して、インスタンスが に接続できることを確認します。

- インスタンスメタデータにアクセスするために、Instance Metadata Service のバージョン 2 (IMDSv2) を使用します。
- ポート 80 (HTTP) を介して IMDS リンク IP アドレスへのアウトバウンド通信を許可します。
- `http://169.254.169.254`からのインスタンスメタデータのリクエスト、IMDSv2 のリンクです。

詳細については、「Amazon EC2 ユーザーガイド」の「IMDSv2 の使用」を参照してください。

クロスインスタンスに関する制限

上記の手順に従ってインスタンスプロファイルを作成する際、AWS マネージドポリシー `AmazonRDSCustomInstanceProfileRolePolicy` を使用して、インスタンスの管理とモニタリングの自動化を可能にする必要なアクセス許可を RDS Custom に付与します。マネージドポリシーは、RDS Custom がオートメーションを実行するために必要なリソースにのみアクセス許可を付与します。マネージドポリシーを使用して新機能をサポートし、手動操作なしで既存のインスタンスプロファイルに自動的に適用されるセキュリティ要件に対応することをお勧めします。詳細については、「AWS マネージドポリシー: AmazonRDSCustom Instance ProfileRolePolicy」を参照してください。

`AmazonRDSCustomInstanceProfileRolePolicy` マネージドポリシーは、インスタンスプロファイルのクロスアカウントアクセスを制限しますが、同じアカウント内の RDS Custom インスタンス間で一部の RDS Custom マネージドリソースへのアクセスを許可する場合があります。必要な場合、アクセス許可の境界を使用して、インスタンス間のアクセスをさらに制限できます。アクセス許可の境界ポリシーは、アイデンティティベースのポリシーでエンティティに付与できるアクセス許可の上限を定義しますが、アクセス許可は付与しません。詳細については、「境界を設定した場合の有効なアクセス許可の評価」を参照してください。

例えば、次の境界ポリシーでは、特定の AWS KMS キーへのアクセスをインスタンスプロファイルロールに制限し、異なる AWS KMS キーを使用しているインスタンス間で RDS Custom マネージドリソースへのアクセスを制限します。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyOtherKmsKeyAccess",
            "Effect": "Deny",
            "Action": "kms:*",
            "NotResource": "arn:aws:kms:us-east-1:111122223333:key/KMS_key_ID"
        }
    ]
}
```

> **Note**
>
> アクセス許可の境界によって、AmazonRDSCustomInstanceProfileRolePolicy が RDS Custom に付与するアクセス許可がブロックされていないことを確認します。

RDS Custom for SQL Server での Bring Your Own Media

RDS Custom for SQL Server は、ライセンス込み (LI) および Bring Your Own Media (BYOM) の 2 つのライセンスモデルをサポートしています。

BYOM では、次のことを実行できます。

1. サポートされている累積更新 (CU) を含む独自の Microsoft SQL Server バイナリを AWS EC2 Windows AMI に用意してインストールします。
2. AMI をゴールデンイメージとして保存します。ゴールデンイメージは、カスタムエンジンバージョン (CEV) の作成に使用できるテンプレートです。
3. ゴールデンイメージから CEV を作成しします。
4. CEV を使用して新しい RDS Custom for SQL Server DB インスタンスを作成します。

その後、Amazon RDS は DB インスタンスをユーザーに代わって管理します。

> **Note**
>
> SQL Server DB インスタンス用のライセンス込み (LI) RDS Custom も所有している場合、この DB インスタンスの SQL Server ソフトウェアを BYOM で使用することはできません。独自の SQL Server バイナリを BYOM に持ち込む必要があります。

RDS Custom for SQL Server の BYOM の一般的な要件

RDS Custom for SQL Server のカスタムエンジンバージョンと同じ一般的な要件が BYOM にも適用されます。詳細については、「RDS Custom for SQL Server CEV の一般的な要件」を参照してください。

BYOM を使用する場合は、次の追加要件を満たしていることを確認してください。

- SQL Server 2022 または 2019 Enterprise、Standard、または Developer エディションの、サポートされているエディションのいずれかを使用します。
- SQL Server システム管理者 (SA) サーバーロール権限を `NT AUTHORITY\SYSTEM` に付与します。
- SSM 接続を許可するには、TCP ポート 1433 と UDP ポート 1434 を開きます。

RDS Custom for SQL Server の制限

RDS Custom for SQL Server と同じ一般的な制限が BYOM にも適用されます。詳細については、「Amazon RDS Custom for SQL Server の要件と制限」を参照してください。

BYOM では、さらに次の制限が適用されます。

- デフォルトの SQL Server インスタンス (MSSQLSERVER) のみがサポートされています。名前付き SQL Server インスタンスはサポートされていません。RDS Custom for SQL Server は、デフォルトの SQL Server インスタンスのみを検出して監視します。
- 各 AMI でサポートされている SQL Server のインストールは 1 つだけです。異なる SQL Server バージョンの複数インストールはサポートされていません。
- SQL Server Web エディションは BYOM ではサポートされていません。
- SQL Server エディションの評価バージョンは BYOM ではサポートされていません。SQL Server をインストールするときは、評価バージョンを使用するためのチェックボックスを選択しないでください。
- 機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョン によって異なります。詳細については、「RDS Custom for SQL Server CEV の利用可能なリージョン」および「RDS Custom for SQL Server CEV のバージョンサポート」を参照してください。

RDS Custom for SQL Server DB インスタンスの BYOM での作成

BYOM で RDS Custom SQL Server DB インスタンスを準備して作成するには、「Bring Your Own Media (BYOM) を使用した CEV の準備」を参照してください。

RDS Custom for SQL Server のカスタムエンジンバージョンの使用

RDS Custom for SQL Server のカスタムエンジンバージョン (CEV) は Microsoft SQL Server を含む Amazon マシンイメージ (AMI) です。

CEV ワークフローの基本的な手順は次のとおりです。

1. CEV のベースイメージとして使用する AWS EC2 Windows AMI を選択します。プリインストールされている Microsoft SQL Server を使用するか、Bring Your Own Media (BYOM) を使用して SQL Server を自分でインストールするかを選択できます。
2. オペレーティングシステム (OS) に他のソフトウェアをインストールし、企業のニーズに合わせて OS と SQL Server の構成をカスタマイズします。
3. AMI をゴールデンイメージとして保存します
4. ゴールデンイメージからカスタムエンジンバージョン (CEV) を作成します。
5. CEV を使用して新しい RDS Custom for SQL Server DB インスタンスを作成します。

その後、Amazon RDS はこれらの DB インスタンスをお客様に代わって管理します。

CEV を使用すると、OS とデータベースの任意のベースライン設定を維持できます。CEV を使用すると、サードパーティーエージェントのインストールやその他の OS カスタマイズなどのホスト設定が RDS Custom for SQL Server DB インスタンスで確実に保持されます。CEV を使用すると、同じ設定で RDS Custom for SQL Server DB インスタンスのフリートをすばやくデプロイできます。

トピック

- RDS Custom for SQL Server の CEV を作成する準備
- RDS Custom for SQL Server の CEV の作成
- RDS Custom for SQL Server の CEV の変更
- Amazon RDS Custom for SQL Server の CEV 詳細を表示する
- RDS Custom for SQL Server の CEV の削除

RDS Custom for SQL Server の CEV を作成する準備

CEV は、ライセンス込み (LI) の Microsoft SQL Server がプリインストールされている Amazon マシンイメージ (AMI) を使用するか、独自の SQL Server インストールメディア (BYOM) をインストールする AMI で作成できます。

CEV の準備

Bring Your Own Media (BYOM) またはプリインストールされた Microsoft SQL Server (LI) を使用して CEV を作成するには、次の手順に従います。

Bring Your Own Media (BYOM) を使用した CEV の準備

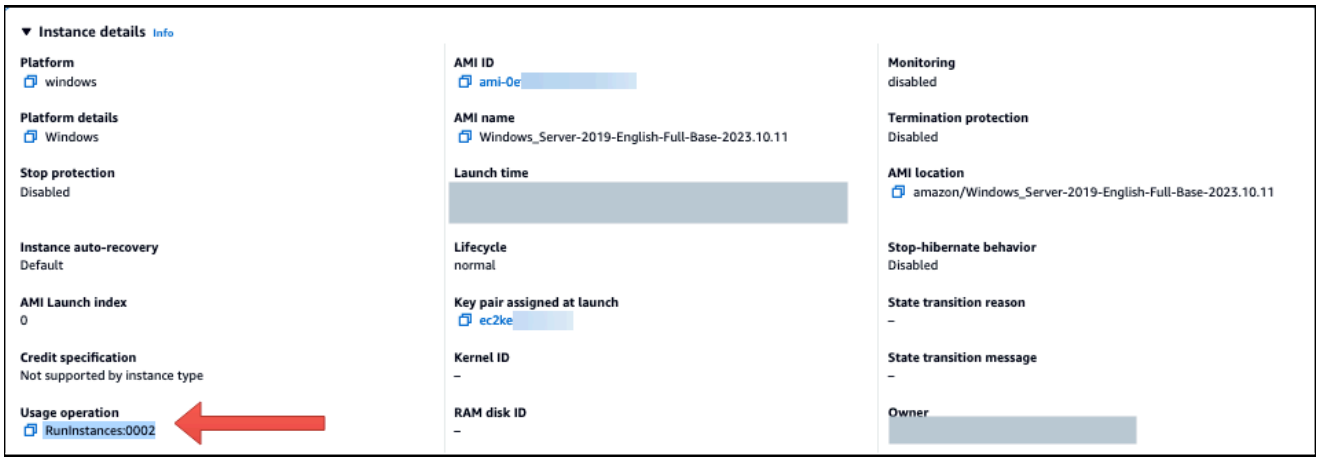
以下の手順では、例として Windows Server 2019 Base での AMI を使用します。

BYOM を使用して CEV を作成するには

1. Amazon EC2 コンソールで、[インスタンスを起動] を選択します。
2. [名前] に、インスタンスの名前を入力します。
3. [クイックスタート] で、[Windows] を選択します。
4. [Microsoft Windows Server 2019 Base] を選択します。
5. 適切なインスタンスタイプ、キーペア、ネットワークとストレージの設定を選択し、インスタンスを起動します。
6. EC2 インスタンスを起動または作成したら、ステップ 4 で正しい Windows AMI が選択されていることを確認します。

   a. Amazon EC2 コンソールで、EC2 インスタンスを選択します。

   b. [詳細] セクションで、[使用オペレーション] チェックボックスをオンにし、[RunInstances:0002] に設定されていることを確認します。



7. EC2 インスタンスにログインし、SQL Server インストールメディアをそのインスタンスにコピーします。

> **Note**
>
> SQL Server Developer エディションを使用して CEV を構築する場合は、Microsoft Visual Studio サブスクリプションを使用してインストールメディアを取得しなければならない場合があります。

8. SQL Server をインストールします。継ぎを実行していることを確認します。

   a. RDS Custom for SQL Server の BYOM の一般的な要件 と RDS Custom for SQL Server CEV のバージョンサポート を確認します。

   b. インスタンスのルートディレクトリをデフォルトの `C:\Program Files\Microsoft SQL Server\` に設定します。このディレクトリは変更しないでください。

   c. [SQL Server データベースエンジンアカウント名] を `NT Service\MSSQLSERVER` または `NT AUTHORITY\NETWORK SERVICE` に設定します。

   d. SQL Server のスタートアップモードを [手動] に設定します。

   e. SQL Server 認証モードを [混在] として選択します。

   f. デフォルトの [データ] ディレクトリと TempDB ロケーションは現在の設定のままにしておきます。

9. SQL Server システム管理者 (SA) サーバーロール権限を `NT AUTHORITY\SYSTEM` に付与します。

```
USE [master]
GO
EXEC master..sp_addsrvrolemember @loginame = N'NT AUTHORITY\SYSTEM' , @rolename =
 N'sysadmin'
GO
```

10. 追加のソフトウェアをインストールするか、要件に合わせて OS とデータベースの設定をカスタマイズします。

11. EC2 インスタンスで Sysprep を実行します。詳細については、「Windows Sysprep を使用して Amazon EC2 AMI を作成する」を参照してください。

12. インストールされている SQL Server のバージョン、他のソフトウェア、およびカスタマイズを含む AMI を保存します。これがゴールデンイメージになります。

13. 作成したイメージの AMI ID を指定して、新しい CEV を作成します。詳細なステップについては、「RDS Custom for SQL Server の CEV の作成」を参照してください。

14. CEV を使用して新しい RDS Custom for SQL Server DB インスタンスを作成します。詳細なステップについては、「CEV から RDS Custom for SQL Server DB インスタンスを作成する」を参照してください。

プリインストールされている SQL Server (LI) を使用した CEV の準備

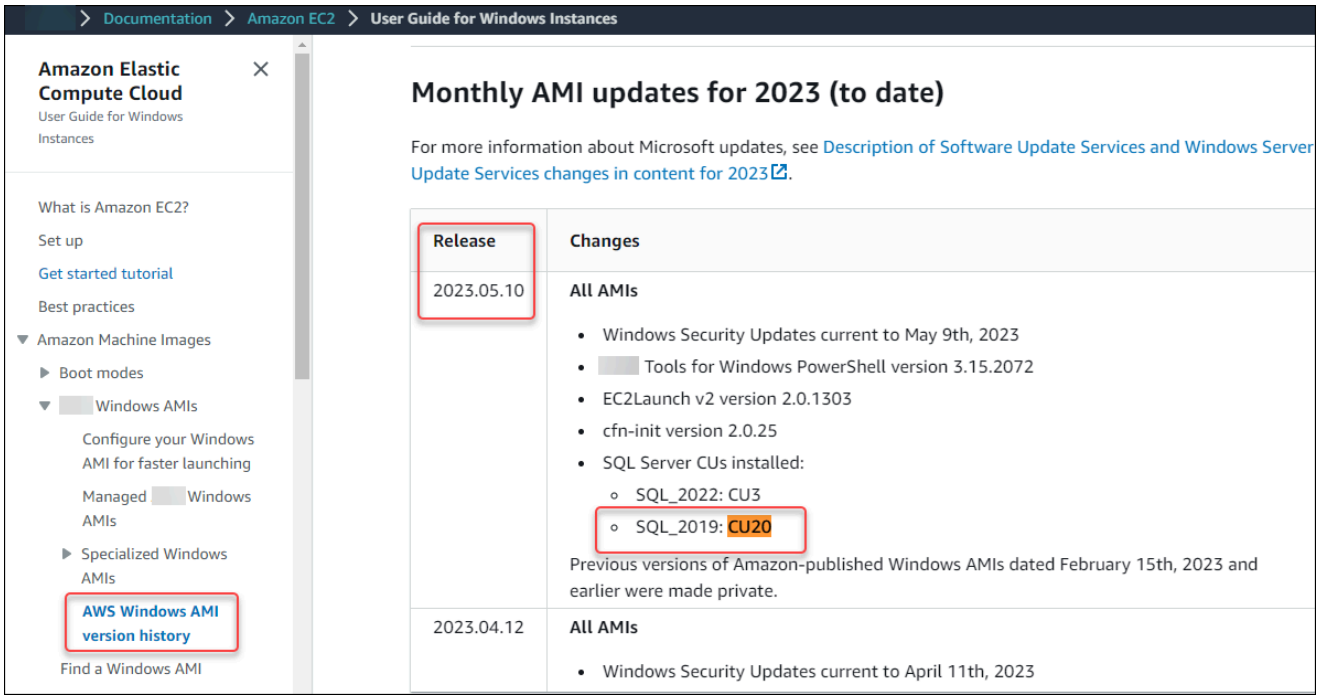
プリインストールされている Microsoft SQL Server (LI) を使用して CEV を作成する次のステップでは、例として SQL Server CU20 リリース番号 `2023.05.10` の AMI を使用します。CEV を作成するときは、最新のリリース番号の AMI を選択してください。これにより、最新の累積更新プログラム (CU) が適用された、サポートされているバージョンの Windows Server と SQL Server を使用していることが保証されます。

プリインストールされている Microsoft SQL Server (LI) を使用して CEV を作成するには

1. ライセンス込み (LI) の Microsoft Windows Server と SQL Server を搭載した最新の AWS EC2 Windows Amazon マシンイメージ (AMI) を選択します。

   a. Windows AMI のバージョン履歴内で CU20 を検索します。

   b. リリース番号をメモしておきます。SQL Server 2019 CU20 のリリース番号は `2023.05.10` です。



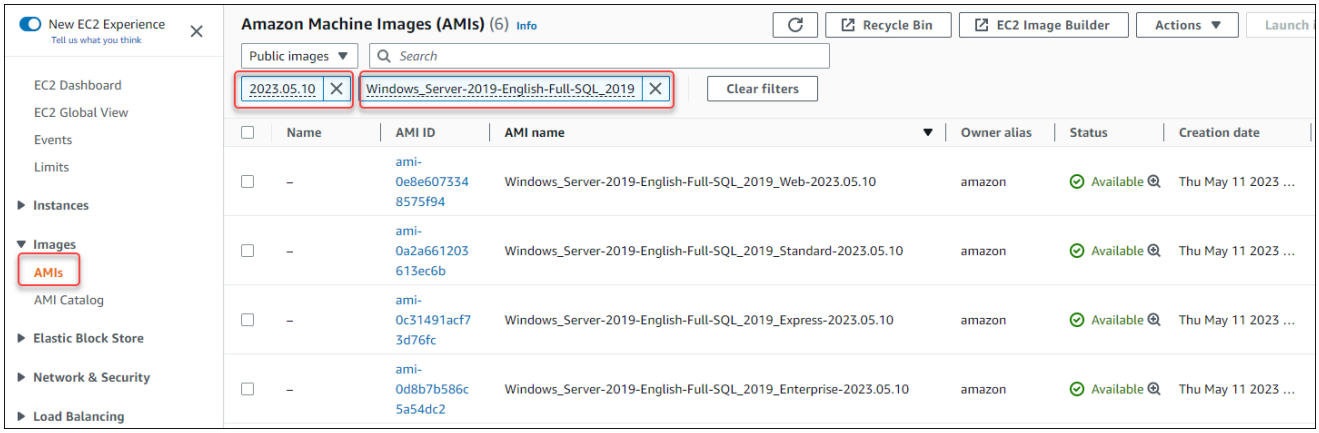
   c. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。

d. Amazon EC2 コンソールのナビゲーションパネルで、[Images] (イメージ)、[AMIs] (AMI) の順に選択します。

e. 「パブリックイメージ」を選択します。

f. 検索ボックスに「`2023.05.10`」と入力します。AMI の一覧が表示されます。

g. 検索ボックスに「`Windows_Server-2019-English-Full-SQL_2019`」と入力し、結果をフィルタリングします。次の結果が表示されます。



h. 使用する SQL Server エディションの AMI を選択します。

2. 選択した AMI から EC2 インスタンスを作成または起動します。
3. EC2 インスタンスにログインして追加のソフトウェアをインストールするか、要件に合わせて OS とデータベースの設定をカスタマイズします。
4. EC2 インスタンスで Sysprep を実行します。Sysprep を使用して AMI を準備する方法の詳細については、「Sysprep を使用して標準化された Amazon マシンイメージ (AMI) を作成する」を参照してください。
5. インストールされている SQL Server のバージョン、他のソフトウェア、およびカスタマイズを含む AMI を保存します。これがゴールデンイメージになります。
6. 作成したイメージの AMI ID を指定して、新しい CEV を作成します。CEV を作成する詳細な手順については、RDS Custom for SQL Server の CEV の作成 を参照してください。
7. CEV を使用して新しい RDS Custom for SQL Server DB インスタンスを作成します。詳細なステップについては、「CEV から RDS Custom for SQL Server DB インスタンスを作成する」を参照してください。

RDS Custom for SQL Server CEV の利用可能なリージョン

RDS Custom for SQL Server のカスタムエンジンバージョン (CEV) サポートは、次の AWS リージョン で利用できます。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (オレゴン)
- 米国西部 (北カリフォルニア)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (ストックホルム)
- 南米 (サンパウロ)

RDS Custom for SQL Server CEV のバージョンサポート

RDS Custom for SQL Server の CEV 作成は、次の AWS EC2 Windows AMI でサポートされています。

- プリインストールされたメディアを使用する CEV の場合、ライセンス込み (LI) Microsoft Windows Server 2019 (OS) および SQL Server 2022 または 2019 搭載の AWS EC2 Windows AMI
- Bring Your Own Media (BYOM) を使用する CEV の場合、Microsoft Windows Server 2019 (OS) 搭載の AWS EC2 Windows AMI

RDS Custom for SQL Server の CEV 作成は、次のオペレーティングシステム (OS) およびデータベースエディションでサポートされています。

- プリインストールされたメディアを使用する CEV の場合：

  - SQL Server 2022 Enterprise、Standard、または Web (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR および CU25)。
  - SQL Server 2019 Enterprise、Standard、または Web (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
- Bring Your Own Media (BYOM) を使用する CEV の場合:
  - SQL Server 2022 Enterprise、Standard、または Developer (CU9、CU13、CU14-GDR、CU15-GDR、CU16、CU17、CU18、CU19、CU19-GDR、CU20-GDR、CU21-GDR、CU22、CU22-GDR、CU23、CU23-GDR、CU24-GDR、および CU25)。
  - SQL Server 2019 Enterprise、Standard、または Developer (CU8、CU17、CU18、CU20、CU24、CU26、CU28-GDR、CU29-GDR、CU30、CU32、および CU32-GDR)。
- プリインストールされたメディアまたは Bring Your Own Media (BYOM) を使用する CEV の場合、サポートされている OS は Windows Server 2019 だけです

詳細については、「AWS Windows AMI バージョン履歴」を参照してください。

RDS Custom for SQL Server CEV の一般的な要件

RDS Custom for SQL Server の CEV を作成するには、次の要件が適用されます。

- CEV の作成に使用される AMI は、RDS Custom for SQL Server でサポートされている OS とデータベースの設定に基づいている必要があります。サポートされている設定の詳細については、「Amazon RDS Custom for SQL Server の要件と制限」を参照してください。
- CEV には一意の名前が必要です。既存の CEV と同じ名前の CEV を作成することはできません。
- CEV は、SQL Server のメジャーバージョン + マイナーバージョン + カスタマイズされた文字列という必須の命名パターンを使用して命名する必要があります。メジャーバージョン + マイナーバージョンは、AMI で提供される SQL Server バージョンと一致する必要があります。例えば、SQL Server 2019 CU17 の AMI に 15.00.4249.2.my_cevtest という名前を付けることができます。
- Sysprep を使用して AMI を準備する必要があります。Sysprep を使用して AMI を準備する方法の詳細については、「Sysprep を使用して標準化された Amazon マシンイメージ (AMI) を作成する」を参照してください。

- AMI のライフサイクルを維持する責任があります。CEV から作成された RDS Custom for SQL Server DB インスタンスには、AMI のコピーは保存されません。CEV の作成に使用した AMI へのポインタが維持されます。RDS Custom for SQL Server DB インスタンスを操作可能な状態に維持するには、AMI が存在する必要があります。

RDS Custom for SQL Server CEV の制限

RDS Custom for SQL Server のカスタムエンジンバージョンでは、次の制約事項が適用されます。

- DB インスタンスや DB スナップショットなどのリソースが関連付けられている場合、CEV を削除することはできません。
- RDS Custom for SQL Server DB インスタンスを作成するには、CEV のステータスが `pending-validation`、`available`、`failed`、または `validating` のいずれかである必要があります。CEV ステータスが `incompatible-image-configuration` の場合、CEV を使用して RDS Custom for SQL Server DB インスタンスを作成することはできません。
- 新しい CEV を使用するように RDS Custom for SQL Server DB インスタンスを変更するには、CEV のステータスが `available` である必要があります。
- 既存の RDS Custom for SQL Server DB インスタンスから AMI または CEV は作成できません。
- 別の AMI を使用するように既存の CEV を変更することはできません。ただし、RDS Custom for SQL Server DB インスタンスを変更して別の CEV を使用するようにすることはできます。詳細については、「RDS Custom for SQL Server DB インスタンスの変更」を参照してください。
- DB インスタンスの作成時に提供される KMS キーとは異なるカスタマー管理の KMS キーを使用した AMI または CEV の暗号化はサポートされていません。
- CEV のクロスリージョンコピーはサポートされていません。
- CEV のクロスアカウントコピーはサポートされていません。
- 削除した CEV を復元または回復することはできません。ただし、同じ AMI から新しい CEV を作成することはできます。
- RDS Custom for SQL Server DB インスタンスは、D:\ ドライブに SQL Server データベースファイルを保存します。CEV に関連付けられた AMI は、Microsoft SQL Server システムデータベースファイルを C:\ ドライブに保存する必要があります。
- RDS Custom for SQL Server DB インスタンスでは、SQL Server に加えられた設定変更が保持されます。CEV から作成された、実行中の RDS Custom for SQL Server DB インスタンスでの OS の設定変更は保持されません。OS の設定を恒久的に変更し、それを新しいベースライン設定として保持する必要がある場合は、新しい CEV を作成し、新しい CEV を使用するように DB インスタンスを変更します。

> **Important**
>
> 新しい CEV を使用するために RDS Custom for SQL Server DB インスタンスを変更するのは、オフライン操作です。変更はすぐに実行することも、毎週のメンテナンスウィンドウに行われるようにスケジュールすることもできます。

- CEV を変更しても、Amazon RDS はそれらの変更を関連する RDS Custom for SQL Server DB インスタンスにプッシュしません。新しいまたは更新された CEV を使用するには、それぞれの RDS Custom for SQL Server DB インスタンスを変更する必要があります。詳細については、「RDS Custom for SQL Server DB インスタンスの変更」を参照してください。
-

> **Important**
>
> CEV が使用する AMI が削除されると、例えば、スケールコンピューティングなど、ホストの交換が必要になる可能性のある変更はすべて失敗します。RDS Custom for SQL Server DB インスタンスが RDS サポートペリメーターの外側に配置されます。CEV に関連付けられている AMI は削除しないことをお勧めします。

RDS Custom for SQL Server の CEV の作成

AWS マネジメントコンソール または AWS CLI を使用してカスタムエンジンバージョン (CEV) を作成できます。その後、CEV を使用して RDS Custom for SQL Server DB インスタンスを作成できます。

Amazon マシンイメージ (AMI) が CEV と同じ AWS アカウントとリージョンに存在することを確認してください。そうではない場合、CEV の作成プロセスが失敗します。

詳細については、「Amazon RDS Custom SQL Server の DB インスタンスの作成と接続」を参照してください。

> **Important**
>
> CEV を作成する手順は、プリインストールされた SQL Server で作成された AMI と、Bring Your Own Media (BYOM) を使用して作成された AMI と同じです。

コンソール

CEV を作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。

   カスタムエンジンバージョンページには、現在存在するすべての CEV が表示されます。CEV をまだ作成していない場合、テーブルは空です。
3. [カスタムエンジンバージョンの作成] を選択します。
4. [Engine type] (エンジンの種類) に [Microsoft SQL Server] を選択します。
5. [エディション] で、使用する DB エンジンエディションを選択します。
6. [Major version] (メジャーバージョン) で、AMI にインストールされているメジャーエンジンバージョンを選択します。
7. 「バージョンの詳細」で、有効な名前を「カスタムエンジンのバージョン名」に入力します。

   名前の形式は*major-engine-version*.*minor-engine-version*.*customized_string*です。ユーザーネームに使用できるのは、1 ~ 50 個の英数字、アンダースコア、ダッシュ、ピリオド(_ - .) のみです。例えば、 名前を**15.00.4249.2.my_cevtest**と入力します。

   必要に応じて、新しい CEV の説明を入力します。
8. [Installation Media] (インストールメディア) には、CEV の作成元となる AMI ID を参照または入力します。
9. [Tags] (タグ) セクションに、CEV を識別するタグを追加します。
10. [カスタムエンジンバージョンの作成] を選択します。

[カスタムエンジンバージョン] ページが表示されます。CEV が [pending-validation] (検証保留中) ステータスで表示されます。

AWS CLI

AWS CLI を使用して CEV を作成するには、create-custom-db-engine-version コマンドを実行します。

以下のオプションは必須です。

- `--engine`

- --engine-version
- --image-id

また、以下のオプションを指定することもできます。

- --description
- --region
- --tags

次の例では、15.00.4249.2.my_cevtest という名前の CEV を作成します。CEV の名前が、メジャーエンジンバージョン番号で始まっていることを確認してください。

Example

Linux、macOS、Unix の場合:

```
aws rds create-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest \
    --image-id ami-0r93cx31t5r596482 \
    --description "Custom SQL Server EE 15.00.4249.2 cev test"
```

次の部分出力は、エンジン、パラメータグループ、およびその他の情報を示しています。

```
"DBEngineVersions": [
    {
    "Engine": "custom-sqlserver-ee",
    "MajorEngineVersion": "15.00",
    "EngineVersion": "15.00.4249.2.my_cevtest",
    "DBEngineDescription": "Microsoft SQL Server Enterprise Edition for RDS Custom for
 SQL Server",
    "DBEngineVersionArn": "arn:aws:rds:us-east-1:<my-account-id>:cev:custom-sqlserver-
ee/15.00.4249.2.my_cevtest/a1234a1-123c-12rd-bre1-1234567890",
    "DBEngineVersionDescription": "Custom SQL Server EE 15.00.4249.2 cev test",

    "Image": [
        "ImageId": "ami-0r93cx31t5r596482",
        "Status": "pending-validation"
     ],
```

```
    "CreateTime": "2022-11-20T19:30:01.831000+00:00",
    "SupportsLogExportsToCloudwatchLogs": false,
    "SupportsReadReplica": false,
    "Status": "pending-validation",
    "SupportsParallelQuery": false,
    "SupportsGlobalDatabases": false,
    "TagList": []
    }
]
```

CEV の作成プロセスが失敗すると、RDS Custom for SQL Server は、メッセージ `RDS-EVENT-0198` とともに `Creation failed for custom engine version` *`major-engine-version.cev_name`* を発行します。メッセージには、イベントが不足ファイルを出力するなど、障害に関する詳細が含まれます。CEV 作成の問題に関するトラブルシューティングのアイデアについては、RDS Custom for SQL Server の CEV エラーのトラブルシューティング を参照してください。

CEV から RDS Custom for SQL Server DB インスタンスを作成する

CEV が正常に作成されると、CEV ステータスに `pending-validation` が表示されます。これで、CEV を使用して新しい RDS Custom for SQL Server DB インスタンスを作成できるようになりました。CEV から新しい RDS Custom for SQL Server DB インスタンスを作成するには、RDS Custom for SQL Server DB インスタンスの作成 を参照してください。

CEV のライフサイクル

CEV ライフサイクルには、以下のステータスが含まれます。

CEV ステータス	説明	トラブルシューティングの推奨事項	
`pending-validation`	CEV が作成され、関連する AMI の検証を保留しています。CEV は、そこから RDS Custom for SQL Server DB インスタンスが作成されるまで `pending-v`	既存のタスクがない場合は、CEV から新しい RDS Custom for SQL Server DB インスタンスを作成します。RDS Custom for SQL Server DB インスタンスを作成する際、システムは CEV の関連する AMI の検証を試みます。	

CEV ステータス	説明	トラブルシューティングの推奨事項	
	alidation に留まります。		
validating	新しい CEV に基づく RDS Custom for SQL Server DB インスタンスを作成するタスクが進行中です。RDS Custom for SQL Server DB インスタンスを作成する際、システムは CEV の関連する AMI の検証を試みます。	既存の RDS Custom for SQL Server DB インスタンスの作成タスクが完了するまで待ちます。RDS EVENTS コンソールを使用して、トラブルシューティングのための詳細なイベントメッセージを確認できます。	
available	CEV は正常に検証されました。CEV は、そこから RDS Custom for SQL Server DB インスタンスが正常に作成されると、available ステータスに入ります。	CEV は追加の検証を必要としません。追加の RDS Custom for SQL Server DB インスタンスを作成する際、または既存のインスタンスを変更するために使用できます。	

CEV ステータス	説明	トラブルシューティングの推奨事項	
`inactive`	CEV は非アクティブ状態に変更されました。	この CEV で RDS Custom DB インスタンスを作成、またはアップグレードすることはできません。また、この CEV を使用して新しい RDS Custom DB インスタンスを作成するために、DB スナップショットを復元することはできません。状態を `ACTIVE` に変更する方法については、RDS Custom for SQL Server の CEV の変更 を参照してください。	
`failed`	この CEV の DB インスタンスの作成ステップは AMI を検証する前に失敗しました。あるいは、CEV が使用している基盤となる AMI が利用可能な状態にありません。	システムが DB インスタンスを作成できなかった根本原因をトラブルシューティングします。詳細なエラーメッセージを表示して、新しい DB インスタンスをもう一度作成してみてください。CEV が使用する基盤となる AMI が使用可能な状態であることを確認します。	

CEV ステータス	説明	トラブルシューティングの推奨事項	
`incompatible-image-configuration`	AMI の検証中にエラーが発生しました。	エラーの技術的な詳細を表示します。この CEV で AMI を再度検証することはできません。次の内容を確認します。推奨事項: • CEV が、SQL Server のメジャーバージョン + マイナーバージョン + カスタマイズされた文字列という必須の命名パターンを使用して命名されていることを確認します。 • CEV 名の SQL Server バージョンが AMI で提供されているバージョンと一致することを確認します。 • OS のビルドバージョンが最低限必要なビルドバージョンを満たしていることを確認します。 • OS のメジャーバージョンが最低限必要なメジャーバージョンを満たしていることを確認します。 正しい情報を使用して新しい CEV を作成します。 必要に応じて、サポートされている AMI を使用して新しい EC2 インスタンスを作成し、そのインスタンスで Sysprep プロセスを実行します。	

RDS Custom for SQL Server の CEV の変更

AWS マネジメントコンソールまたはAWS CLIを使用して CEV を変更できます。CEV の説明またはそのアベイラビリティーステータスを変更できます。CEV には、次のいずれかのステータス値があります。

- `available` - この CEV を使用して、新しい RDS Custom DB インスタンスを作成するか、DB インスタンスをアップグレードできます。これは、新規作成された CEV のデフォルトステータスです。
- `inactive` – この CEV で RDS Custom DB インスタンスを作成、またはアップグレードすることはできません。この CEV を使用して新しい RDS Custom DB インスタンスを作成するために、DB スナップショットを復元することはできません。

CEV ステータスは、`available` から `inactive` または `inactive` から `available` に変更できます。ステータスを `INACTIVE` に変更することで、CEV の誤用を防ぎ、また中止された CEV を再び使用できるようになります。

コンソール

CEV を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、「カスタムエンジンバージョン」 を選択します。
3. 説明またはステータスを変更する CEV を選択します。
4. [アクション] で、[変更] を選択します。
5. 次の変更のいずれか、あるいはすべてを実行します。
   - CEV ステータス設定で、新しいアベイラビリティーステータスを選択します。
   - Version description (バージョンの説明) ページで、新しい説明を入力します。
6. [Modify State] (状態の変更) を選択します。

   CEV が使用中の場合は、コンソールに「CEV ステータスを変更することはできません」と表示されます。問題を修正してから、再試行してください。

[カスタムエンジンバージョン] ページが表示されます。

AWS CLI

AWS CLIを使用して CEV を変更するには、modify-custom-db-engine-versionコマンドを実行します。db-engine-versionsコマンドを実行して、変更するCEVを検索できます。

以下のオプションは必須です。

- `--engine`
- *cev*が変更するカスタムエンジンバージョンの名前である`--engine-version` *cev*です。
- *status*がCEV に割り当てるアベイラビリティーステータスを示す`--status` *status*です。

次の例では、`15.00.4249.2.my_cevtest` という名前の CEV を現在のステータスから `inactive` に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest \
    --status inactive
```

Windows の場合:

```
aws rds modify-custom-db-engine-version ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4249.2.my_cevtest ^
    --status inactive
```

新しい CEV を使用するための RDS Custom for SQL Server DB インスタンスの変更する方法については、「新しい CEV を使用するための RDS Custom for SQL Server DB インスタンスの変更」を参照してください。

新しい CEV を使用するための RDS Custom for SQL Server DB インスタンスの変更

既存の RDS Custom for SQL Server DB インスタンスを変更して別の CEV を使用できます。加えることができる変更には次のものがあります。

- CEV の変更
- DB インスタンスタイプを変更する
- バックアップ保持期間およびバックアップウィンドウを変更する
- メンテナンスウィンドウを変更する

コンソール

RDS Custom for SQL Server DB インスタンスを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] を選択します。
5. 必要に応じて、以下の変更を加えます。
   a. DB エンジンバージョンの場合は、別の CEV を選択してください。
   b. [DB インスタンスクラス] の値を変更します。サポートされているクラスについては、RDS Custom for SQL Server の DB インスタンスクラスでのサポートを参照してください。
   c. バックアップ保持期間の値を変更します。
   d. バックアップウィンドウでは、[Start time] (スタート時間)および [Duration] (期間)の値を設定します。
   e. DB インスタンスのメンテナンスウィンドウでは、[Start day] (スタート日)、[Start time] (スタート時間)、[Duration] (期間)の値を設定します。
6. [Continue] を選択します。
7. すぐに適用または次の定期メンテナンスウィンドウ中に適用を選択します。
8. [DB インスタンスを変更] を選択します。

> **Note**
>
> 例えば、マイナーバージョンをアップグレードする場合など、DB インスタンスをある CEV から別の CEV に変更する場合、SQL Server システムデータベースは、そのデータや設定を含め、現在の RDS Custom for SQL Server DB インスタンスから保持されます。

AWS CLI

AWS CLI を使用して異なる CEV を使用するように DB インスタンスを変更するには、modify-db-instance コマンドを実行します。

以下のオプションは必須です。

- `--db-instance-identifier`
- `--engine-version` *cev*。ここで、*cev* は DB インスタンスを変更するカスタムエンジンバージョンの名前です。

次の例では、`my-cev-db-instance` という名前の DB インスタンスを `15.00.4249.2.my_cevtest_new` という名前の CEV を使用するように変更し、変更を直ちに適用します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-cev-db-instance \
    --engine-version 15.00.4249.2.my_cevtest_new \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-cev-db-instance ^
    --engine-version 15.00.4249.2.my_cevtest_new ^
    --apply-immediately
```

Amazon RDS Custom for SQL Server の CEV 詳細を表示する

CEV の詳細は、AWS マネジメントコンソール または AWS CLI を使用して表示できます。

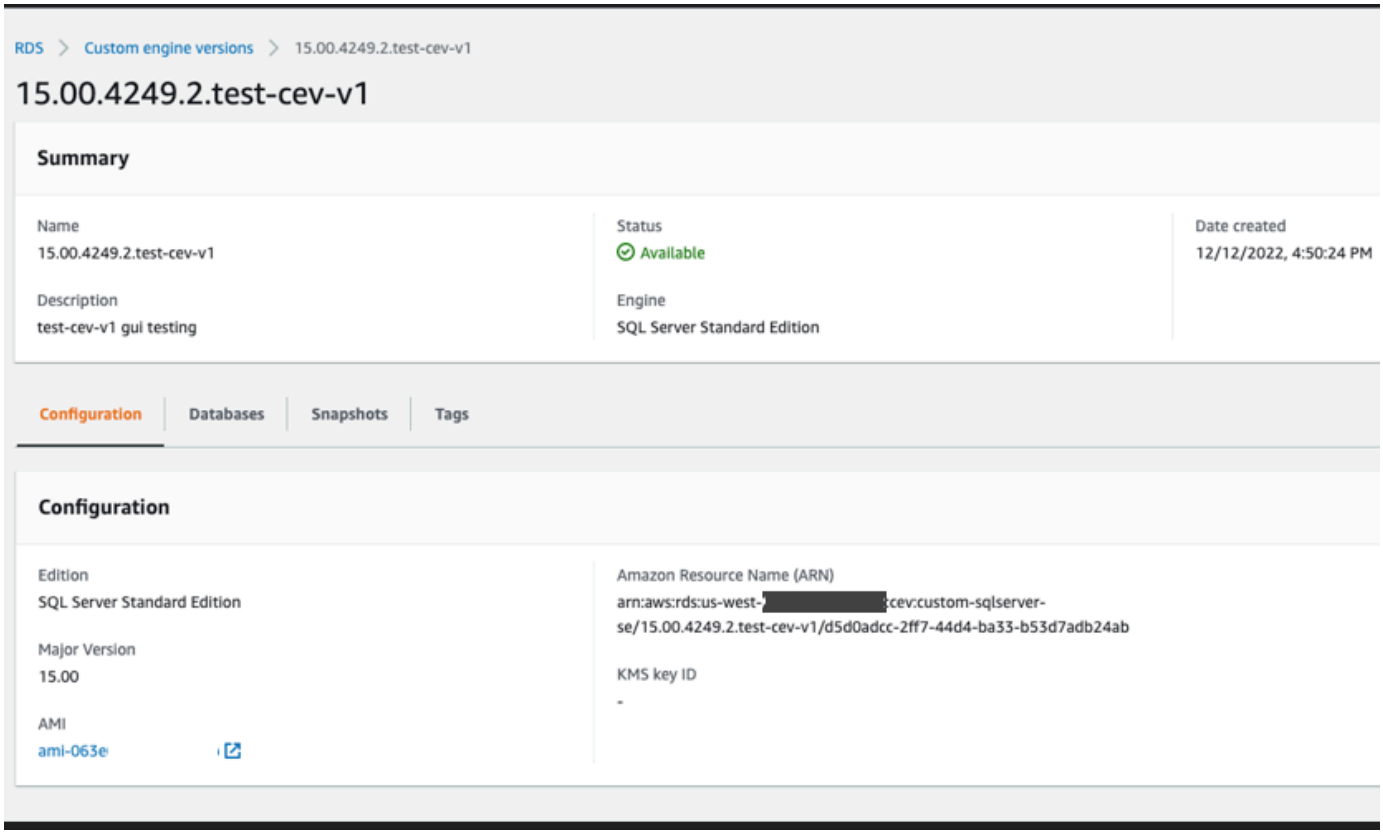
コンソール

CEV の詳細を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。

   カスタムエンジンバージョンページには、現在存在するすべての CEV が表示されます。CEV をまだ作成していない場合、ページは空です。
3. 表示する CEV の名前を選択します。

4. [Configuration] (設定) を選択して、詳細を表示します。



AWS CLI

AWS CLI を使用して CEV に関する詳細を表示するには、describe-db-engine-versions コマンドを実行します。

また、以下のオプションを指定することもできます。

- `--include-all` を使用すると、ライフサイクル状態を問わず、すべての CEV を表示できます。`--include-all` オプションを指定しない場合、`available` ライフサイクル状態の CEV のみが返されます。

```
aws rds describe-db-engine-versions --engine custom-sqlserver-ee --engine-version
 15.00.4249.2.my_cevtest --include-all
{
    "DBEngineVersions": [
        {
            "Engine": "custom-sqlserver-ee",
            "MajorEngineVersion": "15.00",
```

```
            "EngineVersion": "15.00.4249.2.my_cevtest",
            "DBParameterGroupFamily": "custom-sqlserver-ee-15.0",
            "DBEngineDescription": "Microsoft SQL Server Enterprise Edition for custom
 RDS",
            "DBEngineVersionArn": "arn:aws:rds:us-east-1:{my-account-id}:cev:custom-
sqlserver-ee/15.00.4249.2.my_cevtest/a1234a1-123c-12rd-bre1-1234567890",
            "DBEngineVersionDescription": "Custom SQL Server EE 15.00.4249.2 cev test",
            "Image": {
                "ImageId": "ami-0r93cx31t5r596482",
                "Status": "pending-validation"
            },
            "DBEngineMediaType": "AWS Provided",
            "CreateTime": "2022-11-20T19:30:01.831000+00:00",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": false,
            "SupportedFeatureNames": [],
            "Status": "pending-validation",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "TagList": [],
            "SupportsBabelfish": false
        }
    ]
}
```

フィルターを使用して、特定のライフサイクル状態の CEV を表示できます。例えば、ライフサイクル状態が pending-validation、available、failed のいずれかの CEV を表示する場合です。

```
aws rds describe-db-engine-versions engine custom-sqlserver-ee
                region us-west-2 include-all query 'DBEngineVersions[?Status ==
 pending-validation ||
                Status == available || Status == failed]'
```

RDS Custom for SQL Server の CEV の削除

AWS マネジメントコンソール またはAWS CLIを使用して、CEVを削除できます。通常、このタスクには数分かかります。

CEV を削除する前に、CEV が次のいずれかで使用中ではないことを確認してください。

- RDS Custom DB インスタンスを停止します。

- RDS Custom DB インスタンスのスナップショット
- RDS Custom DB インスタンスの自動バックアップ

コンソール

CEV を削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。
3. 説明またはステータスを削除する CEV を選択します。
4. [アクション] で、[削除] を選択します。

   [Delete ***cev_name***?] (cev_name を削除しますか?) ダイアログボックスが表示されます。
5. 「**`delete me`**」と入力し、[削除] を選択します。

   カスタムエンジンバージョンページで、バナーに CEV が削除中であることが示されます。

AWS CLI

AWS CLIを使用して CEV を削除するには、delete-custom-db-engine-versionコマンドを実行します。

以下のオプションは必須です。

- `--engine custom-sqlserver-ee`
- *CEV*が削除するカスタムエンジンバージョンの名前である`--engine-version` *`cev`*です。

次の例では、`15.00.4249.2.my_cevtest` という名前の CEV を削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-custom-db-engine-version \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4249.2.my_cevtest
```

Windows の場合:

```
aws rds delete-custom-db-engine-version ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4249.2.my_cevtest
```

Amazon RDS Custom SQL Server の DB インスタンスの作成と接続

RDS Custom DB インスタンスを作成し、AWS Systems Manager またはリモートデスクトッププロトコル (RDP) を使用してそのインスタンスに接続できます。

> **Important**
>
> DB インスタンスを作成したり、 RDS Custom for SQL Server の DB インスタンスに接続したりする前に、必ず Amazon RDS Custom for SQL Server の環境設定 のタスクを完了してください。
> RDS Custom DBインスタンスの作成時にタグ付けは可能ですが、RDS Custom オートメーションに必要なAWSRDSCustomタグは作成したり変更したりしないでください。詳細については、「RDS Custom for SQL Server リソースのタグ付け」を参照してください。
> RDS カスタム for SQL Server の DB インスタンスを初めて作成するときに、「サービスにリンクされたロールが作成中です」というエラーが表示される場合があります。後ほどもう一度試してください。これを実行した場合は、数分間待ってから DB インスタンスの作成を再試行します。

トピック

- RDS Custom for SQL Server DB インスタンスの作成
- RDS Custom サービスにリンクされたロール
- AWS Systems Managerを使用して RDS カスタム DB インスタンスに接続する
- RDP を使用した RDS Custom DB インスタンスへの接続

RDS Custom for SQL Server DB インスタンスの作成

AWS マネジメントコンソールまたはAWS CLIのいずれかを使用して Amazon RDS Custom SQL Server DB インスタンスを作成します。この手順は、Amazon RDS DB インスタンスの作成と似ています。

詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

コンソール

RDS Custom SQL Server DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. [データベースの作成] を選択します。
4. データベースの作成方法として [スタンダード作成」を選択します。
5. 「エンジンオプション」 で、エンジンタイプとして「Microsoft SQL Server」 を選択します。
6. データベース管理のタイプで、Amazon RDS Customを選択します。
7. 「Edition」セクション で、使用したい DB エンジンのエディションを選択します。
8. (オプション) CEV から DB インスタンスを作成する場合は、[Use custom engine version (CEV)] (カスタムエンジンバージョン (CEV) を使用する) チェックボックスをオンにします。ドロップダウンリストで CEV を選択します。
9. [データベースのバージョン] は、デフォルト値のままにします。
10. 「テンプレート」 では、「作成」 を選択します。
11. 「設定」 で、「DB インスタンス識別子」 の一意の名前を入力します。
12. マスターパスワードを入力するには、以下の操作を行います。

    a. [設定] セクションで、[認証情報の設定] を開きます。

    b. [Auto generate a password (パスワードの自動生成)] チェックボックスをオフにします。

    c. (オプション) マスターユーザーネーム値を変更し、マスターパスワードおよびパスワードを認証するに同じパスワードを入力します。

    デフォルトでは、新規 RDS Custom DB インスタンスはマスターユーザー用に自動生成されたパスワードを使用します。

13. 「DB インスタンスのサイズ」 セクションで、「DB インスタンスクラス」の値を選択します。

    サポートされているクラスについては、RDS Custom for SQL Server の DB インスタンスクラスでのサポートを参照してください。

14. 「ストレージ」 設定を選択します。
15. 「RDS Custom セキュリティ」 で、以下を実行します。

a. [IAM インスタンスプロファイル] では、RDS Custom for SQL Server DB インスタンスとして 2 つのインスタンスプロファイルから選択できます。

   1. [新しいインスタンスプロファイルを作成] を選択して、インスタンスプロファイル名のサフィックスを指定します。詳細については、「AWS マネジメントコンソール を使用したインスタンスプロファイルの自動作成」を参照してください。
   2. 既存のインスタンスプロファイルを選択します。ドロップダウンリストで、AWSRDSCustom で始まるインスタンスプロファイルを選択します。

b. 「暗号化」で、「キーARNを入力」を選択して、使用可能なAWS KMSキーを一覧表示します。次に、リストからキーを選択します。

   AWS KMSRDS Custom にはキーが必要です。詳細については、「対称暗号化 AWS KMS キーであることを確認します。」を参照してください。

16. 残りのセクションで、 RDS Custom DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。次の設定はコンソールに表示されず、サポート対象外です。

    - プロセッサの機能
    - ストレージのオートスケーリング
    - 可用性と耐久性の高い
    - データベース認証のパスワードと Kerberos 認証のオプション (パスワード認証のみサポートされています)
    - 追加設定のデータベースオプショングループ
    - Performance Insights
    - ログのエクスポート
    - マイナーバージョン自動アップグレードの有効化
    - 削除保護

    バックアップ保持期間はサポートされていますが、0 日は選択できません。

17. [データベースの作成] を選択します。

    認証情報の詳細の表示ボタンがデータベースページに表示されます。

    RDS Custom DB インスタンスのマスターユーザー名およびパスワードを表示するには、[認証情報の詳細の表示] を選択します。

マスターユーザーとして DB インスタンスに接続するには、表示されているユーザー名およびパスワードを使用します。

> **Important**
>
> マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。RDS Custom DB インスタンスが利用可能になった後にマスターユーザーのパスワードを変更するには、DB インスタンスを変更します。DB インスタンスの変更の詳細については、「Amazon RDS Custom for SQL Server DB インスタンスの管理」を参照してください。

18. データベースを選択して、RDS Custom DB インスタンスのリストを表示します。
19. 先ほど作成した RDS Custom DB インスタンスを選択します。

    RDS コンソールに、新規の RDS Custom DB インスタンスの詳細が表示されます。

    - RDS Custom DB インスタンスが作成されて使用できるようになるまで、DB インスタンスのステータスは [作成中] となります。ステータスが [利用可能] に変わると、DB インスタンスに接続できます。インスタンスクラスと割り当てられたストレージによっては、新規の DB インスタンスを使用できるようになるまで数分かかることがあります。
    - ロールにはインスタンス (RDS Custom)という値があります。
    - [RDS カスタムオートメーションモード] には [完全なオートメーション] という値があります。この設定は、DB インスタンスが自動モニタリングとインスタンスの回復を提供することを意味します。

AWS CLI

RDS Custom DB インスタンスは、create-db-instance AWS CLI コマンドを使用して作成します。

以下のオプションは必須です。

- `--db-instance-identifier`
- `--db-instance-class` (サポートされている DB インスタンスクラスのリストについては、「」を参照してください)RDS Custom for SQL Server の DB インスタンスクラスでのサポート
- `--engine` (`custom-sqlserver-ee`、`custom-sqlserver-se`、または `custom-sqlserver-web`)
- `--kms-key-id`

- `--custom-iam-instance-profile`

次の例では、`my-custom-instance`という名前の RDS Custom SQL Server DB インスタンスを作成します。バックアップ保持期間は 3 日間です。

> **Note**
>
> カスタムエンジンバージョン (CEV) から DB インスタンスを作成するには、`--engine-version` パラメータに既存の CEV 名を指定します。例: `--engine-version 15.00.4249.2.my_cevtest`

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --engine custom-sqlserver-ee \
    --engine-version 15.00.4073.23.v1 \
    --db-instance-identifier my-custom-instance \
    --db-instance-class db.m5.xlarge \
    --allocated-storage 20 \
    --db-subnet-group mydbsubnetgroup \
    --master-username myuser \
    --master-user-password mypassword \
    --backup-retention-period 3 \
    --no-multi-az \
    --port 8200 \
    --kms-key-id mykmskey \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance
```

Windows の場合:

```
aws rds create-db-instance ^
    --engine custom-sqlserver-ee ^
    --engine-version 15.00.4073.23.v1 ^
    --db-instance-identifier my-custom-instance ^
    --db-instance-class db.m5.xlarge ^
    --allocated-storage 20 ^
    --db-subnet-group mydbsubnetgroup ^
```

```
    --master-username myuser ^
    --master-user-password mypassword ^
    --backup-retention-period 3 ^
    --no-multi-az ^
    --port 8200 ^
    --kms-key-id mykmskey ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

describe-db-instances コマンドを使用して、インスタンスの詳細を入手します。

```
aws rds describe-db-instances --db-instance-identifier my-custom-instance
```

次の部分出力は、エンジン、パラメータグループ、およびその他の情報を示しています。

```
{
    "DBInstances": [
        {
            "PendingModifiedValues": {},
            "Engine": "custom-sqlserver-ee",
            "MultiAZ": false,
            "DBSecurityGroups": [],
            "DBParameterGroups": [
                {
                    "DBParameterGroupName": "default.custom-sqlserver-ee-15",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
            "AutomationMode": "full",
            "DBInstanceIdentifier": "my-custom-instance",
            "TagList": []
        }
    ]
}
```

RDS Custom サービスにリンクされたロール

service-linked role は、AWS アカウント のリソースへのアクセス権を Amazon RDS Custom に付与します。これにより、必要なアクセス許可を手動で追加する必要がなくなるため、RDS Custom の使用が簡単になります。RDS Custom は、サービスにリンクされたロールのアクセス許可を定義し、別途定義されている場合を除き、RDS Custom のみがそのロールを引き受けることができます。定義されるアクセス許可には、信頼ポリシーやアクセス許可ポリシーなどがあり、そのアクセス許可ポリシーを他の IAM エンティティに添付することはできません。

RDS Custom DB インスタンスを作成すると、Amazon RDS と RDS Custom サービスにリンクされたロールの両方が作成され (まだ存在しない場合)、使用されます。詳細については、「Amazon RDS のサービスにリンクされたロールの使用」を参照してください。

RDS カスタム for SQL Server の DB インスタンスを初めて作成するときに、「サービスにリンクされたロールが作成中です」というエラーが表示される場合があります。後ほどもう一度試してください。表示された場合は、数分間待ってから DB インスタンスの作成を再試行します。

AWS Systems Managerを使用して RDS カスタム DB インスタンスに接続する

RDS Custom DB インスタンスを作成した後、AWS Systems Managerセッションマネージャーを使用してインスタンスに接続できます。Session Manager は Systems Manager の機能であり、ブラウザベースのシェルまたはAWS CLI通じて、Amazon EC2 インスタンスの管理に使用できます。詳細については、「AWS Systems Manager Session Manager」を参照してください。

コンソール

Session Manager を使用して DB インスタンスに接続するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、「データベース」 を選択し、停止する RDS Custom DB インスタンスを選択します。
3. [設定] を選択します。
4. リソース ID DB インスタンスの値に注意してください。例えば、リソース ID はdb-ABCDEFGHIJKLMNOPQRS0123456になります。
5. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
6. ナビゲーションペインで、[インスタンス] を選択します。

7. EC2 インスタンスの名前を探し、それに関連付けられているインスタンス ID を選択します。例えば、インスタンス ID はi-abcdefghijklm01234になります。
8. [接続]を選択します。
9. セッションマネージャーを選択します。
10. [接続]を選択します。

    セッションのウィンドウが開きます。

AWS CLI

AWS CLIを使用して RDS Custom DB インスタンスに接続できます。この方法では、AWS CLIのセッションマネージャープラグインが必要です。プラグインをインストールする方法については、「Install the Session Manager plugin for the AWS CLI」を参照してください。

RDS Custom DB インスタンスの DB リソース ID を見つけるには、describe-db-instancesを使用します。

```
aws rds describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

次のサンプル出力は、RDS Custom インスタンスのリソース ID を示しています。プレフィックスはdb-です。

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

DB インスタンスの EC2 インスタンス ID を見つけるには、aws ec2 describe-instancesを使用します。次の例ではリソース ID に db-ABCDEFGHIJKLMNOPQRS0123456 を使用しています。

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

次の出力例は、EC2 インスタンス ID を示しています。

```
i-abcdefghijklm01234
```

`aws ssm start-session`コマンドで、`--target`パラメータに EC2 インスタンス ID を指定します。

```
aws ssm start-session --target "i-abcdefghijklm01234"
```

接続に成功した場合の応答は次のようになります。

```
Starting session with SessionId: yourid-abcdefghijklm1234
[ssm-user@ip-123-45-67-89 bin]$
```

RDP を使用した RDS Custom DB インスタンスへの接続

RDS Custom DB インスタンスを作成すると、RDP クライアントを使用してこのインスタンスに接続できます。この手順は、Amazon EC2 インスタンスへの接続と同じです。詳細については、「Windows インスタンスに接続する」を参照してください。

DB インスタンスに接続するには、インスタンスに関連付けられているキーペアが必要です。RDS Custom は、キーペアを作成します。ペア名はプレフィックス`do-not-delete-rds-custom-`*`DBInstanceIdentifier`*を使用します。AWS Secrets Managerはプライベートキーをシークレットとして保存します。

次のトピックのタスクを完了します。

1. RDP 接続を許可するように DB インスタンスを設定する.
2. シークレットキーを取得する.
3. RDP ユーティリティを使用して EC2 インスタンスに接続します。.

RDP 接続を許可するように DB インスタンスを設定する

RDP 接続を許可するには、VPC セキュリティグループを設定し、ホストでファイアウォールルールを設定します。

VPC セキュリティグループの設定

DB インスタンスに関連付けられた VPC セキュリティグループが、ポート 3389 の送信制御プロトコル (TCP) のインバウンド接続を許可していることを確認してください。VPC セキュリティグループを設定する方法については、「次のように VPC セキュリティグループを設定します。」を参照してください。

ホストでファイアウォールルールを設定する

TCP のポート 3389 でインバウンド接続を許可するには、ホストでファイアウォールルールを設定します。次の例は、その方法を示しています。

次の特定の `-Profile` の値を使用することをお勧めします。`Public`、`Private` または `Domain`。`Any` を使うと、3 つの値すべてを参照します。また、値をカンマで区切って、複数の値を組み合わせて指定することができます。ファイアウォールルールの設定の詳細については、Microsoft のドキュメント「Set-NetFirewallRule」を参照してください。

Systems Manager セッションマネージャーを使用してファイアウォールルールを設定するには

1. AWS Systems Managerを使用して RDS カスタム DB インスタンスに接続するのように、セッションマネージャーに接続します。
2. 以下のコマンドを実行します。

```
Set-NetFirewallRule -DisplayName "Remote Desktop - User Mode (TCP-In)" -Direction
 Inbound -LocalAddress Any -Profile Any
```

Systems Manager CLI コマンドを使用してファイアウォールルールを設定するには

1. 次のコマンドを使用して、ホストで RDP を開きます。

```
OPEN_RDP_COMMAND_ID=$(aws ssm send-command --region $AWS_REGION \
    --instance-ids $RDS_CUSTOM_INSTANCE_EC2_ID \
    --document-name "AWS-RunPowerShellScript" \
    --parameters '{"commands":["Set-NetFirewallRule -DisplayName \"Remote Desktop -
 User Mode (TCP-In)\" -Direction Inbound -LocalAddress Any -Profile Any"]}' \
    --comment "Open RDP port" | jq -r ".Command.CommandId")
```

2. 出力で返されたコマンド ID を使用して、前のコマンドのステータスを取得します。以下のクエリを使用してコマンド ID を返すには、jq プラグインがインストールされていることを確認してください。

```
aws ssm list-commands \
    --region $AWS_REGION \
    --command-id $OPEN_RDP_COMMAND_ID
```

シークレットキーを取得する

AWS マネジメントコンソールまたは、AWS CLIを使用してシークレットキーを取得します。

コンソール

シークレットキーを取得するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、「データベース」 を選択し、接続したい RDS Custom DB インスタンスを選択します。
3. [設定] タブを選択します。
4. DB インスタンスの DB インスタンス ID に注意してください。例えば、*my-custom-instance*。
5. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
6. ナビゲーションペインで、[インスタンス] を選択します。
7. EC2 インスタンスの名前を探し、それに関連付けられているインスタンス ID を選択します。

   この例では、インスタンス ID は`i-abcdefghijklm01234`です。
8. 詳細でキーペア名を探します。ペア名には DB 識別子が含まれます。この例では、ペア名は `do-not-delete-rds-custom-`*`my-custom-instance`*`-0d726c` です。
9. インスタンスの概要で、Public IPv4 DNSを探します。この例では、公開 DNS は`ec2-12-345-678-901.us-east-2.compute.amazonaws.com`です。
10. AWS Secrets Managerコンソールをhttps://console.aws.amazon.com/secretsmanager/で開きます。
11. キーペアと同じ名前のシークレットを選択します。
12. [シークレットの値を取得する] を選択します。

AWS CLI

プライベートキーを取得するには

1. `aws rds describe-db-instances`コマンドを呼び出し、RDS Custom DB インスタンスのリストを取得します。

```
aws rds describe-db-instances \
```

```
    --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
    --output text
```

2. サンプルの出力から DB インスタンス識別子を選択します。例えば、do-not-delete-rds-custom-*my-custom-instance*。
3. aws ec2 describe-instancesコマンドを呼び出し、DB インスタンスの EC2 インスタンス ID を検索します。次の例では、EC2 インスタンス名を使用して DB インスタンスを説明します。

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=do-not-delete-rds-custom-my-custom-instance" \
    --output text \
    --query 'Reservations[*].Instances[*].InstanceId'
```

次の出力例は、EC2 インスタンス ID を示しています。

```
i-abcdefghijklm01234
```

4. 以下の例のように EC2 インスタンス ID を指定して、キー名を検索します。

```
aws ec2 describe-instances \
    --instance-ids i-abcdefghijklm01234 \
    --output text \
    --query 'Reservations[*].Instances[*].KeyName'
```

次の出力例は、プレフィックスdo-not-delete-rds-custom-*DBInstanceIdentifier*を使用するキー名を示しています。

```
do-not-delete-rds-custom-my-custom-instance-0d726c
```

RDP ユーティリティを使用して EC2 インスタンスに接続します。

「Amazon EC2 ユーザーガイド」の「RDP を使用して Windows インスタンスに接続する」の手順に従ってください。この手順では、プライベートキーが含まれる.pem ファイルを作成したことを前提としています。

Amazon RDS Custom for SQL Server DB インスタンスの管理

Amazon RDS Custom for SQL Server は、Amazon RDS DB インスタンスの通常の管理タスクのサブセットをサポートしています。以下では、AWS マネジメントコンソールおよびAWS CLIを使用してサポートされている RDS Custom for SQL Server の管理タスクについて説明します。

トピック

- RDS Custom オートメーションの一時停止と再開
- RDS Custom for SQL Server DB インスタンスの変更
- RDS Custom for SQL Server DB インスタンスのストレージの変更
- RDS Custom for SQL Server リソースのタグ付け
- RDS Custom for SQL Server DB インスタンスの削除
- RDS Custom for SQL Server DB インスタンスの起動と停止

RDS Custom オートメーションの一時停止と再開

RDS Custom は、RDS Custom for SQL Server DB インスタンスのモニタリングとインスタンスのリカバリを自動的に提供します。インスタンスをカスタマイズする必要がある場合は、以下を実行します。

1. RDS Custom オートメーションを指定した期間、一時停止します。一時停止により、カスタマイズが RDS Customオートメーションに干渉しないようにします。
2. 必要に応じて RDS Custom for SQL Server DB インスタンスをカスタマイズします。
3. 次のいずれかを実行します。
   - オートメーションをマニュアルで再開します。
   - 一時停止期間が終了するのを待ちます。この場合、RDS Custom はモニタリングとインスタンスの回復を自動的に再開します。

> **Important**
>
> オートメーションの一時停止と再開は、RDS Custom for SQL Server DBインスタンス変更の際にのみサポートされているオートメーションタスクです。

コンソール

RDS Custom オートメーションを一時停止または再開するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、データベース を選択し、変更する RDS Custom DB インスタンスを選択します。
3. Modify を選択します。Modify DB instance ページが表示されます。
4. 「RDS Customオートメーションモード」では、以下のいずれかのオプションを選択します。
   - 一時停止中は、RDS Custom DB インスタンスのモニタリングとインスタンスのリカバリを一時停止します。目的の一時停止時間 (分単位) を「オートメーションモードの継続期間」に入力します。最小値は 60 分です (デフォルト)。最大値は 1,440 分です。
   - フルオートメーションは、オートメーションを再開します。
5. 「Continue」 選択して、変更の概要をチェックします。

   RDS Custom がすぐに変更を適用することを示すメッセージが表示されます。
6. 変更が正しければ、Modify DB Instance (DB インスタンスを変更) を選択します。または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

   RDS コンソールに、変更の詳細が表示されます。オートメーションを一時停止すると、 RDS Custom DB インスタンスのステータスがオートメーションの一時停止を示します。
7. (オプション) ナビゲーションペインで 「データベース」 を、それから RDS カスタム DB インスタンスを選択します。

   概要ペインで、RDS Custom オートメーションモードはオートメーションのステータスを示します。オートメーションが一時停止されている場合、値は一時停止です。オートメーションは ***num*** 分後に再開されます。

AWS CLI

RDS Custom オートメーションを一時停止または再開するには、`modify-db-instance` AWS CLI コマンドを使用します。必須パラメータ `--db-instance-identifier` を使用して DB インスタンスを指定します。次のパラメータを使用して、オートメーションモードを制御します。

- `--automation-mode`は、DB インスタンスの一時停止状態を指定します。有効な値は`all-paused`で、オートメーションを一時停止し、`full`は再開します。
- `--resume-full-automation-mode-minutes`は一時停止期間を指定します。デフォルト値は 60 分です。

> **Note**
>
> `--no-apply-immediately`または`--apply-immediately`を指定したかどうかにかかわらず、RDS Custom は、変更をできるだけ早く非同期的に適用します。

コマンド応答では、`ResumeFullAutomationModeTime`は UTC タイムスタンプとして再開時刻を示します。オートメーションモードが`all-paused`の場合、`modify-db-instance`を使用してオートメーションモードを再開するか、一時停止期間を延長できます。その他の`modify-db-instance`オプションはサポートされていません。

次の例では、`my-custom-instance` のオートメーションを 90 分間一時停止します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 90
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 90
```

次の例では、一時停止期間をさらに 30 分間延長します。`ResumeFullAutomationModeTime` で示された元の時刻に 30 分が加算されます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode all-paused \
    --resume-full-automation-mode-minutes 30
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode all-paused ^
    --resume-full-automation-mode-minutes 30
```

次の例では、my-custom-instance のフルオートメーションを再開します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --automation-mode full \
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --automation-mode full
```

以下の部分出力例では、保留されている AutomationMode の値は full です。

```
{
    "DBInstance": {
        "PubliclyAccessible": true,
        "MasterUsername": "admin",
        "MonitoringInterval": 0,
        "LicenseModel": "bring-your-own-license",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "0123456789abcdefg"
            }
        ],
```

```
        "InstanceCreateTime": "2020-11-07T19:50:06.193Z",
        "CopyTagsToSnapshot": false,
        "OptionGroupMemberships": [
            {
                "Status": "in-sync",
                "OptionGroupName": "default:custom-oracle-ee-19"
            }
        ],
        "PendingModifiedValues": {
            "AutomationMode": "full"
        },
        "Engine": "custom-oracle-ee",
        "MultiAZ": false,
        "DBSecurityGroups": [],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.custom-oracle-ee-19",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        ...
        "ReadReplicaDBInstanceIdentifiers": [],
        "AllocatedStorage": 250,
        "DBInstanceArn": "arn:aws:rds:us-west-2:012345678912:db:my-custom-instance",
        "BackupRetentionPeriod": 3,
        "DBName": "ORCL",
        "PreferredMaintenanceWindow": "fri:10:56-fri:11:26",
        "Endpoint": {
            "HostedZoneId": "ABCDEFGHIJKLMNO",
            "Port": 8200,
            "Address": "my-custom-instance.abcdefghijk.us-west-2.rds.amazonaws.com"
        },
        "DBInstanceStatus": "automation-paused",
        "IAMDatabaseAuthenticationEnabled": false,
        "AutomationMode": "all-paused",
        "EngineVersion": "19.my_cev1",
        "DeletionProtection": false,
        "AvailabilityZone": "us-west-2a",
        "DomainMemberships": [],
        "StorageType": "gp2",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVW",
        "ResumeFullAutomationModeTime": "2020-11-07T20:56:50.565Z",
        "KmsKeyId": "arn:aws:kms:us-west-2:012345678912:key/
aa111a11-111a-11a1-1a11-1111a11a1a1a",
```

```
        "StorageEncrypted": false,
        "AssociatedRoles": [],
        "DBInstanceClass": "db.m5.xlarge",
        "DbInstancePort": 0,
        "DBInstanceIdentifier": "my-custom-instance",
        "TagList": []
    }
```

RDS Custom for SQL Server DB インスタンスの変更

RDS Custom for SQL Server DB インスタンスの変更は、Amazon RDS での実行と似ていますが、変更できる内容は以下に限られます。

- DB インスタンスタイプを変更する
- バックアップ保持期間およびバックアップウィンドウを変更する
- メンテナンスウィンドウを変更する
- 新しいバージョンが利用可能になったときに DB エンジンのバージョンをアップグレードする
- 割り当てられたストレージ、プロビジョンド IOPS、およびストレージタイプの変更
- マルチ AZ 配置の許可と削除

RDS for SQL Server DB インスタンスの変更には、次の制限が適用されます。

- Custom DB オプショングループおよびパラメータグループはサポートされていません。
- RDS CuCustom DB インスタンスにマニュアルでアタッチするストレージボリュームは、サポートペリメーター外にあります。

  詳細については、「RDS Custom サポート範囲」を参照してください。

コンソール

RDS Custom for SQL Server DB インスタンスを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] を選択します。

5. 必要に応じて、以下の変更を加えます。

   a. [DB engine version] で、新しいバージョンを選択します。

   b. [DB インスタンスクラス] の値を変更します。サポートされているクラスについては、「RDS Custom for SQL Server の DB インスタンスクラスでのサポート」を参照してください。

   c. バックアップ保持期間の値を変更します。

   d. バックアップウィンドウでは、[Start time] (スタート時間)および [Duration] (期間)の値を設定します。

   e. DB インスタンスのメンテナンスウィンドウでは、[Start day] (スタート日)、[Start time] (スタート時間)、[Duration] (期間)の値を設定します。

6. [Continue] を選択します。

7. すぐに適用または次の定期メンテナンスウィンドウ中に適用を選択します。

8. [DB インスタンスを変更] を選択します。

AWS CLI

RDS Custom for SQL Server DB インスタンスを変更するには、modify-db-instance AWS CLIコマンドを使用します。以下のパラメータを必要に応じて調整します。

- `--db-instance-class` - サポートされているクラスについては、「RDS Custom for SQL Server の DB インスタンスクラスでのサポート」を参照してください。
- `--engine-version` - アップグレードするデータベースエンジンのバージョン番号。
- `--backup-retention-period` - 自動バックアップを0～35日間保持する期間。
- `--preferred-backup-window` - 自動バックアップが作成される毎日の時間範囲。
- `--preferred-maintenance-window` - 週 1 回のシステムメンテナンスを実行できる時間帯 (UTC)
- `--apply-immediately` - すぐにストレージの変更を適用するには、`--apply-immediately` を使用します。

  または `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

RDS Custom for SQL Server DB インスタンスのストレージの変更

RDS Custom for SQL Server DB インスタンスのストレージの変更は、Amazon RDS DB インスタンスのストレージの変更と似ていますが、実行できるのは以下の場合に限られます。

- 割り当てられたストレージサイズを増やす。
- ストレージタイプを変更する。汎用またはプロビジョンド IOPS などの利用可能なストレージタイプを使用できます。プロビジョンド IOPS は、gp3、io1、および io2 Block Express ストレージタイプでサポートされています。
- プロビジョンド IOPS をサポートしているボリュームタイプを使用している場合は、プロビジョンド IOPS を変更します。

RDS for SQL Server DB インスタンスのストレージの変更には、次の制限が適用されます。

- RDS Custom for SQL Server に割り当てられる最小ストレージサイズは 20 GiB です。io1、gp2、gp3 の最大ストレージ制限は 16 TiB で、io2 は 64 TiB をサポートしています。
- Amazon RDS と同様に、割り当てられたストレージを減らすことはできません。これは、Amazon Elastic Block Store (Amazon EBS) ボリュームの制限です。詳細については、Amazon RDS DB インスタンスのストレージを使用するを参照してください。
- RDS Custom for SQL Server DB インスタンスでは、ストレージのオートスケーリングはサポートされていません。
- RDS Custom DB インスタンスに手動でアタッチするストレージボリュームは、ストレージのスケーリングの対象にはなりません。RDS が提供するデフォルトのデータボリューム、つまり D ドライブのみがストレージのスケーリングの対象となります。

  詳細については、「RDS Custom サポート範囲」を参照してください。
- ストレージをスケーリングしても、通常、DB インスタンスの停止やパフォーマンスの低下は発生しません。DB インスタンスのストレージサイズを変更すると、DB インスタンスのステータスは [ストレージの最適化] になります。
- ストレージの最適化には数時間かかることがあります。その後 6 時間、またはインスタンスでストレージの最適化が完了するまでのいずれか長い方の時間まで、ストレージにそれ以上の変更を加えることはできません。詳細については、Amazon RDS DB インスタンスのストレージを使用するを参照してください。

ストレージの詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

ストレージの変更に関する一般的な情報については、「Amazon RDS DB インスタンスのストレージを使用する」を参照してください。

> **Important**
>
> Amazon EC2 または Amazon EBS コンソールまたは API を使用して RDS Custom for SQL Server DB インスタンスのストレージを変更しないでください。Amazon RDS コンソールまたは API の外部でストレージを直接変更すると、データベースの状態が `unsupported-configuration` になります。
> Amazon EC2 または Amazon EBS を使用してストレージを直接変更すると、Amazon RDS はデータベースインスタンスの状態を追跡または管理できません。これにより、以下が発生する可能性があります。
>
> - 高可用性フェイルオーバーメカニズムが正しく機能しない
> - データベースレプリケーションの設定が中断する
> - 冗長機能が失敗する
>
> データベースをサポートされている状態に保つために、Amazon RDS コンソールまたは API を介してのみストレージを変更してください。復旧手順については「RDS Custom for SQL Server DB でサポートされていない構成の修正」を参照してください。

コンソール

RDS Custom for SQL Server DB インスタンスのストレージを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。
4. [Modify] を選択します。
5. 必要に応じて、以下の変更を加えます。
   a. [ストレージ割り当て] に新しい値を入力します。現在よりも大きい値で、かつ 20 GiB ~ 16 TiB である必要があります。

b. ストレージタイプの値を変更します。使用可能な汎用またはプロビジョンド IOPS ストレージタイプを選択できます。プロビジョンド IOPS は、gp3、io1、および io2 Block Express ストレージタイプでサポートされています。

c. プロビジョンド IOPS をサポートするストレージタイプを指定する場合は、プロビジョンド IOPS 値を定義できます。

6. [続行] をクリックしてください。

7. すぐに適用または次の定期メンテナンスウィンドウ中に適用を選択します。

8. [DB インスタンスを変更] を選択します。

AWS CLI

RDS Custom for SQL Server DB インスタンスのストレージを変更するには、modify-db-instance AWS CLIコマンドを使用します。以下のパラメータを必要に応じて調整します。

- `--allocated-storage` - DB インスタンスに割り当てるストレージの量 (ギビバイト単位)。現在よりも大きい値で、かつ 20 ~ 16,384 GiB である必要があります。
- `--storage-type` - ストレージタイプ (gp2、gp3、io1、または io2 など)。
- `--iops` - DB インスタンスのプロビジョンド IOPS。これを指定できるのは、gp3、io1、および io2 など、プロビジョンド IOPS をサポートするストレージタイプだけです。
- `--apply-immediately` - すぐにストレージの変更を適用するには、`--apply-immediately` を使用します。

  または `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

次の例では、my-custom-instance のストレージサイズを 200 GiB に、ストレージタイプを io1 に、プロビジョンド IOPS を 3000 に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --storage-type io1 \
    --iops 3000 \
```

```
    --allocated-storage 200 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --storage-type io1 ^
    --iops 3000 ^
    --allocated-storage 200 ^
    --apply-immediately
```

RDS Custom for SQL Server リソースのタグ付け

RDS Custom リソースには Amazon RDS リソースと同様にタグ付けができますが、いくつかの重要な違いがあります。

- `AWSRDSCustom`RDS Custom オートメーションに必要なタグを作成または変更しないでください。そうすると、オートメーションが中断するおそれがあります。
- `Name` タグは、プレフィックス `do-not-delete-rds-custom` で RDS カスタムリソースに追加されます。顧客から渡されたキーの値はすべて上書きされます。
- 作成時に RDS Custom DB インスタンスに追加されたタグは、他のすべての関連する RDS Custom リソースに伝搬されます。
- DB インスタンスの作成後に RDS Custom リソースにタグを追加しても、タグは伝搬されません。

リソースのタグ付けの詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

RDS Custom for SQL Server DB インスタンスの削除

RDS Custom for SQL Server DB インスタンスを削除するには、以下を実行します。

- DB インスタンスの名前を提供します。
- DB インスタンスの最終的な DB スナップショットを取得するオプションを選択または解除します。
- 自動バックアップを保持するオプションを選択または解除します。

RDS Custom for SQL Server DB インスタンスはコンソールまたは CLI を使用して削除できます。DB インスタンスの削除に必要な時間は、バックアップ保持期間 (削除するバックアップの数)、削除するデータの量、最終スナップショットを作成するかどうかによって異なります。

> **Warning**
>
> RDS Custom for SQL Server DB インスタンスを削除すると、EC2 インスタンスと関連する Amazon EBS ボリュームが完全に削除されます。これらのリソースはいつでも終了または削除しないでください。終了または削除すると、削除や最終的なスナップショットの作成が失敗する可能性があります。

> **Note**
>
> DB インスタンスのステータスが `creating`、`failed`、`incompatible-create`、`incompatible-restore`, `incompatible-network` のいずれかである場合、このインスタンスの最終 DB スナップショットは作成できません。詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

> **Important**
>
> 最終スナップショットを撮る場合は、DB インスタンスの削除中は DB インスタンスにデータを書き込まないようにすることをお勧めします。DB インスタンスの削除が開始されると、データの変更が最終的なスナップショットにキャプチャされることが保証されなくなります。

コンソール

RDS Custom DB インスタンスを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[Database] (データベース) を選択し、削除する RDS Custom for SQL Server DB インスタンスを選択します。RDS Custom for SQL Server DB インスタンスは、ロール Instance (RDS Custom for SQL Server) を示します。
3. [アクション] で、[削除] を選択します。

4. 最終スナップショットを撮るには、[Create final snapshot] (最終スナップショットを作成) を選択し、[Final snapshot name] (最終スナップショット名) として名前を指定します。
5. 自動バックアップを保持するには、[自動バックアップの保持] を選択します。
6. ボックスに「**delete me**」と入力します。
7. [削除] をクリックします。

AWS CLI

RDS Custom for SQL Server DB インスタンスを削除するには、delete-db-instance AWS CLI コマンドを使用します。必須パラメータ `--db-instance-identifier` を使用して DB インスタンスを指定します。残りのパラメータは、Amazon RDS DB インスタンスの場合と同じです。

次の例では、`my-custom-instance`という名前の RDS Custom DB インスタンスを削除し、自動バックアップを保持します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier my-custom-instance \
    --no-skip-final-snapshot \
    --final-db-snapshot-identifier my-custom-instance-final-snapshot \
    --no-delete-automated-backups
```

Windows の場合:

```
aws rds delete-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --no-skip-final-snapshot ^
    --final-db-snapshot-identifier my-custom-instance-final-snapshot ^
    --no-delete-automated-backups
```

最終スナップショットを撮るには、`--final-db-snapshot-identifier` オプションを必ず指定する必要があります。

最終スナップショットをスキップするには、コマンドで `--no-skip-final-snapshot` および `--final-db-snapshot-identifier` オプションの代わりに `--skip-final-snapshot` オプションを指定します。

自動バックアップを削除するには、コマンドで `--no-delete-automated-backups` オプションの代わりに `--delete-automated-backups` オプションを指定します。

RDS Custom for SQL Server DB インスタンスの起動と停止

RDS Custom for SQL Server DB インスタンスの起動と停止を行うことができます。RDS for SQL Server DB インスタンスと同じ一般要件と制限が、RDS Custom for SQL Server DB インスタンスの停止と起動にも適用されます。詳細については、「一時的に Amazon RDS DB インスタンスを停止する」を参照してください。

RDS Custom for SQL Server DB インスタンスの起動と停止には、以下の考慮事項が適用されます。

- DB インスタンスが `STOPPED` の間に、RDS Custom for SQL Server DB インスタンスの EC2 インスタンス属性を変更することはサポートされていません。
- RDS Custom for SQL Server DB インスタンスの停止と起動は、単一のアベイラビリティーゾーンに設定されている場合のみ可能です。マルチ AZ 設定の RDS Custom for SQL Server DB インスタンスは停止できません。
- `SYSTEM` スナップショットは、RDS Custom for SQL Server DB インスタンスを停止したときに作成されます。スナップショットは、RDS Custom for SQL Server DB インスタンスをもう一度起動したときに自動的に削除されます。
- RDS Custom for SQL Server DB インスタンスが停止した状態で EC2 インスタンスを削除すると、RDS Custom for SQL Server DB インスタンスを再起動したときに `C:` ドライブが置換されます。
- インスタンスタイプを変更しない限り、`C:\` ドライブ、ホスト名、およびカスタム構成は、RDS Custom for SQL Server DB インスタンスを停止しても保持されます。
- 以下のアクションを実行すると、RDS Custom は DB インスタンスをサポート範囲外に配置することになりますが、DB インスタンス時間に対して引き続き課金されます。
  - Amazon RDS が停止している間に、基盤となる EC2 インスタンスを起動します。解決するには、`start-db-instance` Amazon RDS API を呼び出すか、EC2 を停止して、RDS Custom インスタンスを `STOPPED` に戻します。
  - RDS Custom for SQL Server DB インスタンスが `ACTIVE` のときに、基礎の EC2 インスタンスを停止します。

DB インスタンスの停止と開始の詳細については、「一時的に Amazon RDS DB インスタンスを停止する」および「以前に停止した Amazon RDS DB インスタンスを開始する」を参照してください。

RDS Custom for SQL Server による Microsoft Active Directory の操作

RDS Custom for SQL Server を使用すると、セルフマネージド Active Directory (AD) または AWS Managed Microsoft AD にインスタンスを参加させることができます。これは、オンプレミスデータセンター、Amazon EC2、その他のクラウドサービスプロバイダーなど、AD をホストしている場所に関係なく行われます。

ユーザーとサービスの認証については、中間ドメインやフォレストの信頼を使用することなく、RDS Custom for SQL Server DB インスタンスで NTLM または Kerberos 認証を使用できます。ユーザーがセルフマネージド Active Directory を使用して RDS Custom for SQL Server DB インスタンスで認証を試みると、認証リクエストはセルフマネージド AD または指定した AWS Managed Microsoft AD に転送されます。

以下のセクションでは、RDS Custom for SQL Server でセルフマネージド Active Directory と AWS Managed Active Directory を使用する方法について説明します。

トピック

- 利用可能なリージョンとバージョン
- セルフマネージド AD またはオンプレミス AD を設定する
- Directory Service を使用して Microsoft Active Directory を設定する
- ネットワーク設定ポートルール
- ネットワーク検証
- RDS Custom for SQL Server インスタンスの Windows 認証のセットアップ
- ドメインの DB インスタンスの管理
- ドメインのメンバーシップを理解する
- Active Directory のトラブルシューティング

利用可能なリージョンとバージョン

RDS Custom for SQL Server は、RDS Custom for SQL Server がサポートされているすべてのリージョンで、NTLM または Kerberos を使用したセルフマネージド AD と AWS Managed Microsoft AD の両方をサポートしています。詳細については、「RDS Custom でサポートされているリージョンと DB エンジン」を参照してください。

セルフマネージド AD またはオンプレミス AD を設定する

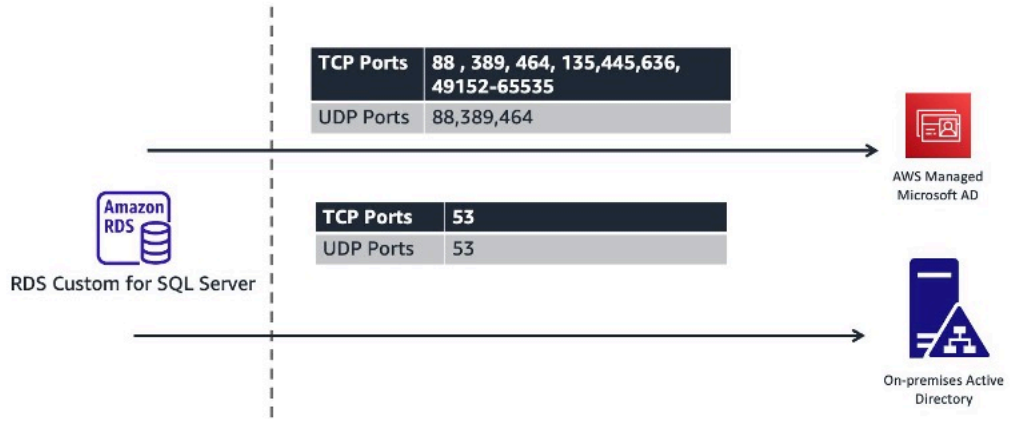
オンプレミスまたはセルフマネージドの Microsoft AD を RDS Custom for SQL Server DB インスタンスに結合するには、アクティブドメインを次のように設定する必要があります。

- RDS Custom for SQL Server DB インスタンスに関連付けられている VPC 内のサブネットを、セルフマネージド AD またはオンプレミス AD で定義します。VPC 内のサブネットと AD サイト内のサブネットの間に競合がないことを確認します。
- AD ドメインコントローラーは、Windows Server 2008 R2 以降のドメイン機能レベルを持ちます。
- AD ドメイン名をシングルラベルドメイン (SLD) 形式にすることはできません。RDS Custom for SQL Server は、SLD ドメインをサポートしていません。
- AD の完全修飾ドメイン名 (FQDN) は 47 文字以内で指定します。

ネットワーク接続の設定

セルフマネージドまたはオンプレミスの AD ネットワーク接続を次のように設定します。

- RDS Custom for SQL Server インスタンスが動作している Amazon VPC と AD との間の接続を設定します。Direct Connect、Site-to-Site VPN、AWS Transit Gateway、および VPC ピアリング接続を使用します。
- RDS Custom for SQL Server セキュリティグループとネットワーク ACL からセルフマネージド AD またはオンプレミス AD へのトラフィックをポートで許可します。詳細については、「ネットワーク設定ポートルール」を参照してください。



DNS 解決を設定する

セルフマネージド AD またはオンプレミス AD で DNS 解決を設定するには、以下の要件をセットアップします。

- セルフホスト Active Directory の完全修飾ドメイン名 (FQDN) を解決するように、VPC 内で DNS 解決を設定します。FQDN の例は `corp.example.local` です。DNS 解決を設定するには、Amazon Route 53 アウトバウンドエンドポイントとリゾルバールールを使用して、特定のドメインのクエリを転送するように VPC DNS リゾルバーを設定します。詳細については、DNS レコードを解決するように Route 53 Resolver のアウトバウンドエンドポイントを設定する方法に関する情報を参照してください。
- VPC とオンプレミスリソースの両方を利用するワークロードの場合は、オンプレミスでホストされている DNS レコードを解決する必要があります。オンプレミスリソースは、AWS でホストされている名前を解決する必要がある場合があります。

  ハイブリッドクラウド設定を作成するには、リゾルバーエンドポイントと条件付き転送ルールを使用して、オンプレミスリソースとカスタム VPC の間の DNS クエリを解決します。詳細については、「Amazon Route 53 デベロッパーガイド」の「VPC とネットワーク間の DNS クエリの解決」を参照してください。

> **Important**
>
> RDS Custom for SQL Server でネットワークインターフェイスの DNS リゾルバー設定を変更すると、DNS 対応の VPC エンドポイントが正しく機能しなくなります。インターネットアクセスのないプライベートサブネット内のインスタンスには、DNS 対応の VPC エンドポイントが必要です。

Directory Service を使用して Microsoft Active Directory を設定する

AWS Managed Microsoft AD は、Windows Server 2019 を搭載した AWS でフルマネージドの Microsoft Active Directory を作成し、2016 フォレストおよびドメインの機能レベルで動作します。Directory Service は Amazon VPC 内の複数の異なるサブネットにドメインコントローラーを作成し、障害が発生した場合でもディレクトリを高可用性にします。

AWS Managed Microsoft AD を使用してディレクトリを作成するには、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft AD の開始」を参照してください。

ネットワーク接続の設定

ディレクトリと DB インスタンスの間のクロス VPC トラフィックを有効にする

同じ VPC 内にディレクトリと DB インスタンスを配置するには、このステップをスキップして「ネットワーク設定ポートルール」の次のステップに進みます。

ディレクトリと DB インスタンスを別の VPC に配置するには、VPC ピアリング接続または AWS Transit Gateway を使用してクロス VPC トラフィックを設定します。VPC ピアリング接続の詳細については、「Amazon VPC ピアリング接続ガイド」の「VPC ピア機能とは」および「Amazon VPC Transit Gateways」の「AWS Transit Gateway とは」を参照してください。

VPC ピアリング接続を使用してクロス VPC トラフィックを有効にするには

1. 適切な VPC ルーティングを設定し、ネットワークトラフィックが双方向にフローするようにします。
2. DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからインバウンドトラフィックを受信できるようにします。詳細については、「ネットワーク設定ポートルール」を参照してください。
3. ネットワークのアクセスコントロールリスト (ACL) はトラフィックをブロックしてはなりません。

別の AWS アカウントがディレクトリを所有している場合は、ディレクトリを共有する必要があります。RDS Custom for SQL Server インスタンスが含まれているディレクトリを AWS アカウント と共有するには、「AWS Directory Service 管理ガイド」の「チュートリアル: シームレスな EC2 ドメイン結合のための AWS Managed Microsoft AD の共有」に従います。

ディレクトリを AWS アカウント間で共有する

1. DB インスタンスのアカウントを使用して Directory Service コンソールにサインインし、ドメインのステータスが SHARED であることを確認してから、続行します。
2. DB インスタンスのアカウントを使用して Directory Service コンソールにサインインした後で、[ディレクトリ ID] の値を書き留めておきます。この ID は、DB インスタンスをドメインに参加させるために使用します。

DNS 解決を設定する

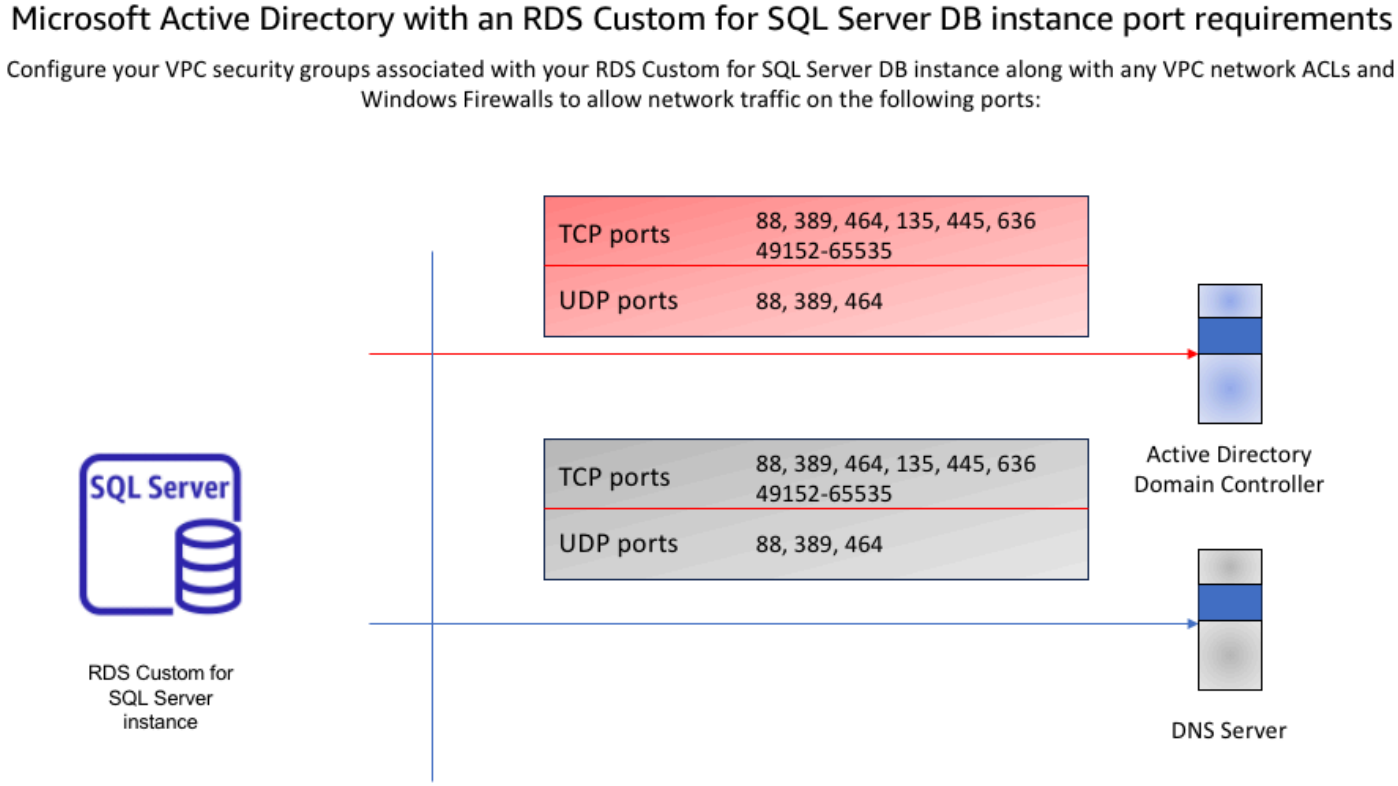
AWS Managed Microsoft AD でディレクトリを作成すると、ユーザーに代わって Directory Service が 2 つのドメインコントローラーを作成し、DNS サービスを追加します。

VPC に既存の AWS Managed Microsoft AD があるか、RDS Custom for SQL Server DB インスタンス以外の AD を起動する予定がある場合は、Route 53 アウトバウンドルールとリゾルバールールを使用して特定のドメインのクエリを転送するように VPC DNS リゾルバーを設定します。詳細については、DNS レコードを解決するように Route 53 Resolver のアウトバウンドエンドポイントを設定する方法に関する情報を参照してください。

ネットワーク設定ポートルール

次のネットワーク設定を満たしていることを確認します。

- RDS Custom for SQL Server DB インスタンスを作成する Amazon VPC と、セルフマネージド Active Directory または AWS Managed Microsoft AD との間に設定した接続。セルフマネージド Active Directory の場合は、AWS Direct Connect、AWS VPN、VPC ピアリング接続、または AWS Transit Gateway を使用して接続を設定します。AWS Managed Microsoft AD の場合は、VPC ピアリング接続を使用して接続を設定します。
- RDS Custom for SQL Server DB インスタンスを作成するサブネットのセキュリティグループと VPC ネットワーク ACL が、次の図に示すポートおよび方向でトラフィックを許可していることを確認します。



以下の表に、各ポートのロールを示します。

プロトコル	ポート	ロール
TCP / UDP	53	ドメインネームシステム (DNS)
TCP / UDP	88	Kerberos 認証
TCP / UDP	464	パスワードを変更 / 設定する
TCP / UDP	389	Lightweight Directory Access プロトコル (LDAP)
TCP	135	Distributed Computing Environment / End Point Mapper (DCE / EPMAP)
TCP	445	Directory Services SMB ファイル共有

プロトコル	ポート	ロール
TCP	636	TLS/SSL (LDAPS) を介した Lightweight Directory Access Protocol (LDAPS)
TCP	49152 - 65535	RPC 用のエフェメラルポート

- 通常、ドメイン DNS サーバーは AD ドメインコントローラーにあります。この機能を使用するために VPC DHCP オプションセットを設定する必要はありません。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。

> **Important**
>
> VPC ネットワーク ACL を使用している場合は、RDS Custom for SQL Server DB インスタンスからのアウトバウンドトラフィックもダイナミックポート (49152-65535) で許可する必要があります。これらのトラフィックルールが、AD ドメインコントローラー、DNS サーバー、RDS Custom for SQL Server DB インスタンスのそれぞれに適用されるファイアウォールにもミラーリングされていることを確認します。
> VPC セキュリティグループでは、ネットワークトラフィックが開始される方向でのみポートを開く必要がありますが、ほとんどの Windows ファイアウォールとおよび VPC ネットワーク ACL では両方向にポートを開く必要があります。

ネットワーク検証

RDS Custom インスタンスをセルフマネージドまたは AWS Managed Microsoft AD のいずれかに参加させる前に、RDS Custom for SQL Server インスタンスの起動を予定している同じ VPC 内の EC2 インスタンスで、以下の点を確認してください。

- 完全修飾ドメイン名 (FQDN) をドメインコントローラー IP に解決できるかどうかを確認します。

```
nslookup corp.example.com
```

コマンドは次のような出力を返す必要があります。

```
Server:  ip-10-0-0-2.us-west-2.compute.internal
```

```
Address:  25.0.0.2

Non-authoritative answer:
Name:    corp.example.com
Addresses:  40.0.9.25 (DC1 IP)
            40.0.50.123 (DC2 IP)
```

- RDS Custom インスタンスを起動する VPC 内の EC2 インスタンスから AWS のサービスを解決します。

```
$region='input-your-aws-region'
$domainFQDN='input-your-domainFQDN'

function Test-DomainPorts {
    param (
        [string]$Domain,
        [array]$Ports
    )

    foreach ($portInfo in $Ports) {
        try {
            $conn = New-Object System.Net.Sockets.TcpClient
            $connectionResult = $conn.BeginConnect($Domain, $portInfo.Port, $null,
 $null)
            $success = $connectionResult.AsyncWaitHandle.WaitOne(1000) # 1 second
 timeout
            if ($success) {
                $conn.EndConnect($connectionResult)
                $result = $true
            } else {
                $result = $false
            }
        }
        catch {
            $result = $false
        }
        finally {
            if ($null -ne $conn) {
                $conn.Close()
            }
        }
        Write-Host "$($portInfo.Description) port open: $result"
    }
```

```
}

# Check if ports can be reached
$ports = @(
    @{Port = 53;   Description = "DNS"},
    @{Port = 88;   Description = "Kerberos"},
    @{Port = 389;  Description = "LDAP"},
    @{Port = 445;  Description = "SMB"},
    @{Port = 5985; Description = "WinRM"},
    @{Port = 636;  Description = "LDAPS"},
    @{Port = 3268; Description = "Global Catalog"},
    @{Port = 3269; Description = "Global Catalog over SSL"},
    @{Port = 9389; Description = "AD DS"}
)

function Test-DomainReachability {
    param (
        [string]$DomainName
    )

    try {
        $dnsResults = Resolve-DnsName -Name $DomainName -ErrorAction Stop
        Write-Host "Domain $DomainName is successfully resolving to following IP
 addresses: $($dnsResults.IpAddress)"
        Write-Host ""
        return $true
    }
    catch {
        Write-Host ""
        Write-Host "Error Message: $($_.Exception.Message)"
        Write-Host "Domain $DomainName reachability check failed, please Configure
 DNS resolution"
        return $false
    }
}

$domain = (Get-WmiObject Win32_ComputerSystem).Domain
if ($domain -eq 'WORKGROUP') {
    Write-Host ""
    Write-Host "Host $env:computername is still part of WORKGROUP and not part of any
 domain"
    }
else {
    Write-Host ""
```

```
    Write-Host "Host $env:computername is joined to $domain domain"
    Write-Host ""
    }


$isReachable = Test-DomainReachability -DomainName $domainFQDN
if ($isReachable) {
    write-Host "Checking if domain $domainFQDN is reachable on required ports  "
    Test-DomainPorts -Domain $domainFQDN -Ports $ports
}
else {
    Write-Host "Port check skipped. Domain not reachable"
}



# Get network adapter configuration
$networkConfig = Get-WmiObject Win32_NetworkAdapterConfiguration |
                 Where-Object { $_.IPEnabled -eq $true } |
                 Select-Object -First 1

# Check DNS server settings
$dnsServers = $networkConfig.DNSServerSearchOrder

if ($dnsServers) {
    Write-Host "`nDNS Server settings:"
    foreach ($server in $dnsServers) {
        Write-Host "  - $server"
    }
} else {
    Write-Host "`nNo DNS servers configured or unable to retrieve DNS server
 information."
}

write-host ""

# Checks reachability to dependent services
$services = "s3", "ec2", "secretsmanager", "logs", "events", "monitoring", "ssm",
 "ec2messages", "ssmmessages"

function Get-TcpConnectionAsync {
    param (
        $ServicePrefix,
        $region
```

```
    )
    $endpoint = "${ServicePrefix}.${region}.amazonaws.com"
    $tcp = New-Object Net.Sockets.TcpClient
    $result = $false

    try {
        $connectTask = $tcp.ConnectAsync($endpoint, 443)
        $timedOut = $connectTask.Wait(3000)
        $result = $tcp.Connected
    }
    catch {
        $result = $false
    }
    return $result
}

foreach ($service in $services) {
    $validationResult = Get-TcpConnectionAsync -ServicePrefix $service -Region
 $region
    Write-Host "Reachability to $service is $validationResult"
}
```

TcpTestSucceeded 値
は、s3、ec2、secretsmanager、logs、events、monitoring、ssm、ec2messages、ssmmessag
に対して True を返す必要があります。

RDS Custom for SQL Server インスタンスの Windows 認証のセットアップ

AD ドメインに参加している RDS Custom for SQL Server DB インスタンスを所有するすべての AWS アカウント アカウントで、専用の OU とその OU を対象としたサービス認証情報を作成することをお勧めします。OU とサービス認証情報を専用にすることで、アクセス許可の競合を回避し、最小特権の原則に従うことができます。

Active Directory レベルのグループポリシーは、AWS のオートメーションやアクセス許可と競合する可能性があります。RDS Custom for SQL Server 用に作成した OU にのみ適用される GPO を選択することをお勧めします。

- セルフマネージド AD またはオンプレミス AD で、OU ドメインユーザーや AD ドメインユーザーを作成するには、ドメインコントローラーをドメイン管理者として接続できます。

- ユーザーとグループを Directory Service ディレクトリに作成するには、管理インスタンスに接続していること、およびユーザーとグループを作成する権限を持つユーザーとしてログインしていることが必要です。詳細については、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft AD でユーザーとグループを管理する」を参照してください。
- Amazon EC2 Windows Server インスタンスから Active Directory を管理するには、EC2 インスタンスに Active Directory ドメインサービスと Active Directory Lightweight Directory サービスツールをインストールする必要があります。詳細については、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft AD の Active Directory 管理ツールをインストールする」を参照してください。
- これらのツールは、管理しやすいように、RDS Custom for SQL Server DB インスタンスではなく、別の管理用の EC2 インスタンスにインストールすることをお勧めします。

AD ドメインサービスアカウントの要件は以下のとおりです。

- コンピュータをドメインに参加させるために、委任されたアクセス許可を持つサービスアカウントが AD ドメインに必要です。ドメインサービスアカウントは、特定のタスクを実行するアクセス許可を委任された AD 内のユーザーアカウントです。
- RDS Custom for SQL Server インスタンスを参加させる組織単位のドメインサービスアカウントに以下のアクセス許可を委任します。
  - DNS ホスト名への書き込みを検証する機能
  - サービスプリンシパル名への書き込みを検証する機能
  - コンピュータオブジェクトを作成および削除する
- セルフマネージド AD およびオンプレミス AD の場合、ドメインサービスアカウントは「AWS 委任されたドメイン名システム管理者」グループのメンバーである必要があります。
- AWS Managed Microsoft AD の場合、ドメインサービスアカウントは「DnsAdmins」グループのメンバーである必要があります。

これらは、コンピュータオブジェクトをセルフマネージド AD および AWS Managed Microsoft AD に参加させるために必要な最小限のアクセス許可のセットです。詳細については、Microsoft Windows Server ドキュメントの「エラー: 制御が委任された管理者以外のユーザーがコンピューターをドメインに参加しようとすると、アクセスが拒否される」を参照してください。

> **Important**
>
> DB インスタンスの作成後に RDS Custom for SQL Server が組織単位 (OU) に作成したコンピュータオブジェクトを移動しないでください。関連オブジェクトを移動すると、RDS Custom for SQL Server DB インスタンスが誤って設定される可能性があります。Amazon RDS で作成したコンピュータオブジェクトを移動する必要がある場合は、ModifyDBInstance アクションを使用して、コンピュータオブジェクトの目的の場所でドメインパラメータを変更します。

トピック

- ステップ 1: AD に組織単位 (OU) を作成する
- ステップ 2: AD に AD ドメインユーザーを作成する
- ステップ 3: セルフマネージドまたは AWS Managed Microsoft AD で AD ユーザーに制御を委任する
- ステップ 4: シークレットを作成する
- ステップ 5: RDS Custom for SQL Server DB インスタンスを作成または変更する
- ステップ 6: Windows 認証の SQL Server ログインを作成する
- ステップ 7: Kerberos または NTLM 認証の使用

ステップ 1: AD に組織単位 (OU) を作成する

AD に組織単位を作成するには、次の手順を実行します。

AD に OU を作成する

1. ドメイン管理者としてドメイン AD に接続します。
2. [Active Directory ユーザーとコンピュータ] を開き、OU を作成するドメインを選択します。
3. ドメインを右クリックし、[新規]、[組織単位] の順に選択します。
4. OU の名前を入力します。

   [誤って削除されないようにコンテナを保護] を有効にします。
5. [OK] を選択してください。新しい OU がドメインの下に表示されます。

AWS Managed Microsoft AD の場合、この OU の名前は、ディレクトリの作成時に入力した NetBIOS 名に基づきます。この OU は AWS が所有し、AWS 関連のディレクトリオブジェクトをすべて含んでいます。ユーザーは、これらのオブジェクトに対して完全な制御権を付与されます。デフォルトでは、この OU の下に 2 つの子 OU (Computers および Users) が存在します。RDS Custom が作成する新しい OU は、NetBIOS に基づく OU の子です。

ステップ 2: AD に AD ドメインユーザーを作成する

ドメインユーザーの認証情報は Secrets Manager のシークレットに使用されます。

AD に AD ドメインユーザーを作成する

1. [Active Directory ユーザーとコンピュータ] を開き、ユーザーを作成するドメインと OU を選択します。
2. [ユーザー] オブジェクトを右クリックして、[新規] を選択し、[ユーザー] を選択します。
3. ユーザーの姓名とログイン名を入力します。[次へ] をクリックします。
4. ユーザーのパスワードを入力します。[ユーザーは次回のログイン時にパスワードを変更する必要があります] または [アカウントは無効になっています] を選択しないでください。[次へ] をクリックします。
5. [OK] をクリックします。新しいユーザーがドメインの下に表示されます。

ステップ 3: セルフマネージドまたは AWS Managed Microsoft AD で AD ユーザーに制御を委任する

ドメイン内の AD ドメインユーザーに制御を委任するには

1. [Active Directory ユーザーとコンピュータ] MMC スナップインを開き、ドメインを選択します。
2. 前に作成した OU を右クリックし、[制御を委任] を選択します。
3. [委任制御ウィザード] で、[次へ] を選択します。
4. [ユーザーまたはグループ] セクションで、[追加] をクリックします。
5. [ユーザー、コンピューター、またはグループを選択] で、作成した AD ユーザーを入力し、[名前を確認] をクリックします。AD ユーザーのチェックが成功したら、[OK] をクリックします。
6. [ユーザーまたはグループ] セクションで、AD ユーザーが追加されたことを確認し、[次へ] をクリックします。
7. [委任するタスク] セクションで、[委任するカスタムタスクを作成] を選択し、[次へ] をクリックします。
8. [Active Directory オブジェクトタイプ] セクションで、次の操作を行います。

[フォルダ内の次のオブジェクトのみ] を選択します。

[コンピュータオブジェクト] を選択します。

[このフォルダに選択したオブジェクトを作成] を選択します。

[このフォルダ内の選択したオブジェクトを削除] を選択し、[次へ] をクリックします。

9. [アクセス許可] セクションで、次の操作を行います。

   [全般] を選択したままにします。

   [DNS ホスト名への検証済み書き込み] を選択します。

   [サービスプリンシパル名への検証済み書き込み] を選択し、[次へ] をクリックします。

10. [制御の委任を完了ウィザード] で、設定を確認して [完了] をクリックします。

ステップ 4: シークレットを作成する

シークレットは、アクティブディレクトリに含める RDS Custom for SQL Server DB インスタンスがある同じ AWS アカウントおよびリージョンに作成します。「ステップ 2: AD に AD ドメインユーザーを作成する」で作成した AD ドメインユーザーの認証情報を保存します。

Console

- AWS Secrets Manager で、[新しいシークレットを保存] を選択します。
- [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
- [キー/値のペア] で、2 つのキーを追加します。
  - 最初のキー `SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME` の値には、AD ユーザーのユーザー名 (ドメインプレフィックスなし) のみを入力します。
  - 2 番目のキーとして `SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD` を入力し、ドメインの AD ユーザーのパスワードを入力します。
- [暗号化キー] に、RDS Custom for SQL Server インスタンスの作成に使用したのと同じ AWS KMS キーを入力します。
- [シークレット名] で、`do-not-delete-rds-custom-` で始まるシークレット名を選択し、このシークレットにアクセスすることをインスタンスプロファイルに許可します。シークレット

に別の名前を選択する場合は、RDSCustomInstanceProfile を更新して [シークレット名] にアクセスします。

- (オプション) [説明] に、シークレット名の説明を入力します。
- タグとして Key="AWSRDSCustom",Value="custom-sqlserver" を追加します。
- [保存]、[次へ] の順にクリックします。
- [ローテーションの設定] は、デフォルト値のままにして、[次へ] を選択します。
- シークレットの設定を確認し、[保存] をクリックします。
- 新しく作成したシークレットを選択し、[シークレット ARN] の値をコピーします。この値は、次のステップで Active Directory を設定するときに使用します。

CLI

シークレットを作成するには、CLI で次のコマンドを実行します。

```
# Linux based
aws secretsmanager create-secret \
--name do-not-delete-rds-custom-DomainUserCredentails \
--description "Active directory user credentials for managing RDS Custom" \
--secret-string "{\"SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME\":\"tester\",
\"SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD\":\"xxxxxxxx\"}" \
--kms-key-id <RDSCustomKMSKey> \
--tags Key="AWSRDSCustom",Value="custom-sqlserver"

# Windows based
aws secretsmanager create-secret ^
--name do-not-delete-rds-custom-DomainUserCredentails ^
--description "Active directory user credentials for managing RDS Custom" ^
--secret-string "{\"SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME\":\"tester\",
\"SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD\":\"xxxxxxxx\"}" ^
--kms-key-id <RDSCustomKMSKey> ^
--tags Key="AWSRDSCustom",Value="custom-sqlserver"
```

ステップ 5: RDS Custom for SQL Server DB インスタンスを作成または変更する

ディレクトリで使用する RDS Custom for SQL Server DB インスタンスを作成または変更します。コンソール、CLI、RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これには以下の 2 つの方法があります。

- コンソール、create-db-instance CLI コマンド、または CreateDBInstance RDS API オペレーションを使用して、新しい SQL Server DB インスタンスを作成します。

  手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance CLI コマンド、または ModifyDBInstance RDS API オペレーションを使用して、既存の SQL Server DB インスタンスを変更します。

  手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、restore-db-instance-from-db-snapshot CLI コマンド、または RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから SQL Server DB インスタンスを復元します。

  手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、または RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、SQL Server DB インスタンスをポイントインタイムに復元します。

  手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

> **Note**
>
> RDS Custom for SQL Server インスタンスを既に手動で AD に参加させている場合は、ネットワーク設定ポートルールとネットワーク検証の設定を確認し、ステップ 1 からステップ 4 までを完了します。`--domain-fqdn`、`--domain-ou`、`--domain-auth-secret-arn` を AD に更新して、ドメイン参加の認証情報と設定を RDS Custom に登録し、モニタリング、CNAME の登録、復旧アクションの実行ができるようにします。

AWS CLI を使用する場合は、DB インスタンスが、作成したディレクトリを使用できるように、以下のパラメータが必要です。

- `--domain-fqdn` パラメータには、セルフマネージド AD の完全修飾ドメイン名を使用します。
- `--domain-ou` パラメータには、セルフマネージド AD で作成した OU を使用します。
- `--domain-auth-secret-arn` パラメータには、作成したシークレット ARN の値を使用します。

> **Important**
>
> DB インスタンスを変更してセルフマネージド AD ドメインや AWS Managed Microsoft AD に参加させたり、これらから削除したりする場合、変更を有効にするには DB インスタンスを再起動する必要があります。変更をすぐに適用するか、次のメンテナンスウィンドウまで待つかを選択できます。[すぐに適用] オプションを選択すると、シングル AZ DB インスタンスのダウンタイムが発生します。マルチ AZ DB クラスターは、再起動を完了する前にフェイルオーバーを実行します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

次の CLI コマンドは、新しい RDS Custom for SQL Server DB インスタンスを作成し、それをセルフマネージドドメインまたは AWS Managed Microsoft AD ドメインに参加させます。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance  \
--engine custom-sqlserver-se \
--engine-version 15.00.4312.2.v1 \
--db-instance-identifier my-custom-instance \
--db-instance-class db.m5.large \
--allocated-storage 100 --storage-type io1 --iops 1000 \
--master-username my-master-username \
--master-user-password my-master-password \
--kms-key-id  my-RDSCustom-key-id \
--custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance  \
--domain-fqdn "corp.example.com" \
--domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" \
--domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-delete-rds-custom-my-AD-test-secret-123456" \
--db-subnet-group-name my-DB-subnet-grp \
--vpc-security-group-ids  my-securitygroup-id \
--no-publicly-accessible \
--backup-retention-period 3 \
--port 8200 \
--region us-west-2 \
--no-multi-az
```

Windows の場合:

```
aws rds create-db-instance  ^
--engine custom-sqlserver-se ^
--engine-version 15.00.4312.2.v1 ^
--db-instance-identifier my-custom-instance ^
--db-instance-class db.m5.large ^
--allocated-storage 100 --storage-type io1 --iops 1000 ^
--master-usernamemy-master-username ^
--master-user-password my-master-password ^
--kms-key-id  my-RDSCustom-key-id ^
--custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance  ^
--domain-fqdn "corp.example.com" ^
--domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" ^
--domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-not-
delete-rds-custom-my-AD-test-secret-123456" ^
--db-subnet-group-name my-DB-subnet-grp ^
--vpc-security-group-ids  my-securitygroup-id ^
--no-publicly-accessible ^
--backup-retention-period 3 ^
--port 8200 ^
--region us-west-2 ^
--no-multi-az
```

Important

AWS Managed Microsoft AD の NetBIOS が corpexample である場合は、それが OU 自体として表示されます。以前に作成した新しい OU は、ネストされた OU として表示されます。AWS Managed Microsoft AD の場合、`--domain-ou` を `"OU=RDSCustomOU,OU=corpexample,DC=corp,DC=example,DC=com"` に設定します。

次のコマンドは、Active Directory ドメインを使用するように既存の RDS Custom for SQL Server DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --domain-fqdn "corp.example.com" \
    --domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-
not-delete-rds-custom-my-AD-test-secret-123456" \
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --domain-fqdn "corp.example.com" ^
    --domain-ou "OU=RDSCustomOU,DC=corp,DC=example,DC=com" ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:do-
not-delete-rds-custom-my-AD-test-secret-123456" ^
```

次の CLI コマンドは、Active Directory ドメインから RDS Custom for SQL Server DB インスタンスを削除します。

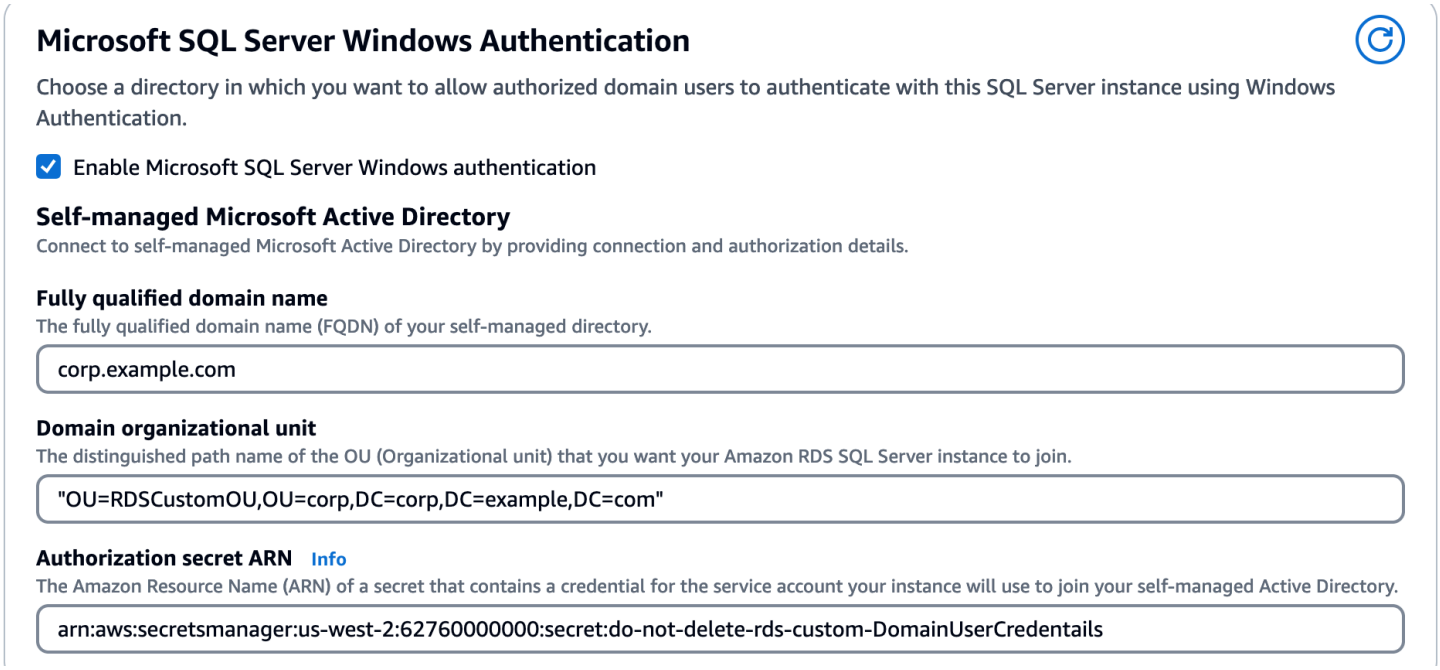
Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-custom-instance \
    --disable-domain
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-custom-instance ^
    --disable-domain
```

コンソールを使用してインスタンスを作成または変更する場合は、[Microsoft SQL Server Windows 認証を有効にする] をクリックして、次のオプションを表示します。



ドメイン FQDN がドメインコントローラーの IP アドレスに解決されていることを確認するのはお客様の責任です。ドメインコントローラー IP が解決されない場合、ドメイン参加オペレーションは失敗しますが、RDS Custom for SQL Server インスタンスの作成は成功します。トラブルシューティング情報については、「Active Directory のトラブルシューティング」を参照してください。

ステップ 6: Windows 認証の SQL Server ログインを作成する

Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同じように SQL Server DB インスタンスに接続します。DB インスタンスは AD ドメインに参加しているため、SQL Server のログインとユーザーをプロビジョニングできます。これは、AD ドメインの AD ユーザーとグループユーティリティから行います。データベースへのアクセス許可は、これらの Windows ログインに付与され無効化されている標準の SQL サーバーのアクセス許可によって管理されています。

AD ユーザーが SQL Server で認証するには、AD ユーザーまたはそのユーザーがメンバーになっている Active Directory グループに SQL Server Windows ログインが必要です。これらの SQL Server ログインでアクセスを許可したり取り消したりして、細分化されたアクセスコントロールを処理します。AD ユーザーが SQL Server ログインを持っていないか、そのようなログインを持つ AD グループに属していない場合、SQL Server DB インスタンスにアクセスすることはできません。

AD SQL Server ログインを作成するには、`ALTER ANY LOGIN` アクセス許可が必要です。このアクセス許可を持つログインをまだ作成していない場合は、SQL Server 認証を使用して DB インスタンスのマスターユーザーとして接続し、マスターユーザーのコンテキストで AD SQL Server ログインを作成します。

AD ユーザーまたは AD グループの SQL Server ログインを作成するには、次に示すようなデータ定義言語 (DDL) コマンドを実行できます。

```
USE [master]
GO
CREATE LOGIN [mydomain\myuser] FROM WINDOWS WITH DEFAULT_DATABASE = [master],
 DEFAULT_LANGUAGE = [us_english];
GO
```

以上により、ドメインのユーザー (人とアプリケーションの両方) は Windows 認証を使用して、ドメインに参加しているクライアントマシンから RDS Custom for SQL Server インスタンスに接続できます。

ステップ 7: Kerberos または NTLM 認証の使用

RDS エンドポイントを使用した NTLM 認証

各 Amazon RDS DB インスタンスにはエンドポイントがあり、各エンドポイントには DB インスタンスの DNS 名とポート番号があります。SQL クライアントアプリケーションを使用して DB インスタンスに接続するには、DB インスタンスの DNS 名とポート番号が必要です。NTLM 認証を使用して認証するには、RDS エンドポイントに接続する必要があります。

計画されたデータベースメンテナンス時や計画外のサービス中断時に、Amazon RDS は自動的に最新のセカンダリデータベースにフェイルオーバーするため、オペレーションは手動の介入なしで速やかに再開できます。プライマリインスタンスおよびセカンダリインスタンスは、同じエンドポイントを使用し、エンドポイントの物理ネットワークアドレスは、フェイルオーバープロセスの一環としてセカンダリに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。

Kerberos 認証

RDS Custom for SQL Server の Kerberos ベースの認証では、特定のサービスプリンシパル名 (SPN) に接続する必要があります。ただし、フェイルオーバーイベント後、アプリケーションは新しい SPN を認識しない場合があります。これに対処するために、RDS Custom for SQL Server には Kerberos ベースのエンドポイントが用意されています。

Kerberos ベースのエンドポイントは、特定の形式に従います。RDS エンドポイントが *rds-instance-name*.*account-region-hash*.*aws-region*.rds.amazonaws.com である場合、対応する Kerberos ベースのエンドポイントは *rds-instance-name*.*account-region-hash*.*aws-region*.awsrds.*fully qualified domain name (FQDN)* になります。

例えば、RDS エンドポイントが ad-test.cocv6zwtircu.us-east-1.rds.amazonaws.com で、ドメイン名が corp-ad.company.com である場合、Kerberos ベースのエンドポイントは ad-test.cocv6zwtircu.us-east-1.awsrds.corp-ad.company.com になります。

この Kerberos ベースのエンドポイントを使用すると、プライマリ SQL Server インスタンスの新しい SPN を指すようにエンドポイントが自動的に更新されるため、フェイルオーバーイベント後でも、Kerberos を使用して SQL Server インスタンスで認証できます。

CNAME の検索

CNAME を検索するには、ドメインコントローラーに接続し、[DNS マネージャー] を開きます。[前方参照ゾーン] と FQDN に移動します。

awsrds、aws-region、および [アカウントとリージョン固有のハッシュ] をナビゲートします。

RDS Custom EC2 インスタンスを接続し、CNAME を使用してデータベースにローカルに接続しようとすると、接続では Kerberos の代わりに NTLM 認証を使用します。

リモートクライアントから CNAME を接続した後に NTLM 接続が返された場合は、必要なポートが許可リストに登録されているかどうかを確認します。

接続で Kerberos を使用しているかどうかを確認するには、次のクエリを実行します。

```
SELECT net_transport, auth_scheme
    FROM sys.dm_exec_connections
    WHERE session_id = @@SPID;
```

ドメインの DB インスタンスの管理

コンソール、AWS CLI、または Amazon RDS API を使用して、DB インスタンスおよびドメインとの関係を管理できます。例えば、DB インスタンスをドメイン内、ドメイン外、またはドメイン間で移動させることができます。

例えば、Amazon RDS API を使用して次を実行できます。

- 失敗したメンバーシップのドメイン参加を再試行するには、ModifyDBInstance API オペレーションを使用して、現在のメンバーシップのディレクトリ ID を指定します。
- メンバーシップの IAM ロール名を更新するには、ModifyDBInstance API オペレーションを使用し、現在のメンバーシップのディレクトリ ID と新しい IAM ロールを指定します。
- ドメインから DB インスタンスを削除するには、ModifyDBInstance API オペレーションを使用し、ドメインパラメータとして none を指定します。

- ドメイン間で DB インスタンスを移動するには、ModifyDBInstance API オペレーションを使用し、新しいドメインのドメイン識別子をドメインパラメータとして指定します。
- 各 DB インスタンスのメンバーシップを一覧表示するには、DescribeDBInstances API オペレーションを使用します。

RDS Custom for SQL Server の DB インスタンスを復元して Active Directory ドメインに追加する

SQL Server DB インスタンスの DB スナップショットを復元するか、ポイントインタイムリカバリ (PITR) を行って、Active Directory ドメインに追加できます。DB インスタンスを復元したら、「ステップ 5: RDS Custom for SQL Server DB インスタンスを作成または変更する」で説明している手順に従って DB インスタンスを変更して AD ドメインに追加します。

ドメインのメンバーシップを理解する

DB インスタンスを作成または変更した後、そのインスタンスは、ドメインのメンバーになります。AWS コンソールは、DB インスタンスのドメインメンバーシップのステータスを示します。DB インスタンスのステータスは、以下のいずれかです。

- joined (参加済み) – インスタンスはドメインのメンバーになっています。
- joining (参加中) – インスタンスは、ドメインのメンバーになる途中です。
- pending-join (参加保留中) – インスタンスのメンバーシップは保留中です。
- メンテナンスまで参加保留中 – 次に予定されているメンテナンスウィンドウ中に、AWS がインスタンスをドメインのメンバーにできるよう試みます。
- pending-removal (削除保留中) – ドメインからのインスタンス削除は保留中です。
- メンテナンスまで削除保留中 – 次に予定されているメンテナンスウィンドウ中に、AWS がドメインからのインスタンスの削除を試みます。
- failed (失敗) – 設定の問題により、インスタンスはドメインに参加できませんでした。インスタンスの変更コマンドを再発行する前に、設定を確認して修正してください。
- removing (削除中) – インスタンスをドメインから削除しています。

ドメインのメンバーになるリクエストは、ネットワーク接続の問題や正しくない IAM ロールが原因で失敗する場合があります。例えば、DB インスタンスを作成した、または既存のインスタンスを変更したが、DB インスタンスをドメインに参加させられない場合があります。この場合、コマンドを再発行して DB インスタンスを作成または変更するか、新しく作成されたインスタンスを変更してドメインに参加させます。

Active Directory のトラブルシューティング

AD をセットアップまたは変更する際に発生する可能性がある問題は以下のとおりです。

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 2 / 0x2	指定されたファイルがシステムで見つかりません。	–domain-ou パラメータで指定された組織単位 (OU) の形式または場所が無効です。AWS Secrets Manager で指定されたドメインサービスアカウントには、OU への参加に必要なアクセス許可がありません。	–domain-ou パラメータを確認してください。ドメインサービスアカウントに OU に対する適切なアクセス許可があることを確認してください。
エラー 5 / 0x5	アクセスが拒否されました。	ドメインサービスアカウントのアクセス許可が正しく設定されていないか、コンピュータアカウントがドメインに既に存在しています。	ドメイン内のドメインサービスアカウントのアクセス許可を確認し、RDS コンピュータアカウントがドメイン内で重複していないことを確認します。RDS コンピュータアカウントの名前は、RDS Custom for SQL Server DB インスタンスで SELECT @@SERVERNAME を実行することで確認できます。マルチ AZ を使用している場合は、フェイルオーバーを使用して再起動し、RDS コンピュータアカウントを再度確認してください。詳細については、「DB インスタン

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
			スの再起動」を参照してください。
エラー 87 / 0x57	パラメータが間違っています。	AWS Secrets Manager で指定されたドメインサービスアカウントには、適切なアクセス許可がありません。ユーザープロファイルが壊れている可能性もあります。	ドメインサービスアカウントの要件を確認します。
エラー 234 / 0xEA	指定された組織単位 (OU) は存在しません。	`–domain-ou` パラメータで指定した OU は、AD 内に存在しません。	`–domain-ou` パラメータを確認して、指定した OU が AD 内に存在することを確認します。
エラー 1326 / 0x52E	ユーザー名またはパスワードが正しくありません。	AWS Secrets Manager で指定されたドメインサービスアカウントの認証情報に、不明なユーザー名または不正なパスワードが含まれています。AD でドメインアカウントが無効になっている場合もあります。	AWS Secrets Manager で指定した認証情報が正しく、ドメインアカウントが Active Directory で有効になっていることを確認します。

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 1355 / 0x54B	指定されたドメインが存在しないか、接続できませんでした。	ドメインがダウンしているか、指定された DNS IP セットにアクセスできないか、指定された FQDN にアクセスできません。	`–domain-dns-ips` および `–domain-fqdn` パラメータが正しいことを確認します。RDS Custom for SQL Server DB インスタンスのネットワーク設定を確認し、AD にアクセスできることを確認します。
エラー 1772 / 0x6BA	RPC サーバーは使用できません。	AD ドメインの RPC サービスへのアクセスに問題がありました。これはサービスまたはネットワークの問題かもしれません。	RPC サービスがドメインコントローラーで実行されていることと、RDS Custom for SQL Server DB インスタンスからドメインの TCP ポート 135 および `49152-65535` にアクセスできることを確認します。
エラー 2224 / 0x8B0	ユーザーアカウントは既に存在しています。	AD に追加しようとしているコンピュータアカウントは既に存在します。	RDS Custom for SQL Server DB インスタンスで `SELECT @@SERVERNAME` を実行してコンピュータアカウントを特定し、それを AD から慎重に削除します。

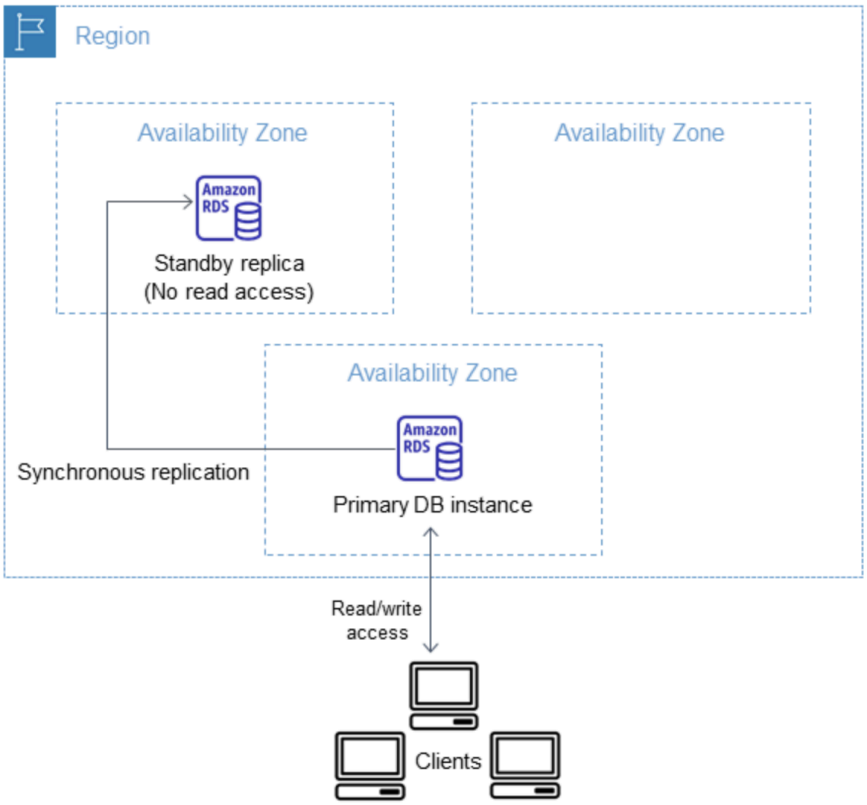
エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 2242 / 0x8c2	このユーザーのパスワードは有効期限が切れています。	AWS Secrets Manager で指定されたドメインサービスアカウントのパスワードは有効期限が切れています。	RDS Custom for SQL Server DB インスタンスを AD に参加させるために使用しているドメインサービスアカウントのパスワードを更新します。

RDS Custom for SQL Server のマルチ AZ 配置の管理

RDS Custom for SQL Server のマルチ AZ DB インスタンスのデプロイでは、Amazon RDS は、異なるアベイラビリティーゾーン (AZ) で同期スタンバイレプリカを自動的にプロビジョンおよび維持します。プライマリ DB インスタンスは、同期的にアベイラビリティーゾーン間でスタンバイレプリカにレプリケートされ、データの冗長性が提供されます。

> **Important**
>
> SRDS Custom for SQL Server のマルチ AZ 配置は、RDS for SQL Server のマルチ AZ 配置とは異なります。RDS for SQL Server のマルチ AZ とは異なり、RDS Custom はアクセス許可が必要な自身のアカウント内で実行されるため、マルチ AZ DB インスタンスを作成する前に RDS Custom for SQL Server の前提条件を設定する必要があります。
> 前提条件を満たさないと、マルチ AZ DB インスタンスは実行に失敗するか、自動的にシングル AZ DB インスタンスに戻る可能性があります。前提条件の詳細については、「RDS Custom for SQL Server を使用したマルチ AZ 配置の前提条件」を参照してください。

高可用性を備えた DB インスタンスを実行すると、計画されたシステムメンテナンス中の可用性が向上する可能性があります。予定されたデータベースメンテナンスまたは予期しないサービス障害時に、Amazon RDS は自動的に最新のセカンダリ DB インスタンスにフェイルオーバーします。この機能により、データベースオペレーションを手動介入なしで速やかに再開できます。プライマリインスタンスおよびスタンバイインスタンスは、同じエンドポイントを使用します。このエンドポイントの物理的なネットワークアドレスは、フェイルオーバープロセスの一環としてセカンダリレプリカに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。



RDS Custom DB インスタンスを作成する際にマルチ AZ を指定すると、RDS Custom for SQL Server マルチ AZ 配置を作成できます。コンソールを使用し、DB インスタンスを変更してマルチ AZ オプションを指定することで、既存の RDS Custom for SQL Server DB インスタンスをマルチ AZ 配置に変換できます。AWS CLI または Amazon RDS API を使用して、マルチ AZ DB インスタンスのデプロイを指定することもできます。

RDS コンソールには、スタンバイレプリカ (セカンダリ AZ) のアベイラビリティーゾーンが表示されます。また、`describe-db-instances` CLI コマンドまたは `DescribeDBInstances` API オペレーションを使用してセカンダリ AZ を見つけることもできます。

マルチ AZ DB 配置を使用する RDS Custom for SQL Server インスタンスは、シングル AZ 配置と比較して書き込みとコミットのレイテンシーが増加する可能性があります。この増加は DB インスタンス間の同期データレプリケーションが原因で起こる可能性があります。AWS はアベイラビリティーゾーン間でのネットワーク接続レイテンシーが低くなるように設計されていますが、配置がスタンバイレプリカにフェイルオーバーした場合はレイテンシーに変化が見られる可能性があります。

> **Note**
>
> 本番ワークロードの場合、高速で一貫したパフォーマンスを実現するために、プロビジョンド IOPS (1 秒あたりの入出力操作) を備えた DB インスタンスクラスを使用することをお

勧めします。DB インスタンスクラスの詳細については、「Amazon RDS Custom for SQL Server の要件と制限」を参照してください。

トピック

- 利用可能なリージョンとバージョン
- RDS Custom for SQL Server を使用したマルチ AZ 配置の制限事項
- RDS Custom for SQL Server を使用したマルチ AZ 配置の前提条件
- RDS Custom for SQL Server マルチ AZ 配置の作成
- RDS Custom for SQL Server シングル AZ 配置からマルチ AZ 配置への変更
- RDS Custom for SQL Server マルチ AZ 配置からシングル AZ 配置への変更
- RDS Custom for SQL Server マルチ AZ 配置のフェイルオーバープロセス

利用可能なリージョンとバージョン

RDS Custom for SQL Server のマルチ AZ 配置は、次の SQL Server エディションでサポートされています。

- SQL Server 2022 および 2019: Enterprise、Standard、Web、Developer Edition

> **Note**
>
> RDS Custom for SQL Server のマルチ AZ 配置は、SQL Server 2019 CU8 (15.00.4073.23) 以前のバージョンではサポートされていません。

RDS Custom for SQL Server のマルチ AZ 配置は、RDS Custom for SQL Server が利用できるすべてのリージョンで利用できます。RDS Custom for SQL Server のマルチ AZ 配置の利用可能なリージョンの詳細については、「RDS Custom for SQL Server でサポートされているリージョンと DB エンジン」を参照してください。

RDS Custom for SQL Server を使用したマルチ AZ 配置の制限事項

RDS Custom for SQL Server のマルチ AZ 配置には、次のような制限事項があります。

- クロスリージョンマルチ AZ 配置はサポートされていません。

- データベースの読み取りアクティビティを受け入れるように、セカンダリ DB インスタンスを設定することはできません。
- マルチ AZ 配置でカスタムエンジンバージョン (CEV) を使用する場合、セカンダリ DB インスタンスも同じ CEV を使用します。セカンダリ DB インスタンスは別の CEV を使用できません。

RDS Custom for SQL Server を使用したマルチ AZ 配置の前提条件

SQL Server シングル AZ 配置の既存の RDS Custom がある場合は、マルチ AZ 配置に変更する前に、次の追加の前提条件が必要です。前提条件は手動で完了するか、提供されている CloudFormation テンプレートを使用して完了するかを選択できます。最新の CloudFormation テンプレートには、シングル AZ 配置とマルチ AZ 配置の両方の前提条件が含まれています。

> **Important**
>
> セットアップを簡単にするために、ネットワークセットアップ手順に記載されている最新の CloudFormation テンプレートファイルを使用して前提条件を作成することをお勧めします。詳細については、「による設定CloudFormation」を参照してください。

> **Note**
>
> 既存の RDS Custom for SQL Server シングル AZ 配置をマルチ AZ 配置に変更すると、これらの前提条件を満たす必要があります。前提条件を満たさないと、マルチ AZ の設定は失敗します。前提条件を満たし、RDS Custom for SQL Server シングル AZ 配置からマルチ AZ 配置への変更 の手順を実行します。

- RDS セキュリティグループのインバウンドルールおよびアウトバウンドルールを更新して、ポート 1120 を許可します。
- プライベートネットワークのアクセスコントロールリスト (ACL) に DB インスタンス VPC の TCP ポート `0-65535` を許可するルールを追加します。
- 新しい Amazon SQS VPC エンドポイントを作成して、RDS Custom for SQL Server DB インスタンスが SQS と通信できるようにします。
- インスタンスプロファイルロールの SQS アクセス許可を更新します。

RDS Custom for SQL Server マルチ AZ 配置の作成

RDS Custom for SQL Server マルチ AZ 配置を作成するには、Amazon RDS Custom SQL Server の DB インスタンスの作成と接続 の手順に従ってください。

> **Important**
>
> セットアップを簡単にするために、ネットワーク設定手順に記載されている最新の CloudFormation テンプレートファイルを使用することをお勧めします。詳細については、「による設定CloudFormation」を参照してください。

マルチ AZ 配置の作成は、完了するまでに数分かかります。

RDS Custom for SQL Server シングル AZ 配置からマルチ AZ 配置への変更

既存の RDS Custom for SQL Server DB インスタンスをシングル AZ 配置からマルチ AZ 配置に変更できます。DB インスタンスを変更すると、Amazon RDS はいくつかのアクションを実行します。

- プライマリ DB インスタンスのスナップショットを取得します。
- スナップショットからスタンバイレプリカ用の新しいボリュームを作成します。これらのボリュームはバックグラウンドで初期化され、データが完全に初期化された後に最大のボリュームパフォーマンスが得られます。
- プライマリおよびセカンダリ DB インスタンス間の同期ブロックレベルレプリケーションをオンにします。

> **Important**
>
> アクティビティのピーク時には、RDS Custom for SQL Server DB インスタンスを本稼働 DB インスタンス上のシングル AZ 配置からマルチ AZ 配置に変更しないことをお勧めします。

AWS では、シングル AZ からマルチ AZ への変換時にダウンタイムを回避するためにスナップショットを使用してスタンバイインスタンスを作成しますが、マルチ AZ への変換時および変換後にパフォーマンスに影響が出ることがあります。この影響は、書き込みレイテンシーに敏感なワークロードにとって重大な可能性があります。この機能により、スナップショットから大量のボリュームをすばやく復元できますが、同期レプリケーションのため、I/O 操作のレイテンシーが著しく増加す

る可能性があります。このレイテンシーはデータベースのパフォーマンスに影響を与える可能性があります。

> **Note**
>
> 2024 年 8 月 29 日より前に RDS Custom for SQL Server DB インスタンスを作成した場合は、変更前に最新のマイナーバージョンにパッチを適用してください。
>
> - SQL Server 2019 インスタンスの場合は、DB エンジンのバージョンを `15.00.4410.1.v1` 以降にアップグレードします。
> - SQL Server 2022 インスタンスの場合は、DB エンジンのバージョンを `16.00.4150.1.v1` 以降にアップグレードします。

トピック

- CloudFormation を使用してシングル AZ 配置からマルチ AZ 配置に変更するための前提条件の設定
- シングル AZ 配置をマルチ AZ 配置に手動で変更するための前提条件を設定する
- RDS コンソール、AWS CLI、または RDS API を使用して変更します。

CloudFormation を使用してシングル AZ 配置からマルチ AZ 配置に変更するための前提条件の設定

マルチ AZ 配置を使用するには、前提条件を含む最新の CloudFormation テンプレートを適用していることを確認するか、最新の前提条件を手動で設定する必要があります。最新の CloudFormation 前提条件テンプレートを既に適用している場合は、これらの手順をスキップできます。

CloudFormation を使用して RDS Custom for SQL Server マルチ AZ 配置の前提条件を設定するには

1. CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. スタックの作成ウィザードを開始するには、シングル AZ 配置の作成に使用した既存のスタックを選択し、[更新] を選択します。

   [スタックの更新] ページが表示されます。
3. [前提条件 – テンプレートの準備] で、[最新のテンプレートを置き換える] を選択します。
4. [Specify template] (テンプレートの指定) ページで、以下を実行します。

a. 最新の CloudFormation テンプレートファイルをダウンロードします。リンク custom-sqlserver-onboard.zip のコンテキスト (右クリック) メニューを開き、[Save Link As] (名前を付けてリンク先を保存) を選択します。

b. `custom-sqlserver-onboard.json` ファイルをコンピュータに保存し、解凍します。

c. [テンプレートソース] で、[テンプレートファイルのアップロード] を選択します。

d. [Choose file] (ファイルを選択) で、`custom-sqlserver-onboard.json` に移動して選択します。

5. [次へ] を選択します。

   [Specify stack details] (DB 詳細の指定) ページが表示されます。

6. デフォルトのオプションを保持するには、[次へ] をクリックします。

   [詳細オプション] ページが表示されます。

7. デフォルトのオプションを保持するには、[次へ] をクリックします。

8. デフォルトのオプションを保持するには、[次へ] をクリックします。

9. [変更の確認] ページで以下の操作を実行します。

   a. [機能] で、[CloudFormation が IAM リソースを作成する可能性があることに同意する] チェックボックスをオンにします。

   b. [Submit] を選択してください。

10. 更新が成功したことを確認します。成功した場合のステータスは `UPDATE_COMPLETE` と表示されます。

更新に失敗すると、更新プロセスで指定された新しい設定はすべてロールバックされます。既存のリソースは引き続き使用できます。例えば、18 と 19 の番号が付いたネットワーク ACL ルールを追加しても、同じ番号の既存のルールが存在する場合、更新によって次のエラーが返されます。`Resource handler returned message: "The network acl entry identified by 18 already exists.` このシナリオでは、既存の ACL ルールを 18 未満の番号を使用するように変更してから、更新を再試行できます。

シングル AZ 配置をマルチ AZ 配置に手動で変更するための前提条件を設定する

> **Important**
>
> セットアップを簡単にするために、ネットワーク設定手順に記載されている最新の CloudFormation テンプレートファイルを使用することをお勧めします。詳細については、「CloudFormation を使用してシングル AZ 配置からマルチ AZ 配置に変更するための前提条件の設定」を参照してください。

前提条件を手動で設定する場合は、以下のタスクを実行します。

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. [エンドポイント] を選択します。[Create Endpoint] (エンドポイントの作成) ページが表示されます。
3. [サービスカテゴリ] で、[AWS のサービス] を選択します。
4. [サービス] で *[SQS]* を検索します
5. [VPC] で、RDS Custom for SQL Server DB インスタンスがデプロイされている VPC を選択します。
6. [サブネット] で、RDS Custom for SQL Server DB インスタンスがデプロイされているサブネットを選択します。
7. [セキュリティグループ] で、*[-vpc-endpoint-sg]* グループを選択します。
8. [ポリシー] で [カスタム] を選択します。
9. カスタムポリシーで、*[AWS #######]*、*[#####]*、*[accountId]*、および *[IAM-Instance-role]* を独自の値に置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/AWSRDSCustom": "custom-sqlserver"
                }
            },
```

```
            "Action": [
                "SQS:SendMessage",
                "SQS:ReceiveMessage",
                "SQS:DeleteMessage",
                "SQS:GetQueueUrl"
            ],
            "Resource": "arn:aws:sqs:us-east-1:111122223333:do-not-delete-
rds-custom-*",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{IAM-Instance-role}"
            }
        }
    ]
}
```

10. Amazon SQS にアクセスするアクセス許可を持つ [インスタンスプロファイル] を更新します。*[AWS #######]*、*[#####]*、*[accountId]* を独自の値に置き換えます。

```
                        {
    "Sid": "SendMessageToSQSQueue",
    "Effect": "Allow",
    "Action": [
      "SQS:SendMessage",
      "SQS:ReceiveMessage",
      "SQS:DeleteMessage",
      "SQS:GetQueueUrl"

    ],
    "Resource": [
      {
        "Fn::Sub": "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:do-
not-delete-rds-custom-*"
      }
    ],
    "Condition": {
      "StringLike": {
        "aws:ResourceTag/AWSRDSCustom": "custom-sqlserver"
      }
    }
  }
                        >
```

11. Amazon RDS セキュリティグループのインバウンドおよびアウトバウンドルールを更新して、ポート 1120 を許可します。

    a. [セキュリティグループ] で、*[-rds-custom-instance-sg]* グループを選択します。
    b. [インバウンドルール] で、ソースの *[-rds-custom-instance-sg]* グループからのポート *1120* を許可するカスタム TCP ルールを作成します。
    c. [アウトバウンドルール] で、カスタム TCP ルールを作成して、送信先の *[-rds-custom-instance-sg]* グループにポート *1120* を許可します。

12. DB インスタンスのソースサブネットの TCP ポート 0-65535 を許可するルールをプライベートネットワークのアクセスコントロールリスト (ACL) に追加します。

> **Note**
>
> インバウンドルールとアウトバウンドルールを作成するときは、既存のルール番号の最大値を書き留めてください。作成する新しいルールのルール番号は 100 未満で、既存のルール番号と一致しない必要があります。

    a. [ネットワーク ACL] で、*[-private-network-acl]* グループを選択します。
    b. [インバウンドルール] で、*privatesubnet1* と *privatesubnet2* のソースを持つ TCP ポート 0-65535 を許可する すべての TCP ルールを作成します。
    c. [アウトバウンドルール] で、TCP ポート 0-65535 を宛先の *privatesubnet1* と *privatesubnet2* に許可するすべての TCP ルールを作成します。

RDS コンソール、AWS CLI、または RDS API を使用して変更します。

前提条件を満たしたら、RDS コンソール、AWS CLI、または RDS API を使用して、RDS Custom for SQL Server DB インスタンスをシングル AZ 配置からマルチ AZ 配置に変更できます。

コンソール

既存の RDS Custom for SQL Server シングル AZ 配置をマルチ AZ 配置に変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

[データベース] ペインが表示されます。

3. 変更する RDS Custom for SQL Server DB インスタンスを選択します。
4. [アクション] で、[マルチ AZ 配置に変換] を選択します。
5. 変更をすぐに適用するには、[確認] ページで [すぐに適用] を選択します。このオプションを選択してもダウンタイムは発生しませんが、パフォーマンスに影響する可能性があります。または、次のメンテナンスウィンドウの間に更新を適用することもできます。詳細については、「スケジュール変更設定の使用」を参照してください。
6. [確認] ページで、[マルチ AZ に変換] を選択します。

AWS CLI

AWS CLI を使用してマルチ AZ DB インスタンス配置に変換するには、modify-db-instance コマンドを呼び出して `--multi-az` オプションを設定します。DB インスタンス識別子と、変更する他のオプションの値を指定します。各オプションの詳細については、「」を参照してください。DB インスタンスの設定

Example

次のコードは、`--multi-az` オプションを含むことで `mycustomdbinstance` を変更します。変更は、`--no-apply-immediately` を使用して次のメンテナンスウィンドウ中に適用されます。今すぐ変更を適用するには、`--apply-immediately` を使用します。詳細については、「スケジュール変更設定の使用」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --multi-az \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --multi-az  \ ^
    --no-apply-immediately
```

RDS API

RDS API を使用してマルチ AZ DB インスタンス配置に変換するには、ModifyDBInstance オペレーションを呼び出し、`MultiAZ` パラメータを true に設定します。

RDS Custom for SQL Server マルチ AZ 配置からシングル AZ 配置への変更

既存の RDS Custom for SQL Server DB インスタンスをマルチ AZ 配置からシングル AZ 配置に変更することができます。

コンソール

RDS Custom for SQL Server DB インスタンスをマルチ AZ 配置からシングル AZ 配置に変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

   [データベース] ペインが表示されます。
3. 変更する RDS Custom for SQL Server DB インスタンスを選択します。
4. [マルチ AZ 配置] で、[いいえ] を選択します。
5. 変更をすぐに適用するには、[確認] ページで [すぐに適用] を選択します。このオプションを選択してもダウンタイムは発生しませんが、パフォーマンスに影響する可能性があります。または、次のメンテナンスウィンドウの間に更新を適用することもできます。詳細については、「スケジュール変更設定の使用」を参照してください。
6. [確認] ページで、[DB インスタンスの変更] を選択します。

AWS CLI

AWS CLI を使用してマルチ AZ 配置をシングル AZ 配置に変更するには、modify-db-instance コマンドを呼び出し、`--no-multi-az` オプションを含めます。DB インスタンス識別子と、変更する他のオプションの値を指定します。各オプションの詳細については、「DB インスタンスの設定」を参照してください。

Example

次のコードは、`--no-multi-az` オプションを含むことで `mycustomdbinstance` を変更します。変更は、`--no-apply-immediately` を使用して次のメンテナンスウィンドウ中に適用されます。

今すぐ変更を適用するには、--apply-immediately を使用します。詳細については、「スケジュール変更設定の使用」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mycustomdbinstance \
    --no-multi-az  \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mycustomdbinstance ^
    --no-multi-az \ ^
    --no-apply-immediately
```

RDS API

RDS API を使用してマルチ AZ 配置をシングル AZ 配置に変更するには、ModifyDBInstance オペレーションを呼び出し、MultiAZ パラメータを false に設定します。

RDS Custom for SQL Server マルチ AZ 配置のフェイルオーバープロセス

DB インスタンスの計画的または計画外の停止がインフラストラクチャの欠陥に起因する場合、マルチ AZ を有効にしていると、Amazon RDS は別のアベイラビリティーゾーンのスタンバイレプリカに自動的に切り替わります。 フェイルオーバーが完了するまでにかかる時間は、プライマリ DB インスタンスが使用できなくなったときのデータベースアクティビティおよびその他の条件によって異なります。フェイルオーバー時間は通常 60 ~ 120 秒です。ただし、大規模なトランザクションや長期にわたる復旧プロセスによって、フェイルオーバー時間が増加する場合があります。フェイルオーバーが完了してから、新しいアベイラビリティーゾーンが RDS コンソールに表示されるまでさらに時間がかかる場合があります。

> **Note**
>
> フェイルオーバーを使用して DB インスタンスを再起動するときに手動でフェイルオーバーを強制的に実行することができます。DB インスタンスの再起動方法については、「DB インスタンスの再起動」を参照してください。

Amazon RDS はフェイルオーバーを自動的に処理するため、管理者が操作しなくても可能な限りすみやかにデータベース操作を再開できます。次の表に記載されている条件のいずれかが発生した場合、プライマリ DB インスタンスがスタンバイレプリカに自動的に切り替わります。これらのフェイルオーバーの理由は、RDS イベントログで確認できます。

フェイルオーバーした理由	説明
The operating system for the RDS Custom for SQL Server Multi-AZ DB instance is being patched in an offline operation	OS パッチまたはセキュリティ更新プログラムのメンテナンス期間中に、フェイルオーバーがトリガーされました。詳細については、「DB インスタンスのメンテナンス」を参照してください。
The primary host of the RDS Custom for SQL Server Multi-AZ DB instance is unhealthy.	マルチ AZ DB インスタンスのデプロイは、障害のあるプライマリ DB インスタンスを検出し、フェイルオーバーしました。
The primary host of the RDS Custom for SQL Server Multi-AZ DB instance is unreachable due to loss of network connectivity.	RDS モニタリングは、プライマリ DB インスタンスへのネットワーク到達可能性による障害を検出し、フェイルオーバーをトリガーしました。
The RDS Custom for SQL Server Multi-AZ DB instance was modified by the customer.	DB インスタンスの変更により、フェイルオーバーがトリガーされました。詳細については、「RDS Custom for SQL Server DB インスタンスの変更」を参照してください。
The storage volume of the primary host of the RDS Custom	マルチ AZ DB インスタンスのデプロイは、プライマリ DB インスタンスでストレージの問題を検出し、フェイルオーバーしました。

フェイルオーバーした理由	説明
for SQL Server Multi-AZ DB instance experienced a failure.	
The user requested a failover of the RDS Custom for SQL Server Multi-AZ DB instance.	RDS Custom for SQL Server Multi-AZ DB インスタンスがフェイルオーバーにより再起動されました。詳細については、「DB インスタンスの再起動」を参照してください。
The RDS Custom for SQL Server Multi-AZ primary DB instance is busy or unresponsive.	プライマリ DB インスタンスが応答しません。次の手順を実行することをお勧めします。 • イベントログと CloudWatch Logs を調べて、CPU、メモリ、またはスワップ領域の使用量が多すぎるかどうかを調べます。詳細については、「Amazon RDS イベント通知の操作」を参照してください。 • Amazon RDS イベントでトリガーするルールを作成します。詳細については、「Amazon RDS イベントでトリガーするルールの作成」を参照してください。 • ワークロードを評価して、適切な DB インスタンスクラスを使用しているかどうかを判断します。詳細については、「DB インスタンスクラス」を参照してください。

マルチ AZ DB インスタンスがフェイルオーバーされたかどうかを判断するには、次を実行します。

- フェイルオーバーがスタートされたことを電子メールまたはSMSで通知するようにDBイベントサブスクリプションを設定します。 イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。
- RDS コンソールまたは API オペレーションを使用して DB イベントを表示します。
- RDS コンソール、CLI、または API オペレーションを使用して、RDS Custom for SQL Server Multi-AZ DB インスタンスのデプロイの現在の状態を表示します。

RDS Custom for SQL Server Multi-AZ マルチ AZ 配置を使用するアプリケーションでの Time to live (TTL) 設定

フェイルオーバーメカニズムでは、スタンバイ DB インスタンスをポイントするように DB インスタンスのドメインネームシステム (DNS) レコードが自動的に変更されます。したがって、DB インスタンスへの既存の接続の再確立が必要になります。DNS キャッシュ Time to live (TTL) 設定値が低いことを確認し、アプリケーションが DNS を長期間キャッシュしないことを確認します。TTL 値が高いと、フェイルオーバー後にアプリケーションが DB インスタンスにすぐに再接続できなくなる可能性があります。

Amazon RDS Custom for SQL Server DB インスタンスのバックアップと復元

Amazon RDS 同様、RDS Custom は、バックアップ保持が有効になっている場合に RDS Custom for SQL Server DB インスタンスの自動バックアップを作成して保存します。また、手動で DB インスタンスをバックアップすることもできます。自動バックアップは、スナップショットバックアップとトランザクションログバックアップで構成されます。スナップショットバックアップは、指定したバックアップウィンドウ中に DB インスタンスのストレージボリューム全体に対して実行されます。PITR の対象となるデータベースのトランザクションログのバックアップは、一定の間隔で実行されます。RDS Custom は、指定したバックアップ保持期間に従って DB インスタンスの自動バックアップを保存します。自動バックアップを使用して、バックアップの保持期間内の特定時点に DB インスタンスを復元できます。

スナップショットのバックアップを手動で作成することもできます。これらのスナップショットバックアップからいつでも新しい DB インスタンスを作成できます。DB スナップショットの手動作成について詳しくは、「RDS Custom for Oracle スナップショットの作成」を参照してください。

スナップショットバックアップはフルバックアップとして機能しますが、課金はストレージの増分使用に対してのみ発生します。 RDS Custom DB インスタンスの初期のスナップショットには、フル DB インスタンスのデータが含まれています。同じ DB インスタンスの後続のスナップショットは増分です。つまり、直近のスナップショットの保存後に変更されたデータのみが含まれます。

トピック

- RDS Custom for Oracle スナップショットの作成
- RDS Custom for SQL Server DB スナップショットからの復元
- RDS Custom for SQL Server インスタンスをポイントインタイムに復元する
- RDS Custom for SQL Server スナップショットの削除
- RDS Custom for SQL Server 自動バックアップの削除

RDS Custom for Oracle スナップショットの作成

RDS Custom for SQL Server は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、DB インスタンス全体をバックアップします。スナップショットを作成するときに、バックアップする RDS Custom for SQL Server DB インスタンスを指定します。次に、スナップショットに名前を付けて、後でそこから復元できるようにします。

スナップショットを作成すると、RDS Custom for SQL Server は DB インスタンスにアタッチされたデータベースボリュームである、ボリューム (`D:`) の Amazon EBS スナップショットを作成します。特定の DB インスタンスと簡単に関連付けられるように、スナップショットには `DBSnapshotIdentifier`、`DbiResourceId`、`VolumeType` のタグが付けられています。

DB スナップショットを作成すると、短時間の I/O が発生します。この一時停止は、DB インスタンスのサイズとクラスに応じて、数秒から数分続く場合があります。スナップショットの作成時間は、データベースの合計数とサイズによって異なります。ポイントインタイム復元 (PITR) オペレーションの対象となるデータベースの数の詳細については、「インスタンスクラスタイプごとに PITR の対象となるデータベースの数」を参照してください。

スナップショットにはストレージボリューム全体が含まれており、テンポラリファイルなどのファイルサイズも、スナップショットの作成時間に影響します。スナップショットの作成の詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

コンソールまたはAWS CLIを使用して RDS Custom for SQL Server スナップショットを作成します。

コンソール

RDS Custom スナップショットを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [データベース] を選択します。
3. RDS Custom DB インスタンスのリストで、スナップショットを取得する DB インスタンスを選択します。
4. アクション で、スナップショットの取得 を選択します。

   [Take DB snapshot] (DB スナップショットの取得) ウィンドウが表示されます。
5. [スナップショット名] に、スナップショットの名前を入力します。
6. スナップショットの取得 を選択します。

AWS CLI

create-db-snapshot AWS CLIコマンドを使用して、RDS Custom DB インスタンスのスナップショットを作成します。

以下のオプションを指定します。

- `--db-instance-identifier` - バックアップする RDS Custom DB インスタンスを指定します。
- `--db-snapshot-identifier` - RDS Custom スナップショットに名前を付けて、後でそこから復元できるようにします。

この例では、*my-custom-instance* という RDS Custom DB インスタンスの、*my-custom-snapshot* という DB スナップショットを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-snapshot \
    --db-instance-identifier my-custom-instance \
    --db-snapshot-identifier my-custom-snapshot
```

Windows の場合:

```
aws rds create-db-snapshot ^
    --db-instance-identifier my-custom-instance ^
    --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for SQL Server DB スナップショットからの復元

RDS Custom for SQL Server DB インスタンスを復元するときは、DB スナップショットの名前と新しいインスタンスの名前を指定します。スナップショットから既存の RDS Custom DB インスタンスに復元することはできません。復元するときに新しい RDS Custom for SQL Server DB インスタンスが作成されます。

スナップショットから復元すると、スナップショットが作成された時点までストレージボリュームが復元されます。これには、すべてのデータベースと、(`D:`) ボリュームに存在していた他のファイルが含まれます。

コンソール

DB スナップショットから RDS Custom DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. 復元の元にする DB スナップショットを選択します。
4. [アクション] で、[スナップショットの復元] を選択します。
5. DB インスタンスの復元 ページで、DB インスタンス識別子 に、復元した RDS Custom DB インスタンスの名前を入力します。
6. DB インスタンスの復元 を選択します。

AWS CLI

RDS Custom DB スナップショットを復元するには、DBスナップショットからDBインスタンスを復元する AWS CLI コマンドを使用します。

復元元のスナップショットがプライベート DB インスタンスの場合、`db-subnet-group-name` と`no-publicly-accessible` の両方を正しく指定してください。そうでなければ、DB インスタンスはデフォルトでパブリックアクセスに設定されます。以下のオプションは必須です。

- `db-snapshot-identifier` - 復元元のスナップショットを識別します。
- `db-instance-identifier` - DB スナップショットから作成する RDS Custom DB インスタンスの名前を指定します。
- `custom-iam-instance-profile` — RDS Custom DB インスタンスの基盤となる Amazon EC2 インスタンスに関連付けられているインスタンスプロファイルを指定します。

次のコードは、`my-custom-instance` の `my-custom-snapshot` という名前のスナップショットを復元します。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot \
```

```
  --db-instance-identifier my-custom-instance \
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
  --no-publicly-accessible
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot ^
  --db-instance-identifier my-custom-instance ^
  --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
  --no-publicly-accessible
```

RDS Custom for SQL Server インスタンスをポイントインタイムに復元する

DB インスタンスを特定の時点に復元し (PITR)、新しい DB インスタンスを作成できます。PITR をサポートするには、DB インスタンスのバックアップ保持が有効になっている必要があります。

RDS Custom for SQL Server DB インスタンスの最新の復元可能時間はいくつかの要因によって異なりますが、通常は現在時間から 5 分以内です。DB インスタンスの復元可能な直近の時間を確認するには、AWS CLI の describe-db-instances コマンドを使用し、DB インスタンスの [LatestRestorableTime] フィールドに返される値を確認します。Amazon RDS コンソールで各 DB インスタンスの復元可能な直近の時間を表示するには、[自動バックアップ] をクリックします。

バックアップ保持期間の任意の時点に復元できます。各 DB インスタンスの復元可能な最も早い時間を表示するには、Amazon RDS コンソールで [自動バックアップ] をクリックします。

PITR の一般情報については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

トピック

- RDS Custom for SQL Serverの PITR に関する考慮事項
- インスタンスクラスタイプごとに PITR の対象となるデータベースの数
- データベースを PITR の対象外にする
- Amazon S3 のトランザクションログ
- AWS マネジメントコンソール、AWS CLI、または RDS API を使用した PITR 復元

RDS Custom for SQL Serverの PITR に関する考慮事項

RDS Custom for SQL Serverでは、PITR は Amazon RDS の PITR と以下の重要な点で異なります。

- PITR は DB インスタンス内のデータベースのみを復元します。C: ドライブ上のオペレーティングシステムやファイルは復元されません。
- RDS Custom for SQL Server DB インスタンスの場合、データベースは自動的にバックアップされ、次の条件下でのみ PITR の対象となります。
  - データベースはオンラインです。
  - その回復モデルはFULLに設定されています。
  - 書き込み可能です。
  - D: ドライブに物理ファイルがあります。
  - `rds_pitr_blocked_databases`テーブルには記載されていません。詳細については、「データベースを PITR の対象外にする」を参照してください。
- PITR の対象となるデータベースは、データベース ID の順序によって決まります。RDS Custom for SQL Server では、DB インスタンスごとに最大 5,000 のデータベースを扱えます。ただし、RDS Custom for SQL Server DB インスタンスの PITR オペレーションで復元されるデータベースの最大数は、インスタンスクラスタイプによって異なります。詳細については、「インスタンスクラスタイプごとに PITR の対象となるデータベースの数」を参照してください。

  PITR には含まれない他のデータベースは、PITR で使用される自動スナップショットバックアップを含め、DB スナップショットから復元できます。
- 新しいデータベースの追加、データベースの名前変更、または PITR の対象となるデータベースの復元を行うと、DB インスタンスのスナップショットがスタートされます。
- PITR の対象となるデータベースの最大数は、ターゲットのインスタンスクラスタイプに応じて、データベースインスタンスでスケールコンピューティングオペレーションが実行される場合に変化します。インスタンスがスケールアップされ、インスタンス上のより多くのデータベースが PITR の対象になると、新しいスナップショットが作成されます。
- 復元されたデータベースの名前は、出典 DB インスタンスと同じです。必要に応じて、別の名前を指定できます。
- `AWSRDSCustomSQLServerIamRolePolicy` は、他の AWS サービスへのアクセスを必要とします。詳細については、「AWSRDSCustomsqlServerInstanceRole にアクセスポリシーを追加します。」を参照してください。
- タイムゾーンの変更は、 RDS Custom for SQL Server ではサポートされていません。OSまたは DB インスタンスのタイムゾーンを変更すると、PITR (およびその他のオートメーション) は機能しません。

インスタンスクラスタイプごとに PITR の対象となるデータベースの数

次の表は、インスタンスクラスタイプに基づいて PITR の対象となるデータベースの最大数を示しています。

インスタンスクラスのタイプ	PITR 対象のデータベースの最大数
db.*.large	100
db.*.xlarge〜db.*.2xlarge	150
db.*.4xlarge〜db.*.8xlarge	300
db.*.12xlarge〜db.*.16xlarge	600
db.*.24xlarge、db.*32xlarge	1,000

* さまざまなインスタンスクラスタイプを表します。

1 つの DB インスタンスで PITR の対象となるデータベースの最大数は、インスタンスクラスタイプによって異なります。数値の範囲は、RDS Custom for SQL Server でサポートされる最小のインスタンスクラスタイプで 100、最大のインスタンスクラスタイプで 1000 に及びます。SQL Server システムデータベース (`master`, `model`, `msdb`, `tempdb`) は、この制限には含まれていません。ターゲットインスタンスクラスタイプに応じて DB インスタンスがスケールアップ/スケールダウンされると、RDS Custom は PITR の対象となるデータベースの数を自動的に更新します。RDS Custom for SQL Server は、DB インスタンスで PITR の対象となるデータベースの最大数が変更されると `RDS-EVENT-0352` を送信します。詳細については、「カスタムエンジンバージョンイベント」を参照してください。

> **Note**
>
> 100 を超えるデータベースの PITR サポートは、2023 年 8 月 26 日以降に作成された DB インスタンスでのみ使用できます。2023 年 8 月 26 日より前に作成されたインスタンスの場合、PITR の対象となるデータベースの最大数は、インスタンスクラスに関係なく 100 です。2023 年 8 月 26 日より前に作成された DB インスタンスで 100 を超えるデータベースの PITR サポートを有効にするには、次のアクションを実行できます。
>
> - DB エンジンバージョンを 15.00.4322.2.v1 以降にアップグレードする

PITR オペレーション中、RDS Custom は復元時にソース DB インスタンスの PITR に含まれていたすべてのデータベースを復元します。ターゲット DB インスタンスが復元オペレーションを完了すると、バックアップ保持が有効になっている場合は、ターゲット DB インスタンスで PITR の対象となるデータベースの最大数に基づいて DB インスタンスのバックアップが開始されます。

例えば、DB インスタンスが 200 個のデータベースを持つ `db.*.xlarge` で実行されている場合：

1. RDS Custom for SQL Server は、PITR バックアップ用に、データベース ID 順に最初の 150 個のデータベースを選択します。
2. インスタンスを変更して db.*.4xlarge にスケールアップします。
3. スケールコンピューティングオペレーションが完了すると、RDS Custom for SQL Server は PITR バックアップ用に、データベース ID 順に最初の 300 個のデータベースを選択します。PITR の要件を満たす 200 個のデータベースのそれぞれが PITR の対象になります。
4. 次に、インスタンスを変更して db.*.xlarge にスケールダウンします。
5. スケールコンピューティングオペレーションが完了すると、RDS Custom for SQL Server は、PITR バックアップ用に、データベース ID 順に最初の 150 個のデータベースを再び選択します。

データベースを PITR の対象外にする

個々のデータベースを PITR から除外することもできます。これには、`database_id`値を`rds_pitr_blocked_databases`テーブルに入力します。以下のSQLスクリプトを使用してテーブルを作成します。

rds_pitr_blocked_databases テーブルを作成するには

- 次のSQLスクリプトを実行します。

```
create table msdb..rds_pitr_blocked_databases
(
database_id INT NOT NULL,
database_name SYSNAME NOT NULL,
db_entry_updated_date datetime NOT NULL DEFAULT GETDATE(),
db_entry_updated_by SYSNAME NOT NULL DEFAULT CURRENT_USER,
PRIMARY KEY (database_id)
);
```

対象となるデータベースと対象とならないデータベースの一覧は、Amazon S3 バケットdo-not-delete-rds-custom-*$ACCOUNT_ID*-*$REGION*-*unique_identifier*内のRDSCustomForSQLServer/Instances/*DB_instance_resource_ID*/TransactionLogMetadataディレクトリにあるRI.Endファイルを参照してください。RI.Endファイルの詳細については、「Amazon S3 のトランザクションログ」を参照してください。

次の SQL スクリプトを使用して、PITR の対象となるデータベースのリストを確認することもできます。@limit 変数を、インスタンスクラスで PITR の対象となるデータベースの最大数に設定します。詳細については、「インスタンスクラスタイプごとに PITR の対象となるデータベースの数」を参照してください。

DB インスタンスクラスで PITR の対象となるデータベースのリストを確認するには

- 次のSQLスクリプトを実行します。

```
DECLARE @Limit INT;
SET @Limit = (insert-database-instance-limit-here);

USE msdb;
IF (EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'dbo' AND
 TABLE_NAME = 'rds_pitr_blocked_databases'))
    WITH TABLE0 AS (
        SELECT hdrs.database_id as DatabaseId, sdb.name as DatabaseName,
 'ALWAYS_ON_NOT_WRITABLE_REPLICA' as Reason, NULL as DatabaseNameOnPitrTable
        FROM sys.dm_hadr_database_replica_states hdrs
        INNER JOIN sys.databases sdb ON sdb.database_id = hdrs.database_id
        WHERE (hdrs.is_local = 1 AND hdrs.is_primary_replica = 0)
        OR (sys.fn_hadr_is_primary_replica (sdb.name) = 1 AND DATABASEPROPERTYEX
 (sdb.name, 'Updateability') = 'READ_ONLY')
    ),
    TABLE1 as (
```

```
            SELECT dbs.database_id as DatabaseId, sysdbs.name as DatabaseName,
'OPTOUT' as Reason,
            CASE WHEN dbs.database_name = sysdbs.name THEN NULL ELSE
dbs.database_name END AS DatabaseNameOnPitrTable
            FROM msdb.dbo.rds_pitr_blocked_databases dbs
            INNER JOIN sys.databases sysdbs ON dbs.database_id = sysdbs.database_id
            WHERE sysdbs.database_id > 4
            ),
   TABLE2 as (
            SELECT
            db.name AS DatabaseName,
            db.create_date AS CreateDate,
            db.state_desc AS DatabaseState,
            db.database_id AS DatabaseId,
            rs.database_guid AS DatabaseGuid,
            rs.last_log_backup_lsn AS LastLogBackupLSN,
            rs.recovery_fork_guid RecoveryForkGuid,
            rs.first_recovery_fork_guid AS FirstRecoveryForkGuid,
            db.recovery_model_desc AS RecoveryModel,
            db.is_auto_close_on AS IsAutoClose,
            db.is_read_only as IsReadOnly,
            NEWID() as FileName,
            CASE WHEN(db.state_desc = 'ONLINE'
                    AND db.recovery_model_desc != 'SIMPLE'
                    AND((db.is_auto_close_on = 0 and db.collation_name IS NOT NULL)
OR db.is_auto_close_on = 1))
                    AND db.is_read_only != 1
                    AND db.user_access = 0
                    AND db.source_database_id IS NULL
                    AND db.is_in_standby != 1
                    THEN 1 ELSE 0 END AS IsPartOfSnapshot,
            CASE WHEN db.source_database_id IS NULL THEN 0 ELSE 1 END AS
IsDatabaseSnapshot
            FROM sys.databases db
            INNER JOIN sys.database_recovery_status rs
            ON db.database_id = rs.database_id
            WHERE DB_NAME(db.database_id) NOT IN('tempdb') AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE1) AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE0)
       ),
       TABLE3 as(
            Select @Limit+count(DatabaseName) as TotalNumberOfDatabases from TABLE2
where TABLE2.IsPartOfSnapshot=1 and DatabaseName in ('master','model','msdb')
       )
```

```
        SELECT TOP(SELECT TotalNumberOfDatabases from TABLE3)
 DatabaseName,CreateDate,DatabaseState,DatabaseId from TABLE2 where
 TABLE2.IsPartOfSnapshot=1
        ORDER BY TABLE2.DatabaseID ASC
ELSE
    WITH TABLE0 AS (
        SELECT hdrs.database_id as DatabaseId, sdb.name as DatabaseName,
 'ALWAYS_ON_NOT_WRITABLE_REPLICA' as Reason, NULL as DatabaseNameOnPitrTable
        FROM sys.dm_hadr_database_replica_states hdrs
        INNER JOIN sys.databases sdb ON sdb.database_id = hdrs.database_id
        WHERE (hdrs.is_local = 1 AND hdrs.is_primary_replica = 0)
        OR (sys.fn_hadr_is_primary_replica (sdb.name) = 1 AND DATABASEPROPERTYEX
 (sdb.name, 'Updateability') = 'READ_ONLY')
    ),
    TABLE1 as (
            SELECT
            db.name AS DatabaseName,
            db.create_date AS CreateDate,
            db.state_desc AS DatabaseState,
            db.database_id AS DatabaseId,
            rs.database_guid AS DatabaseGuid,
            rs.last_log_backup_lsn AS LastLogBackupLSN,
            rs.recovery_fork_guid RecoveryForkGuid,
            rs.first_recovery_fork_guid AS FirstRecoveryForkGuid,
            db.recovery_model_desc AS RecoveryModel,
            db.is_auto_close_on AS IsAutoClose,
            db.is_read_only as IsReadOnly,
            NEWID() as FileName,
            CASE WHEN(db.state_desc = 'ONLINE'
                    AND db.recovery_model_desc != 'SIMPLE'
                    AND((db.is_auto_close_on = 0 and db.collation_name IS NOT NULL)
 OR db.is_auto_close_on = 1))
                    AND db.is_read_only != 1
                    AND db.user_access = 0
                    AND db.source_database_id IS NULL
                    AND db.is_in_standby != 1
                    THEN 1 ELSE 0 END AS IsPartOfSnapshot,
            CASE WHEN db.source_database_id IS NULL THEN 0 ELSE 1 END AS
 IsDatabaseSnapshot
            FROM sys.databases db
            INNER JOIN sys.database_recovery_status rs
            ON db.database_id = rs.database_id
            WHERE DB_NAME(db.database_id) NOT IN('tempdb') AND
            db.database_id NOT IN (SELECT DISTINCT DatabaseId FROM TABLE0)
```

```
        ),
        TABLE2 as(
            SELECT @Limit+count(DatabaseName) as TotalNumberOfDatabases from TABLE1
where TABLE1.IsPartOfSnapshot=1 and DatabaseName in ('master','model','msdb')
        )
        select top(select TotalNumberOfDatabases from TABLE2)
DatabaseName,CreateDate,DatabaseState,DatabaseId from TABLE1 where
TABLE1.IsPartOfSnapshot=1
        ORDER BY TABLE1.DatabaseID ASC
```

> **Note**
>
> シンボリックリンクのみのデータベースも、PITR オペレーションの対象となるデータベースから除外されます。上記のクエリでは、この基準に基づいてフィルタリングされません。

Amazon S3 のトランザクションログ

バックアップ保持期間は、RDS Custom for SQL Server DB インスタンスのトランザクションログが自動的に抽出され、Amazon S3 にアップロードされるかどうかを決定します。ゼロ以外の値は、自動バックアップが作成され、RDS Custom エージェントが 5 分ごとにトランザクションログを S3 にアップロードすることを意味します。

S3 上のトランザクションログファイルは、AWS KMS keyDB インスタンス作成時に提供したものを使って、静止時に暗号化されます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「サーバー側の暗号化を使用したデータの保護」を参照してください。

各データベースのトランザクションログは、do-not-delete-rds-custom-*$ACCOUNT_ID*-*$REGION*-*unique_identifier*という名前の S3 バケットにアップロードされます。S3 バケットのRDSCustomForSQLServer/Instances/*DB_instance_resource_ID*ディレクトリには、2 つのサブディレクトリが含まれます。

- TransactionLogs - 各データベースのトランザクションログとそれぞれのメタデータが含まれます。

  トランザクションログファイル名は、例えば*yyyyMMddHHmm*.*database_id*.*timestamp*というパターンに従っています。

```
202110202230.11.1634769287
```

同じファイル名でサフィックスが_metadataのものには、ログシーケンス番号、データベース名、RdsChunkCountなどのトランザクションログに関する情報が含まれます。RdsChunkCountは、1つのトランザクションログファイルを表す物理ファイルの数を決定します。_0001、_0002などのサフィックスを持つファイルがありますが、これはトランザクションログファイルの物理的チャンクを意味します。チャンクされたトランザクションログファイルを使用する場合は、ダウンロード後にチャンクの結合を必ず行ってください。

以下のファイルがある場合を考えてみましょう。

- 202110202230.11.1634769287
- 202110202230.11.1634769287_0001
- 202110202230.11.1634769287_0002
- 202110202230.11.1634769287_metadata

RdsChunkCount は、3 です。ファイルをマージする順序は202110202230.11.1634769287、202110202230.11.1634769287_0001、202110202230.11.1634769287_0002です。

- TransactionLogMetadata - トランザクションログ抽出の各反復処理についてのメタデータ情報が含まれます。

  RI.Endファイルには、トランザクションログが抽出されたすべてのデータベース、存在するがトランザクションログが抽出されていないすべてのデータベースについての情報が含まれます。RI.Endファイル名はパターン*yyyyMMddHHmm*.RI.End.*timestamp*に従います。例えば、

  ```
  202110202230.RI.End.1634769281
  ```

AWS マネジメントコンソール、AWS CLI、または RDS API を使用した PITR 復元

AWS マネジメントコンソール、AWS CLI、またはRDS API を使用して、 RDS Custom for SQL Server DB インスタンスをあるポイントに復元することができます。

コンソール

特定の時点に RDS Custom DB インスタンスを復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「自動バックアップ」を選択します。
3. 復元する RDS Custom DB インスタンスを選択します。
4. 「アクション」 で、「特定時点への復元」 を選択します。

   [特定時点への復元] ウィンドウが表示されます。
5. 「Latest restorable time」 を選択してできるだけ最新の時点に復元するか、「カスタム」 を選択して時刻を選択します。

   「カスタム」 を選択した場合、インスタンスクラスターを復元する日時を入力します。

   時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。
6. 「DB インスタンス識別子」 に、ターゲットが復元された RDS Custom DB インスタンスの名前を入力します。名前は一意である必要があります。
7. 必要に応じて、DB インスタンスクラスなどの他のオプションを選択します。
8. [Restore to point in time] (特定時点への復元) を選択します。

AWS CLI

特定のポイントに DB インスタンスを復元するには、restore-db-instance-to-point-in-time AWS CLIコマンドを使用して、RDS Custom DB インスタンスを作成します。

次のいずれかのオプションを使用して、復元元のバックアップを指定します。

- `--source-db-instance-identifier` *`mysourcedbinstance`*
- `--source-dbi-resource-id` *`dbinstanceresourceID`*
- `--source-db-instance-automated-backups-arn` *`backupARN`*

`custom-iam-instance-profile` オプションは必須です。

次の例は、指定された時刻に my-custom-db-instance を my-restored-custom-db-instance という新しい DB インスタンスに復元します。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my-custom-db-instance\
    --target-db-instance-identifier my-restored-custom-db-instance \
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance \
    --restore-time 2022-10-14T23:45:00.000Z
```

Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my-custom-db-instance ^
    --target-db-instance-identifier my-restored-custom-db-instance ^
    --custom-iam-instance-profile AWSRDSCustomInstanceProfileForRdsCustomInstance ^
    --restore-time 2022-10-14T23:45:00.000Z
```

RDS Custom for SQL Server スナップショットの削除

RDS Custom for SQL Server で管理しているDBスナップショットは、不要になったら削除することができます。削除手順は、Amazon RDS および RDS Custom DB インスタンスのどちらでも同じです。

バイナリボリュームとルートボリュームの Amazon EBS スナップショットは、アカウントで実行されているいくつかのインスタンスや、他の RDS Custom for SQL Server スナップショットにリンクされている可能性があるため、長期間アカウントに残ります。これらの EBS スナップショットは、既存の RDS Custom for SQL Server リソース (DB インスタンスまたはバックアップ) に関連しなくなった後に自動的に削除されます。

コンソール

RDS Custom DB インスタンスのスナップショットを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。

3. 削除する DB スナップショットを選択します。
4. 「アクション」 で、「スナップショットの削除」 を選択します。
5. 確認ページで、「削除」 を選択します。

AWS CLI

RDS Custom スナップショットを削除するには、AWS CLIコマンドdelete-db-snapshotを使用します。

以下のような必須オプションがあります。

- `--db-snapshot-identifier` - 削除するスナップショット

以下の例では、DB スナップショット `my-custom-snapshot` を削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-snapshot \
  --db-snapshot-identifier my-custom-snapshot
```

Windows の場合:

```
aws rds delete-db-snapshot ^
  --db-snapshot-identifier my-custom-snapshot
```

RDS Custom for SQL Server 自動バックアップの削除

RDS Custom for SQL Server の保持された自動バックアップは、不要になったら削除できます。手順は、Amazon RDS バックアップの削除と同じです。

コンソール

保持されている自動バックアップを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups] (自動バックアップ) を選択します。

3. [Retained] (保持) を選択します。
4. 削除する保持された自動バックアップを選択します。
5. 「アクション」 で、「削除」 を選択します。
6. 確認ページで、**`delete me`** を入力し、[Delete] (削除) を選択します。

AWS CLI

AWS CLI コマンド delete-db-instance-automated-backup を使用して、保持されている自動バックアップを削除できます。

保持されている自動バックアップを削除するには、以下のオプションを使用します。

- `--dbi-resource-id` - 出典 RDS Custom DB インスタンスのリソース識別子です。

  AWS CLI コマンド describe-db-instance-automated-backups を使用すると、保持された自動バックアップの出典 DB インスタンスのリソース識別子を見つけることができます。

次の例では、ソース DB インスタンスのリソース識別子 `custom-db-123ABCEXAMPLE` を持つ保持された自動バックアップを削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance-automated-backup \
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Windows の場合:

```
aws rds delete-db-instance-automated-backup ^
    --dbi-resource-id custom-db-123ABCEXAMPLE
```

Amazon RDS Custom for SQL Server DB スナップショットのコピー

RDS Custom for SQL Server を使用すると、自動バックアップまたは手動 DB スナップショットをコピーできます。スナップショットをコピーすると、作成するコピーは手動スナップショットになります。自動バックアップまたは手動スナップショットは複数のコピーを作成できますが、各コピーには一意の識別子が必要です。

RDS Custom for SQL Server が利用可能な異なる AWS リージョン の同じ AWS アカウント内でのみスナップショットをコピーできます。以下のオペレーションは現在サポートされていません。

- 同じ AWS リージョン 内で DB スナップショットをコピーする。
- AWS アカウント間で DB スナップショットをコピーする。

RDS Custom for SQL Server は、増分スナップショットコピーをサポートしています。詳細については、「増分スナップショットのコピーに関する考慮事項」を参照してください。

トピック

- 制限事項
- 暗号化の処理
- リージョン間のコピー
- カスタムエンジンバージョン (CEV) で作成された DB インスタンスのスナップショット
- IAM プリンシパルに必要なアクセス権限を付与します。
- DB スナップショットのコピー

制限事項

RDS Custom for SQL Server の DB スナップショットのコピーには、次の制限が適用されます。

- ターゲットスナップショットが使用可能になる前に出典スナップショットを削除すると、スナップショットはコピーされない場合があります。ターゲットスナップショットのステータスが `AVAILABLE` になったことを確認してから、ソーススナップショットを削除してください。
- DB スナップショットのコピーリクエストでオプショングループ名を指定したり、オプショングループをコピーしたりすることはできません。
- コピープロセス前またはコピープロセス中にソース DB スナップショットの依存 AWS リソースを削除すると、コピースナップショットリクエストが非同期的に失敗する可能性があります。

  - アカウントの RDS Custom マネージド S3 バケットに保存されているソース DB インスタンスのサービスマスターキー (SMK) バックアップファイルを削除すると、DB スナップショットのコピーは非同期的に成功します。ただし、TDE が有効になっているデータベースなど、SQL Server の機能は SMK に依存するため、問題が発生します。詳細については、「RDS Custom for SQL Server の TDE が有効なデータベースの PENDING_RECOVERY 状態のトラブルシューティング」を参照してください。
- 同じ AWS リージョン 内の DB スナップショットのコピーは現在サポートされていません。
- AWS アカウント間での DB スナップショットのコピーは現在サポートされていません。

Amazon RDS の DB スナップショットのコピーの制限は、RDS Custom for SQL Server にも適用されます。詳細については、「制限事項」を参照してください。

暗号化の処理

すべての RDS Custom for SQL Server DB インスタンスと DB スナップショットは KMS キーで暗号化されます。暗号化されたスナップショットは暗号化されたスナップショットにのみコピーできるため、DB スナップショットのコピーリクエストのコピー先 AWS リージョン で有効な KMS キーを指定する必要があります。

ソーススナップショットはコピープロセス全体で暗号化されたままになります。Amazon RDS は、エンベロープ暗号化を使用して、指定されたコピー先の AWS リージョン KMS キーによるコピーオペレーション中にデータを保護します。詳細については、「AWS Key Management Service デベロッパーガイド」の「エンベロープ暗号化」を参照してください。

リージョン間のコピー

AWS リージョン 間で DB スナップショットをコピーできます。ただし、クロスリージョンのスナップショットコピーには、特定の制約と考慮事項があります。

RDS がスナップショットのコピーのために AWS リージョン 間で通信することを許可する

リージョン間の DB スナップショットのコピーリクエストが正常に処理されると、RDS はコピーを開始します。RDS がソーススナップショットにアクセスするための認証リクエストが作成されます。この認証リクエストは、ソース DB スナップショットをターゲット DB スナップショットにリンクします。これにより、RDS は指定されたターゲットスナップショットにのみコピーできます。

RDS は、サービスにリンクされた IAM ロールの `rds:CrossRegionCommunication` アクセス許可を使用して認証を検証します。コピーが認証されると、RDS はコピー元リージョンと通信し、コピーオペレーションを完了します。

RDS は、CopyDBSnapshot リクエストによって以前に許可されていない DB スナップショットにアクセスできません。コピーが完了すると、認証は取り消されます。

RDS は、サービスリンクされたロールを使用して、出典リージョンでの承認を確認します。コピープロセス中にサービスリンクロールを削除すると、コピーは失敗します。

詳細については、「AWS Identity and Access Management ユーザーガイド」の「サービスにリンクされたロールの使用」を参照してください。

AWS Security Token Service 認証情報の使用

グローバル AWS Security Token Service (AWS STS) エンドポイントからのセッショントークンは、デフォルトで有効になっている AWS リージョン (商用リージョン) でのみ有効です。`assumeRole` の AWS STS API 操作からの認証情報を使用する場合、出典リージョンがオプトインリージョンである場合は、そのリージョンのエンドポイントを使用します。それ以外の場合、このリクエストは失敗します。認証情報が両方のリージョンで有効である必要があります。これは、そのリージョンの AWS STS エンドポイントが使用されている場合のみオプトインリージョンに当てはまります。

グローバルエンドポイントを使用するには、オペレーションで両方のリージョンで有効になっていることを確認します。AWS STS アカウント設定のすべての AWS リージョン でグローバルエンドポイントを `Valid` に設定します。

詳細については、「AWS Identity and Access Management ユーザーガイド」の「AWS リージョンでの AWS STS の管理」を参照してください。

カスタムエンジンバージョン (CEV) で作成された DB インスタンスのスナップショット

カスタムエンジンバージョン (CEV) を使用する DB インスタンスの DB スナップショットの場合、RDS は CEV を DB スナップショットに関連付けます。CEV に関連付けられたソース DB スナップショットを AWS リージョン 間でコピーするために、RDS はソース DB スナップショットとともに CEV をコピー先リージョンにコピーします。

同じ CEV に関連付けられた複数の DB スナップショットを同じコピー先リージョンにコピーする場合、最初のコピーリクエストは関連付けられた CEV をコピーします。次のリクエストのコピープロセスでは、最初にコピーされた CEV を検索し、それを次の DB スナップショットコピーに関連付けます。DB スナップショットコピーに関連付けるには、既存の CEV コピーが `AVAILABLE` 状態である必要があります。

CEV に関連付けられた DB スナップショットをコピーするには、リクエスタの IAM ポリシーに、DB スナップショットのコピーと関連付けられた CEV のコピーの両方を許可するアクセス許可が必要で

す。関連付けられた CEV のコピーを許可するには、リクエスタの IAM ポリシーで次のアクセス許可が必要です。

- `rds:CopyCustomDBEngineVersion` - リクエスタ IAM プリンシパルには、ソース DB スナップショットとともにソース CEV をコピー先リージョンにコピーするアクセス許可が必要です。リクエスタ IAM プリンシパルがソース CEV のコピーを認証されていない場合、スナップショットのコピーリクエストは認証エラーにより失敗します。
- `ec2:CreateTags` - ソース CEV の基盤となる EC2 AMI は、CEV のコピーの一部としてコピー先リージョンにコピーされます。RDS Custom は、AMI をコピーする前に AMI に `AWSRDSCustom` タグを付けようとします。リクエスタ IAM プリンシパルに、ソースリージョンのソース CEV の基盤となる AMI に対してタグを作成するアクセス許可があることを確認します。

CEV コピーのアクセス許可の設定については、「IAM プリンシパルに必要なアクセス権限を付与します。」を参照してください。

IAM プリンシパルに必要なアクセス権限を付与します。

RDS Custom for SQL Server DB スナップショットをコピーするための十分なアクセス権があることを確認してください。コンソールまたは CLI を使用して DB スナップショットをコピーするための IAM ロールまたは IAM ユーザー (IAM プリンシパル) には、DB インスタンスを正常に作成するために、次のいずれかのポリシーが必要です。

- `AdministratorAccess` ポリシー、または
- 追加のアクセス許可を持つ`AmazonRDSFullAccess`ポリシーです。

```
s3:CreateBucket
s3:GetBucketPolicy
s3:PutBucketPolicy
kms:CreateGrant
kms:DescribeKey
ec2:CreateTags
```

RDS Custom は、AWS リージョン 間でスナップショットをコピーするときにこれらのアクセス許可を使用します。これらのアクセス許可は、RDS Custom オペレーションに必要なアカウントのリソースを設定します。`kms:CreateGrant` アクセス許可の詳細については、「AWS KMS key 管理」を参照してください。

次のサンプル JSON ポリシーにより、AmazonRDSFullAccess ポリシーの他に必要なアクセス許可が付与されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateS3BucketAndReadWriteBucketPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::do-not-delete-rds-custom-*"
        },
        {
            "Sid": "CreateKmsGrant",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateEc2Tags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> リストされたアクセス許可が、IAM プリンシパルに関連付けられたサービスコントロールポリシー (SCP)、アクセス許可の境界、またはセッションポリシーによって制限されていないことを確認します。

リクエスタの IAM ポリシーでコンテキストキーを使用した条件を指定すると、特定の条件によってリクエストが失敗する可能性があります。IAM ポリシー条件による一般的な落とし穴についての詳細は、「クロスリージョン DB スナップショットコピーのリクエスト」を参照してください。

DB スナップショットのコピー

DB スナップショットをコピーするには、次の手順に従います。AWS アカウントごとに、AWS リージョン 間で同時に最大 20 個の DB スナップショットをコピーできます。別の AWS リージョン に DB スナップショットをコピーする場合には、その AWS リージョン に保持されている手動 DB スナップショットを作成します。コピー元の AWS リージョン から DB スナップショットをコピーすると、Amazon RDS のデータ転送料金が発生します。データ転送料金の詳細については、「Amazon RDS の料金」を参照してください。

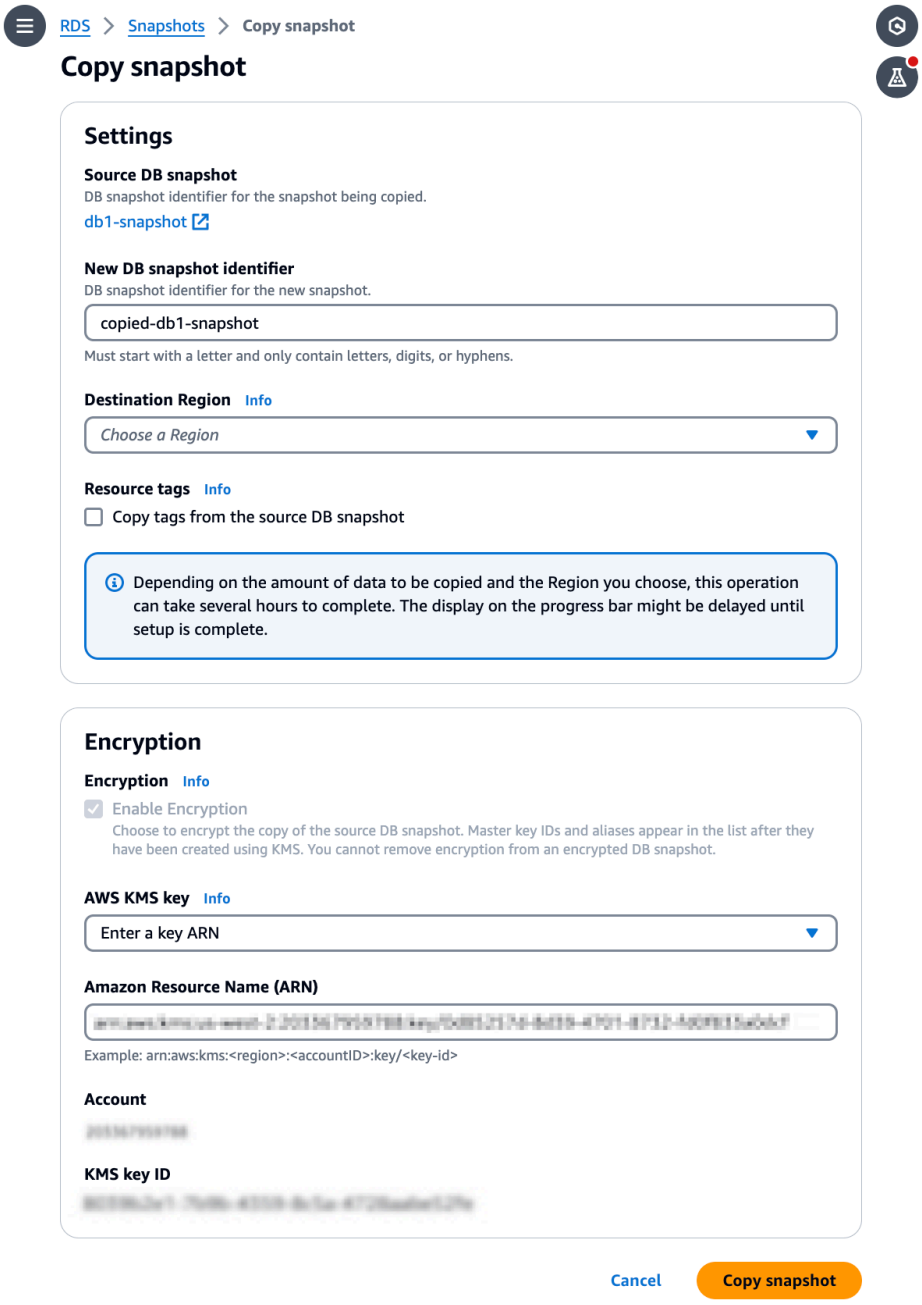
DB スナップショットのコピーが新しい AWS リージョン に作成されると、その DB スナップショットのコピーは、AWS リージョン にある他のすべての DB スナップショットと同じように動作します。

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して DB スナップショットをコピーできます。

Console

次の手順では、AWS マネジメントコンソール を使用して RDS Custom for SQL Server DB スナップショットをコピーします。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. コピーする RDS Custom for SQL Server DB スナップショットを選択します。
4. [アクション] ドロップダウンで、[スナップショットをコピー] を選択します。



5. DB スナップショットを別の AWS リージョン にコピーするには、[送信先リージョン] を必要な値に設定します。

> **Note**
>
> コピー先 AWS リージョン は、コピー元 AWS リージョン と同じデータベースエンジンのバージョンを使用できる必要があります。

6. [新しい DB スナップショット ID] に DB スナップショットの一意の名前を入力します。自動バックアップまたは手動スナップショットは複数のコピーを作成できますが、各コピーには一意の識別子が必要です。
7. (オプション) スナップショットからスナップショットのコピーにタグと値をコピーするには、[Copy Tags] を選択します。
8. [暗号化] で、DB スナップショットコピーの暗号化に使用する KMS キー識別子を指定します。

   > **Note**
   >
   > RDS Custom for SQL Server は、すべての DB スナップショットを暗号化します。暗号化されていない DB スナップショットを作成することはできません。

9. [スナップショットをコピー] を選択します。

RDS Custom for SQL Server は、選択した AWS リージョン に DB インスタンスの DB スナップショットコピーを作成します。

AWS CLI

AWS CLI の copy-db-snapshot コマンドを使用して、RDS Custom for SQL Server DB スナップショットをコピーできます。スナップショットのコピー先が新しい AWS リージョン である場合は、その新しい AWS リージョン でコマンドを実行します。DB スナップショットをコピーするには、以下のオプションを使用します。状況に応じてオプションを使い分けることができます。

- `--source-db-snapshot-identifier` - ソース DB スナップショットの識別子。
  - スナップショットのコピー元とコピー先が異なる AWS リージョン である場合は、有効な DB スナップショットの ARN を指定します。例: `arn:aws:rds:us-west-2:123456789012:snapshot:instance1-snapshot-12345678`
- `--target-db-snapshot-identifier` - DB スナップショットの新しいコピーの識別子。
- `--kms-key-id` - 暗号化された DB スナップショットの KMS キー識別子。KMS キー識別子は、KMS キーの Amazon リソースネーム (ARN)、キー識別子、またはキーエイリアスです。
  - 暗号化されたスナップショットを別の AWS リージョン にコピーする場合は、コピー先 AWS リージョン の KMS キーを指定する必要があります。KMS キーは、それらが作成された AWS リージョン に固有のものであるため、マルチリージョンキーを使用する場合を除き、ある AWS リージョン の暗号化キーを別の AWS リージョン で使用することはできま

せん。マルチリージョンの KMS キーの詳細については、「AWS KMS でマルチリージョンキーを使用する」を参照してください。

- `--copy-tags` - ソーススナップショットからスナップショットのコピーにのタグと値を含めます。

RDS Custom for SQL Server DB スナップショットのコピーでは、以下のオプションはサポートされていません。

- `--copy-option-group`
- `--option-group-name`
- `--pre-signed-url`
- `--target-custom-availability-zone`

次のコード例では、米国西部 (オレゴン) リージョンから米国東部 (バージニア北部) リージョンに暗号化された DB スナップショットをコピーします。コピー先リージョン (us-east-1) でコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds copy-db-snapshot \
     --region us-east-1 \
    --source-db-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:snapshot:instance1-snapshot-12345678 \
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --kms-key-id a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
```

Windows の場合:

```
aws rds copy-db-snapshot ^
     --region us-east-1 ^
    --source-db-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:snapshot:instance1-snapshot-12345678 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --kms-key-id a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
```

RDS API

Amazon RDS API の CopyDBSnapshot オペレーションを使用して、RDS Custom for SQL Server DB スナップショットをコピーできます。スナップショットのコピー先が新しい AWS リージョン

である場合は、その新しい AWS リージョン でアクションを実行します。DB スナップショットをコピーするには、以下のパラメータを使用します。すべてのパラメータが必須なわけではありません。

- SourceDBSnapshotIdentifier - ソース DB スナップショットの識別子。
  - スナップショットのコピー元とコピー先が異なる AWS リージョン である場合は、有効な DB スナップショットの ARN を指定します。例: arn:aws:rds:us-west-2:123456789012:snapshot:instance1-snapshot-12345678
- TargetDBSnapshotIdentifier - DB スナップショットの新しいコピーの識別子。
- KmsKeyId - 暗号化された DB スナップショットの KMS キー識別子。KMS キー識別子は、KMS キーの Amazon リソースネーム (ARN)、キー識別子、またはキーエイリアスです。
  - 暗号化されたスナップショットを別の AWS リージョン にコピーする場合は、コピー先 AWS リージョン の KMS キーを指定する必要があります。KMS キーは、それらが作成された AWS リージョン に固有のものであるため、マルチリージョンキーを使用する場合を除き、ある AWS リージョン の暗号化キーを別の AWS リージョン で使用することはできません。マルチリージョンの KMS キーの詳細については、「AWS KMS でマルチリージョンキーを使用する」を参照してください。
- CopyTags - ソーススナップショットからスナップショットのコピーにタグと値をコピーするには、このパラメータを true に設定します。デフォルトは false です。

以下のオプションは、RDS Custom for SQL Server DB スナップショットのコピーではサポートされていません。

- CopyOptionGroup
- OptionGroupName
- PreSignedUrl
- TargetCustomAvailabilityZone

次のコードでは、スナップショットのコピーを mydbsnapshotcopy という新しい名前で米国東部 (バージニア北部) リージョンに作成します。

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBSnapshot
    &KmsKeyId=a1b2c3d4-1234-5678-wxyz-a1b2c3d4d5e6
    &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Asnapshot%3Ainstance1-snapshot-12345678
```

```
    &TargetDBSnapshotIdentifier=mydbsnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
    &X-Amz-
Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

オンプレミスデータベースを Amazon RDS Custom for SQL Server に移行する

次の手順で、ネイティブバックアップと復元を使用し、オンプレミス Microsoft SQL Server データベースを Amazon RDS Custom for SQL Server に移行できます。

1. オンプレミス DB インスタンスでデータベースのフルバックアップを作成します。
2. Amazon S3 にバックアップファイルをアップロードします。
3. S3 から RDS Custom for SQL Server DB インスタンスにバックアップファイルをダウンロードします。
4. RDS Custom for SQL Server DB インスタンスで、ダウンロードしたバックアップファイルを使用してデータベースを復元します。

このプロセスでは、ネイティブのフルバックアップと復元を使用して、オンプレミスから RDS Custom for SQL Server へのデータベースの移行について説明します。移行プロセス中のカットオーバー時間を短縮するために、差分バックアップまたはログバックアップの使用を検討してもよいでしょう。

RDS for SQL Server のネイティブバックアップおよびリストアに関する一般的な情報については、ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート を参照してください。

トピック

- 前提条件
- オンプレミスデータベースのバックアップ
- Amazon S3 へのバックアップファイルのアップロード
- Amazon S3 からのバックアップファイルのダウンロード
- RDS Custom for SQL Server DB インスタンスへのバックアップファイルの復元

前提条件

データベースを移行する前に、次のタスクを実行します。

1. RDS Custom for SQL Server DB インスタンスに、リモートデスクトップ接続 (RDP) を設定します。詳細については、「RDP を使用した RDS Custom DB インスタンスへの接続」を参照してください。

2. Amazon S3 へのアクセス権を設定して、データベースのバックアップファイルをアップロードおよびダウンロードできるようにします。

オンプレミスデータベースのバックアップ

SQL Server ネイティブバックアップを使用して、オンプレミス DB インスタンス上のデータベースのフルバックアップを作成します。

次の例は、mydatabase という名前のデータベースのバックアップで、バックアップファイルのサイズを小さくするために指定される COMPRESSION オプションを示しています。

オンプレミスデータベースをバックアップするには

1. SQL Server Management Studio (SSMS) を使用して、オンプレミス SQL Server インスタンスに接続します。
2. 次の T-SQL コマンドを実行します。

```
backup database mydatabase to
disk ='C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Backup\mydb-full-compressed.bak'
with compression;
```

Amazon S3 へのバックアップファイルのアップロード

バックアップファイルを mydb-full-compressed.bak Amazon S3 にアップロードするには AWS マネジメントコンソール を使用します。

S3 にバックアップファイルをアップロードするには

1. AWS マネジメントコンソール にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
2. [Buckets] (バケット) では、バックアップファイルのアップロード先のバケットの名前を選択します。
3. [Upload] (アップロード) を選択します。
4. [Upload] (アップロード) ウィンドウで、次のいずれかの操作を行います。
   - mydb-full-compressed.bak を アップロードウインドウにドラッグアンドドロップします。

- ファイルを追加するを選択し、「mydb-full-compressed.bak」を選択してから、オープンを選択します。

Amazon S3 はバックアップファイルを S3 オブジェクトとしてアップロードします。アップロードが完了すると、[Upload: status] (アップロード: ステータス) ページに成功のメッセージが表示されます。

Amazon S3 からのバックアップファイルのダウンロード

コンソールを使用して、S3 から RDS Custom for SQL Server DB インスタンスにバックアップファイルをダウンロードします。

S3 からバックアップファイルをダウンロードするには

1. RDP を使用して、RDS Custom for SQL Server DB インスタンスに接続します。
2. AWS マネジメントコンソール にサインインし、Amazon S3 コンソール https://console.aws.amazon.com/s3/ を開きます。
3. [Buckets] (バケット)リストで、バックアップファイルのアップロード先のバケットの名前を選択します。
4. バックアップファイルmydb-full-compressed.bakを選択します。
5. 「アクション」で、「名前を付けてダウンロード」を選択します。
6. 表示されたリンクのコンテキスト (右クリック) メニューを開き、名前を付けて保存 を選択します。
7. mydb-full-compressed.bakをD:\rdsdbdata\BACKUP ディレクトリに保存します。

RDS Custom for SQL Server DB インスタンスへのバックアップファイルの復元

SQL Server ネイティブ復元を使用して、バックアップファイルを RDS Custom for SQL Server DB インスタンスに復元します。

この例では、データファイルとログファイルのディレクトリがオンプレミス DB インスタンスと異なるため、MOVE オプションが指定されます。

バックアップファイルを復元するには

1. SSMS を使用して、RDS Custom for SQL Server DB インスタンスに接続します。

2. T-SQL コマンドを実行します。

```
restore database mydatabase from disk='D:\rdsdbdata\BACKUP\mydb-full-
compressed.bak'
with move 'mydatabase' to 'D:\rdsdbdata\DATA\mydatabase.mdf',
move 'mydatabase_log' to 'D:\rdsdbdata\DATA\mydatabase_log.ldf';
```

RDS Custom for SQL Server オペレーティングシステムの更新

RDS Custom for SQL Server には、RDS 提供のエンジンバージョン (RPEV) インスタンスにオペレーティングシステムの更新を適用するための以下の方法が用意されています。

- システム更新メンテナンスアクション
- データベースのマイナーバージョンアップグレード
  - RPEV を使用した DB マイナーエンジンバージョンのアップグレードには、オペレーティングシステムの最新の更新が含まれています。このアプローチは、OS の更新を SQL Server のマイナーバージョンアップグレードと組み合わせる場合に特に役立ちます。詳細については、「Amazon RDS Custom for SQL Server の DB インスタンスのアップグレード」を参照してください。

オペレーティングシステムの更新シナリオ

RDS Custom for SQL Server インスタンスのオペレーティングシステムの更新を管理するには、次の 2 つの方法があります。

- シングル AZ インスタンスの場合、オペレーティングシステムの更新中はインスタンスを使用できません。
- マルチ AZ 配置の場合、RDS は次の方法でオペレーティングシステムの更新を適用します。
  - まず、RDS はスタンバイインスタンスでオペレーティングシステムの更新を実行します。
  - RDS はアップグレードされたスタンバイ DB インスタンスにフェイルオーバーし、それを新しいプライマリ DB インスタンスにします。
  - 最後に、RDS は新しいスタンバイ DB インスタンスでオペレーティングシステムの更新を実行します。

マルチ AZ 配置におけるダウンタイムとは、フェイルオーバーにかかる時間のことです。

システム更新メンテナンスアクションを使用してオペレーティングシステムの更新を適用する

Amazon RDS RPEV インスタンスにオペレーティングシステムの更新を適用するには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用できます。詳細については、「RDS DB インスタンスのオペレーティングシステムの更新」を参照してください。

Example

Linux、macOS、Unix の場合:

ステップ 1: 利用可能な更新を確認する

`describe-pending-maintenance-actions` コマンドを使用して、インスタンスの OS 更新が利用可能かどうかを確認します。

```
aws rds describe-pending-maintenance-actions
```

レスポンスの例:

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-east-1:111122223333:db:my-sqlserver-
instance",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "Description": "New Operating System update is available"
                }
            ]
        }
    ]
}
```

アクションタイプ `system-update` は、OS 更新がインスタンスで使用できることを示します。

ステップ 2: OS 更新を適用する

`apply-pending-maintenance-action` コマンドを使用して更新をスケジュールします。

```
aws rds apply-pending-maintenance-action \
                --resource-identifier arn:aws:rds:us-east-1:111122223333:db:my-
sqlserver-instance \
                --apply-action system-update \
                --opt-in-type immediate
```

`opt-in-type` 入力には次のオプションがあります。

- immediate: 更新をすぐに適用する
- next-maintenance: 次に予定されているメンテナンス期間中に更新を適用する
- undo-opt-in: 事前にスケジュールされた更新をキャンセルする

レスポンスの例:

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:us-east-1:111122223333:db:my-sqlserver-
instance",
        "PendingMaintenanceActionDetails": [
            {
                "Action": "system-update",
                "AutoAppliedAfterDate": "2024-04-10T20:41:01.695000+00:00",
                "ForcedApplyDate": "2024-04-10T20:41:01.694000+00:00",
                "CurrentApplyDate": "2024-04-10T20:41:01.695000+00:00",
                "Description": "New Operating System update is available"
            }
        ]
    }
}
```

OS 更新通知

新しいオプションのオペレーティングシステムパッチが利用可能になったときに通知を受けるには、セキュリティパッチイベントカテゴリの RDS-EVENT-0230 をサブスクライブできます。RDS イベントへのサブスクライブの詳細については、「Amazon RDS イベント通知にサブスクライブする」を参照してください。

考慮事項

OS の更新には、次の考慮事項と制限が適用されます。

- C:\ ドライブに対して行われたオペレーティングシステムのカスタマイズは、オペレーティングシステムの更新時には保持されません。
- 更新を適用する前に、手動スナップショットを作成することをお勧めします。

Amazon RDS Custom for SQL Server の DB インスタンスのアップグレード

Amazon RDS Custom for SQL Server の DB インスタンスをアップグレードするには、新しい DB エンジンバージョンを使用するようにインスタンスを変更します。DB インスタンスのアップグレードに関する一般的な情報については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

トピック

- RDS Custom for SQL Server のアップグレードの概要
- メジャーエンジンバージョンとマイナーエンジンバージョンのアップグレード
- データベース互換性レベル

RDS Custom for SQL Server のアップグレードの概要

Amazon RDS Custom for SQL Server は、メジャーおよびマイナーバージョンのアップグレードをサポートします。マイナーバージョンのアップグレードには、セキュリティパッチ、バグ修正、エンジンの改善が含まれます。Microsoft は、これらの更新を累積更新 (CU) としてリリースします。メジャーバージョンのアップグレードでは、SQL Server 2019 から 2022 へのアップグレードなど、バージョン間での新機能やエンジンの変更を導入します。両方のアップグレードは、すぐに適用することも、スケジュールされたメンテナンスウィンドウ中に適用することもできます。下位互換性の潜在的な問題を防ぐために、アップグレード前に非本番環境でアプリケーションをテストすることをお勧めします。

RDS Custom for SQL Server では、RDS が提供するエンジンバージョン (RPEV) またはカスタムエンジンバージョン (CEV) をアップグレードできます。

- RDS が提供するエンジンバージョン (RPEV) には、オペレーティングシステム (OS) の最新のパッチと SQL Server の累積更新 (CU) が含まれています。
- カスタムエンジンバージョン (CEV) の場合は、2 ステップのプロセスに従う必要があります。まず、ターゲットの SQL Server バージョンで新しい CEV を作成します。「RDS Custom for SQL Server の CEV を作成する準備」を参照してください。このターゲットバージョンは、現在のバージョンと同等かそれ以降である必要があります。新しい CEV を作成したら、この新しいバージョンを使用するようにデータベースインスタンスを変更します。詳細については、「Performing a minor version upgrade for Amazon RDS Custom for SQL Server CEV with Multi-AZ」を参照してください。

実行中の RDS Custom インスタンスに SQL Server 累積更新をインプレースで適用しないでください。特定の SQL Server バージョン (SQL Server 2022 CU16 など) で CEV を作成し、インスタンスに直接新しい累積更新を適用すると、サポートペリメーターから外れ、エラー SP-S3006 が報告されます。CEV を使用して既存の SQL Server インスタンスにパッチを適用するには、必要な累積更新を含む新しい CEV を作成してから、既存のインスタンスを変更して新しい CEV に切り替えます。

マルチ AZ 配置で RDS Custom for SQL Server の DB インスタンスをアップグレードすると、RDS Custom for SQL Server はインスタンスのローリングアップグレードを実行します。このアプローチでは、一度に 1 つのインスタンスをアップグレードすることでダウンタイムを最小限に抑えます。RDS は、ローリングアップグレードを実行するために以下のアクションを実行します。

1. スタンバイ DB インスタンスをアップグレードします。
2. アップグレードされたスタンバイ DB インスタンスにフェイルオーバーし、新しいプライマリ DB インスタンスにします。
3. 新しいスタンバイ DB インスタンスをアップグレードします。

マルチ AZ 配置の DB インスタンスのダウンタイムは、フェイルオーバーにかかる時間です。

RDS Custom for SQL Server の DB インスタンスのアップグレードには、以下の制限が適用されます。

- Custom DB オプショングループおよびパラメータグループはサポートされていません。
- RDS Custom for SQL Server DB インスタンスにアタッチした追加のストレージボリュームは、アップグレード後はアタッチされません。
- CEV では、SQL Server 累積更新のインプレース適用はサポートされておらず、インスタンスがサポートペリメーターから外れます。

メジャーエンジンバージョンとマイナーエンジンバージョンのアップグレード

メジャーエンジンバージョンとマイナーエンジンバージョンの両方のアップグレードは元に戻すことができないため、必ずより新しいバージョンにアップグレードする必要があります。使用可能なターゲットバージョンを特定するには、AWS マネジメントコンソールを使用し、DB インスタンスを変更するときに使用可能なバージョンから選択します。または、CLI コマンド `describe-db-engine-versions` を使用するか、RDS API コマンド DescribeDBEngineVersions を使用します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
    --engine custom-sqlserver-se \
    --engine-version 15.00.4322.2.v1 \
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" \
    --output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
    --engine custom-sqlserver-se ^
    --engine-version 15.00.4322.2.v1 ^
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
    --output table
```

出力には、使用可能なターゲットエンジンのバージョンが表示されます。

```
--------------------------
|DescribeDBEngineVersions|
+------------------------+
|      EngineVersion     |
+------------------------+
|  15.00.4410.1.v1       |
|  15.00.4415.2.v1       |
|  15.00.4430.1.v1       |
|  16.00.4165.4.v1       |
|  16.00.4175.1.v1       |
|  16.00.4185.3.v1       |
+------------------------+
```

ターゲットバージョンを特定したら、AWS マネジメントコンソールを使用し、「RDS Custom for SQL Server DB インスタンスの変更」の手順に従います。または、CLI コマンド `modify-db-instance` を使用するか、RDS API コマンド ModifyDBInstance を使用します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier DB_INSTANCE_IDENTIFIER \
    --engine-version ENGINE_VERSION \
    --allow-major-version-upgrade \
```

```
    --region Region \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier DB_INSTANCE_IDENTIFIER ^
    --engine-version ENGINE_VERSION ^
    --allow-major-version-upgrade ^
    --region Region ^
    --no-apply-immediately
```

> **Note**
>
> メジャーバージョンのアップグレードを実行するには、`--allow-major-version-upgrade` パラメータを含める必要があります。

データベース互換性レベル

Microsoft SQL Server データベース互換性レベルを使用して、いくつかのデータベースの動作を調整し、以前のバージョンの SQL Server を模倣することができます。詳細については、Microsoft ドキュメントの「互換性レベル」を参照してください。

DB インスタンスをアップグレードしても、既存のすべてのデータベースは元の互換性レベルのままとなります。例えば、SQL Server 2019 から SQL Server 2022 にアップグレードした場合、既存のすべてのデータベースの互換性レベルは 150 となります。アップグレード後に作成した新しいデータベースは互換性レベル 160 となります。

ALTER DATABASE コマンドを使用して、データベースの互換性レベルを変更できます。例えば、`customeracct` という名前のデータベースが、SQL Server 2022 との互換性を持つように変更するには、次のコマンドを発行します。

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 160
```

Amazon RDS Custom for SQL Server の DB に関する問題のトラブルシューティング

RDS Custom の責任共有モデルは、OS シェルレベルのアクセスとデータベース管理者アクセスを提供します。RDS Custom は、システムアカウントでリソースを実行する Amazon RDS とは異なり、アカウント内でリソースを実行します。アクセスが増えるほど、責任も重くなります。以降のセクションで、Amazon RDS Custom for SQL Server DB インスタンスに関する問題のトラブルシューティング方法を学ぶことができます。

> **Note**
>
> このセクションでは RDS Custom for SQL Server DB をトラブルシューティングする方法について説明します。RDS Custom for Oracle に関するトラブルシューティングについては、「Amazon RDS Custom for Oracle の DB に関する問題のトラブルシューティング」を参照してください。

トピック

- RDS Custom イベントの表示
- RDS Custom イベントへのサブスクライブ
- RDS Custom for SQL Server の CEV エラーのトラブルシューティング
- RDS Custom for SQL Server DB でサポートされていない構成の修正
- RDS Custom for SQL Server の Storage-Full に関するトラブルシューティング
- RDS Custom for SQL Server の TDE が有効なデータベースの PENDING_RECOVERY 状態のトラブルシューティング

RDS Custom イベントの表示

イベントを表示する手順は、RDS Custom とAmazon RDS DB インスタンスでは同じです。詳細については、「Amazon RDS イベントの表示」を参照してください。

AWS CLIを使用してRDS Custom イベント通知を表示するには、`describe-events`コマンドを使用します。RDS Custom では、いくつかの新しいイベントを導入しています。イベントカテゴリは Amazon RDS の場合と同じです。イベントのリストについては、Amazon RDS イベントカテゴリとイベントメッセージを参照してください。

次の例では、指定した RDS Custom DB インスタンスで発生したイベントの詳細を取得します。

```
aws rds describe-events \
    --source-identifier my-custom-instance \
    --source-type db-instance
```

RDS Custom イベントへのサブスクライブ

イベント受信の手順は、RDS Custom とAmazon RDS DB インスタンスでは同じです。詳細については、「Amazon RDS イベント通知にサブスクライブする」を参照してください。

CLI を使用して RDS Custom イベント通知をサブスクライブするには、`create-event-subscription` コマンドを使用します。以下の必須パラメータを含めます。

- `--subscription-name`
- `--sns-topic-arn`

次の例では、現在のAWSアカウントのRDS Custom DB インスタンスのバックアップおよびリカバリイベントの受信を作成します。通知は、`--sns-topic-arn`で指定されたAmazon Simple Notification Service (Amazon SNS) のトピックに送信されます。

```
aws rds create-event-subscription \
    --subscription-name my-instance-events \
    --source-type db-instance \
    --event-categories '["backup","recovery"]' \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:interesting-events
```

RDS Custom for SQL Server の CEV エラーのトラブルシューティング

CEV を作成しようとすると、エラーが発生する可能性があります。この場合、RDS Custom では RDS-EVENT-0198 イベントメッセージを発行します。RDS イベントの表示の詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

以下の情報を参考にして、考えられる原因に対処します。

メッセージ	トラブルシューティングの推奨事項		
`Custom Engine Version creation expected a Sysprep'd AMI. Retry`	AMI から作成した EC2 インスタンスで Sysprep を実行します。Sysprep を使用して AMI を準備する方		

メッセージ	トラブルシューティングの推奨事項		
creation using a Sysprep'd AMI.	法の詳細については、「Sysprep を使用して標準化された Amazon マシンイメージ (AMI) を作成する」を参照してください。		
EC2 Image permissions for image (AMI_ID) weren't found for customer (Customer_ID). Verify customer (Customer_ID) has valid permissions on the EC2 Image.	作成に使用したアカウントとプロファイルに、選択した AMI の create EC2 Instance と Describe Images に必要な権限があることを確認します。		
Failed to rebuild databases with server collation (collation name) due to missing setup.exe file for SQL Server.	setup ファイルが C:\Program Files\Microsoft SQL Server\nnn\Setup Bootstrap\SQLnnnn\setup.exe にあることを確認してください。		
Image (AMI_ID) doesn't exist in your account (ACCOUNT_ID). Verify (ACCOUNT_ID) is the owner of the EC2 image.	AMI が同じお客様のアカウントに存在することを確認します。		
Image id (AMI_ID) isn't valid. Specify a valid image id, and try again.	AMI の名前が正しくありません。正しい AMI ID が提供されていることを確認します。		

メッセージ	トラブルシューティングの推奨事項		
`Image (AMI_ID) operating system platform isn't supported. Specify a valid image, and try again.`	Windows Server with SQL Server Enterprise、Standard、または Web エディションを備えた、サポートされている AMI を選択します。EC2 Marketplace から、以下のいずれかの使用オペレーションコードの AMI を選択してください。 • RunInstances:0102 - Windows with SQL Server Enterprise • RunInstances:0006 - Windows with SQL Server Standard • RunInstances:0202 - Windows with SQL Server Web		
`SQL Server Web Edition isn't supported for creating a Custom Engine Version using Bring Your Own Media. Specify a valid image, and try again.`	サポートされている SQL Server のエディションを含む AMI を使用してください。詳細については、「RDS Custom for SQL Server CEV のバージョンサポート」を参照してください。		
`The custom engine version can't be the same as the OEV engine version. Specify a valid CEV, and try again.`	Classic RDS Custom for SQL Server エンジンはサポートされていません。例えば、バージョン 15.00.4073.23.v1 などです。サポートされているバージョン番号を使用してください。		
`The custom engine version isn't in an active state. Specify a valid CEV, and try again.`	オペレーションを完了するには、CEV が `AVAILABLE` 状態である必要があります。CEV を `INACTIVE` から `AVAILABLE` に変更します。		

メッセージ	トラブルシューティングの推奨事項		
The custom engine version isn't valid for an upgrade. Specify a valid CEV with an engine version greater or equal to (X), and try again.	ターゲット CEV は無効です。有効なアップグレードパスの要件を確認してください。		
The custom engine version isn't valid. Names can include only lowercase letters (a-z), dashes (-), underscores (_), and periods (.). Specify a valid CEV, and try again.	必要な CEV 命名規則に従ってください。詳細については、「RDS Custom for SQL Server CEV の一般的な要件」を参照してください。		
The custom engine version isn't valid. Specify valid database engine version, and try again. Example: 15.00.4073.23-cev123.	サポートされていない DB エンジンバージョンが提供されました。サポートされている DB エンジンのバージョンを使用してください。		
The expected architecture is (X) for image (AMI_ID), but architecture (Y) was found.	x86_64 アーキテクチャで構築された AMI を使用してください。		
The expected owner of image (AMI_ID) is customer account ID (ACCOUNT_ID), but owner (ACCOUNT_ID) was found.	権限がある AMI から EC2 インスタンスを作成します。EC2 インスタンスで Sysprep を実行して、ベースイメージを作成して保存します。		
The expected platform is (X) for image (AMI_ID), but platform (Y) was found.	Windows プラットフォームで構築された AMI を使用してください。		

メッセージ	トラブルシューティングの推奨事項		
The expected root device type is (X) for image %s, but root device type (Y) was found.	EBS デバイスタイプで AMI を作成します。		
The expected SQL Server edition is (X), but (Y) was found.	Windows Server with SQL Server Enterprise、Standard、または Web エディションを備えた、サポートされている AMI を選択します。EC2 Marketplace から、以下のいずれかの使用オペレーションコードの AMI を選択してください。 • RunInstances:0102 - Windows with SQL Server Enterprise • RunInstances:0006 - Windows with SQL Server Standard • RunInstances:0202 - Windows with SQL Server Web		
The expected state is (X) for image (AMI_ID), but the following state was found: (Y).	AMI が AVAILABLE の状態であることを確認します。		
The provided Windows OS name (X) isn't valid. Make sure the OS is one of the following: (Y).	サポートされている Windows OS を使用してください。		
Unable to find bootstrap log file in path.	ログファイルが C:\Program Files\Microsoft SQL Server\nnn\Setup Bootstrap\Log\Summary.txt にあることを確認してください。		

メッセージ	トラブルシューティングの推奨事項		
`RDS expected a Windows build version greater than or equal to (X), but found version (Y)..`	OS ビルドバージョンが最低でも 14393 の AMI を使用してください。		
`RDS expected a Windows major version greater than or equal to (X), but found version (Y)..`	OS のメジャーバージョンが 10.0 以降の AMI を使用してください。		

RDS Custom for SQL Server DB でサポートされていない構成の修正

共有責任モデルであるため、RDS Custom for SQL Server DB インスタンスを`unsupported-configuration`状態にする設定上の問題は、お客様の責任で解決していただく必要があります。問題がAWSインフラストラクチャであれば、コンソールやAWS CLIを使用して修正できます。OSまたはデータベースの設定に問題がある場合は、ホストにログインして修正できます。

> **Note**
>
> このセクションでは RDS Custom for SQL Server でサポートされていない構成を修正する方法について説明します。RDS Custom for Oracle の詳細については、「RDS Custom for Oracle でサポートされていない構成の修正」を参照してください。

次の表では、サポートペリメーターが送信する通知とイベント、その修正方法について説明します。これらの通知とサポートペリメーターは変更されることがあります。サポート周辺の背景については、「RDS Custom サポート範囲」を参照してください。イベントの説明については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
`SP-S0000`	サポートされていない手動設定	X のため、RDS Custom DB インスタンスのステータス	この問題を解決するには、サポートケースを作成します。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
		は、[サポートされていない設定] に設定されています。	

AWS リソース (インフラストラクチャ)

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1001	EC2 インスタンスのステータス	基盤となる EC2 インスタンス %s が RDS インスタンスを停止せずに停止されたため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、基盤となる EC2 インスタンスを起動し、バイナリボリュームとデータボリュームがアタッチされていることを確認します。RDS インスタンスを停止する場合は、まず基盤となる EC2 インスタンスのステータスが [利用可能] であることを確認してから、RDS	DB インスタンスのステータスを確認するには、コンソールを使用するか、次の AWS CLI コマンドを実行します。 `aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep DBInstanceStatus`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
		コンソールまたは CLI を使用して RDS インスタンスを停止します。	

SP-S1002	EC2 インスタンスのステータス	RDS DB インスタンスのステータスは [STOPPED] に設定されているが、基盤となる EC2 インスタンス %s が起動されたため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、基盤となる EC2 インスタンスを停止します。RDS インスタンスを起動する場合は、コンソールまたは CLI を使用します。	DB インスタンスのステータスを確認するには、次の AWS CLI コマンドを使用します。
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep DBInstanceStatus`
			EC2 インスタンスのステータスは EC2 コンソールで確認できます。
			DB インスタンスを起動するには、コンソールを使用するか、次の AWS CLI コマンドを実行します。
			`aws rds start-db-instance \ --db-instance-identifier db-instance-name`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1003	EC2 インスタンスクラス	EC2 ホストの期待される DB インスタンスクラスと設定されたクラスが一致しないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、DB インスタンスクラスを元のクラスタイプに変更します。	次の CLI コマンドを使用して、期待される DB インスタンスクラスを確認します。 `aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep DBInstanceClass`
SP-S1004	EBS ストレージボリュームにアクセスできない	EC2 インスタンスに関連付けられた元の EBS ストレージボリューム %s に現在アクセスできないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1005	EBS ストレージボリュームがデタッチされている	元の EBS ストレージボリューム「volume-id」がアタッチされていないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、関連付けられている EBS ボリュームを EC2 インスタンスにアタッチします。	EBS ボリュームを再アタッチしたら、次の CLI コマンドを使用して、EBS ボリューム「volume-id」が RDS インスタンスに適切にアタッチされているかどうかを確認します。 `aws ec2 describe-volumes \ --volume-ids volume-id | grep InstanceId`

SP-S1006	EBS ストレージボリュームサイズ	EBS ストレージボリューム「volume-id」の期待される設定と設定された設定が一致しないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。ボリュームサイズが EC2 レベルで元の値 [%s] から手動で変更されました。この問題を解決するには、サポートケースを作成します。	次の CLI コマンドを使用して、EBS ボリューム「volume-id」の詳細と RDS インスタンスの詳細のボリュームサイズを比較します。
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep AllocatedStorage`
			次の CLI コマンドを使用して、実際に割り当てられたボリュームサイズを表示します。
			`aws ec2 describe-volumes \ --volume-ids |grep Size`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S1007	EBS ストレージボリュームの設定	EBS ストレージボリューム「volume-id」の期待される設定と設定された設定が一致しないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、EBS ストレージボリューム設定 [IOPS、スループット、ボリュームタイプ] を、EC2 レベルで [IOPS: %s、スループット: %s、ボリュームタイプ: %s] の元の値に変更します。今後のストレージの変更には、RDS コンソールまたは CLI を使用します。ボリュームサイズも EC2 レベルで元の値 [%s] から手動で変更されました。この問題を解決するには、サポートケースを作成します。	次の CLI コマンドを使用して、EBS ボリューム「volume-id」の詳細と RDS インスタンスの詳細のボリュームタイプを比較します。EBS レベルの値が RDS レベルの値と一致していることを確認します。
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`
			RDS レベルでストレージスループットの想定値を取得するには:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageThroughput`
			RDS レベルでボリューム IOPS の想定値を取得するには:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep Iops`
			EC2 レベルで現在のストレージタイプを取得するには:
			`aws ec2 describe-volumes \`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

			`--volume-ids |grep VolumeType`
			
			EC2 レベルでストレージスループットの現在の値を取得するには:
			
			`aws ec2 describe-volumes \ --volume-ids |grep Throughput`
			
			EC2 レベルでボリューム IOPS の現在の値を取得するには:
			
			`aws ec2 describe-volumes \ --volume-ids |grep Iops`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S1008	EBS ストレージボリュームのサイズと設定	EBS ストレージボリューム「volume-id」の期待される設定と設定された設定が一致しないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、EBS ストレージボリューム設定 [IOPS、スループット、ボリュームタイプ] を、EC2 レベルで [IOPS: %s、スループット: %s、ボリュームタイプ: %s] の元の値に変更します。今後のストレージの変更には、RDS コンソールまたは CLI を使用します。ボリュームサイズも EC2 レベルで元の値 [%s] から手動で変更されました。この問題を解決するには、サポートケースを作成します。	次の CLI コマンドを使用して、EBS ボリューム「volume-id」の詳細と RDS インスタンスの詳細のボリュームタイプを比較します。EBS レベルの値が RDS レベルの値と一致していることを確認します。
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageType`
			RDS レベルでストレージスループットの想定値を取得するには:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep StorageThroughput`
			RDS レベルでボリューム IOPS の想定値を取得するには:
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name | grep Iops`
			EC2 レベルで現在のストレージタイプを取得するには:
			`aws ec2 describe-volumes \`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

			`--volume-ids |grep VolumeType`
			
			EC2 レベルでストレージスループットの現在の値を取得するには:
			
			`aws ec2 describe-volumes \ --volume-ids |grep Throughput`
			
			EC2 レベルでボリューム IOPS の現在の値を取得するには:
			
			`aws ec2 describe-volumes \ --volume-ids |grep Iops`
			
			想定される割り当て済みボリュームサイズを取得するには:
			
			`aws rds describe-db-instances \ --db-instance-identifier *db-instance-name* | grep AllocatedStorage`
			
			割り当てられた実際のボリュームサイズを取得するには:
			
			`aws ec2 describe-volumes \ --volume-ids |grep Size`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1009	SQS アクセス許可	IAM インスタンスプロファイルに Amazon Simple Queue Service (SQS) のアクセス許可がないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、ホストに関連付けられた IAM プロファイルに次のアクセス許可があることを確認します: ["SQS:SendMessage"、"SQS:ReceiveMessage"、"SQS:DeleteMessage"、"SQS:GetQueueUrl"]。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1010	SQS VPC エンドポイント	VPC エンドポイントポリシーが Amazon Simple Queue Service (SQS) オペレーションをブロックしているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、必要な SQS アクションを許可するように VPC エンドポイントポリシーを変更します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1011	イベントバスポリシー	イベントバス arn:aws:events:region-1:123456789012:event-bus/default のリソースベースのポリシーが Amazon CloudWatch の events:PutEvents アクションを拒否しているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、EventBus %s の events:PutEvents アクションを許可するようにリソースベースのポリシーを変更します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1012	CloudWatch VPC のアクセス許可	VPC エンドポイントポリシーに Amazon CloudWatch イベントへのアクセス許可がないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、EventBus arn:aws:events:region-1:123456789012:event-bus/default の events:PutEvents を許可するように VPC エンドポイントポリシーを変更します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1013	サービスコントロールポリシー	AWS Organizations のサービスコントロールポリシーに Amazon CloudWatch イベントへのアクセス許可がないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、EventBus arn:aws:events:region-1:123456789012:event-bus/default の events:PutEvents を許可するようにサービスコントロールポリシーを変更します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1014	IAM インスタンスプロファイル	IAM インスタンスプロファイル %s のアクセス許可が Amazon CloudWatch イベントを拒否しているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、["events:PutEvents"] を 'Allow' に設定し、インスタンスに関連付けられた IAM プロファイルで EventBus arn:aws:events:region-1:123456789012:event-bus/default の events:PutEvents を許可します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1015	IAM インスタンスプロファイル	IAM インスタンスプロファイル %s に Amazon CloudWatch イベントへのアクセス許可がないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、["events:PutEvents"] アクセス許可を含め、インスタンスに関連付けられた IAM プロファイルで EventBus arn:aws:events:region-1:123456789012:event-bus/default の events:PutEvents を許可します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S1016	IAM アクセス許可の境界	IAM インスタンスプロファイル %s には Amazon CloudWatch イベントを拒否するアクセス許可の境界があるため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されます。これを解決するには、IAM インスタンスプロファイルのアクセス許可の境界で EventBus arn:aws:events:region-1:123456789012:event-bus/default の ["events:PutEvents"] を 'Allow' に設定します。	

オペレーティングシステム

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2001	SQL サービスステータス	SQL Server サービスが起動されていないため、RDS Custom DB インスタンスのステータスは [サポートされて	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
		いない設定] に設定されています。これを解決するには、ホストで SQL Server サービスを再起動します。この DB インスタンスがマルチ AZ DB インスタンスで、再起動が失敗した場合、ホストを停止して起動し、フェイルオーバーを開始します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S2002	RDS Custom エージェントステータス	RDS Custom エージェントサービスがインストールされていないか、起動できなかったため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、Windows イベントログを確認して、サービスが起動されない理由を特定し、問題を解決するための適切な手順を実行します。さらにサポートが必要な場合は、サポートケースを作成してください。	ホストにログインし、RDS Custom エージェントが実行されていることを確認します。
			
			エージェントステータスを表示するには、次のコマンドを使用します。
			
			`$name = "RDSCustomAgent"`
			`$service = Get-Service $name`
			`Write-Host $service.Status`
			
			ステータスが Running でなければ、次のコマンドを使ってサービスを開始できます。
			
			`Start-Service $name`
			
			エージェントが起動できない場合は、Windows イベントをチェックして、起動できない理由を確認します。エージェントでは、Windows ユーザーがサービスを起動する必要があります。Windows ユーザーが存在し、サービスを実行する権限があることを確認します。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2003	SSM エージェントステータス	Amazon SSM エージェントサービスにアクセスできないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これをトラブルシューティングするには、`Get-Service AmazonSSMAgent` PowerShell コマンドでサービスステータスを確認するか、`Start-Service AmazonSSMAgent` でサービスを起動します。ssm、ssmmessages、および ec2messages のリージョン別エンドポイントへの HTTPS (ポート 443) アウトバウンドトラフィックが許可されていることを確認します。	詳細については、「SSM Agent のトラブルシューティング」を参照してください。 SSM エンドポイントのトラブルシューティングについては、「Unable to connect to SSM endpoints」および「Use ssm-cli to troubleshoot managed node availability」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2004	RDS Custom エージェントログイン	SQL ログイン "$HOSTNAME/RDSAgent” で予期しない問題が発生したため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。この問題を解決するには、サポートケースを作成します。	
SP-S2005	Timezone	Amazon EC2 インスタンス [%s] のタイムゾーンが変更されたため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、タイムゾーンをインスタンスの作成時に指定された設定に戻します。特定のタイムゾーンでインスタンスを作成する場合は、RDS Custom のドキュメントを参照してください。	Get-Timezone PowerShell コマンドを実行して、タイムゾーンを確認します。 詳細については、「RDS Custom for SQL Server DB インスタンスのローカルタイムゾーン」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2006	高可用性ソフトウェアソリューションバージョン	現在のインスタンスの高可用性ソフトウェアソリューションが想定バージョンと異なるため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。この問題を解決するには、サポートケースを作成します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2007	高可用性ソフトウェアソリューションの設定	高可用性ソフトウェアソリューションの設定がインスタンス %s で予期しない値に変更されているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。この問題を解決するには、EC2 インスタンスを再起動します。EC2 インスタンスを再起動すると、高可用性ソフトウェアソリューションに必要な設定に自動的に更新されます。	

SP-S2008	SQL Server サービス	RDS Custom DB インスタンスが [サポートされていない設定] に設定されています。SQLServer (MSSQLServer) サービスがホストに存在しません。これを解決するには、サポートケースを作成します。	エージェントステータスを表示するには、次のコマンドを使用します。
			`$name = "MSSQLServer"`
			`$service = Get-Service $name`
			`Write-Host $service.Status`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2009	SSL 証明書	RDS Custom DB インスタンスは [サポートされていない設定] に設定されています。これは、RDS の中断を引き起こす自己署名されていない SSL 証明書が原因です。この問題を解決するには、信頼されたルート証明書ストアから自己署名証明書を削除します。	次の PowerShell コマンドを実行して、自己署名証明書を確認します。 `Get-ChildItem cert:\LocalMachine\root -Recurse | Where-Object {$_.Issuer -ne $_.Subject -and $_.Issuer -notlike "*RDSCustomAgentCA*"}` 詳細については、「HTTP Error 403.16 when you try to access a website that's hosted on IIS」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S2010	ルートボリュームのストレージステータス	RDS Custom DB インスタンスは、ルートボリュームストレージが満杯であるため、[サポートされていない設定] に設定されています。この問題を解決するには、ルート EBS ボリューム「volume-id」で少なくとも 500 MiB のストレージスペースを解放するか、ボリュームサイズを増やして EC2 インスタンス「instance-id」の C ドライブのサイズを変更します。EC2 インスタンスを置き換えると、ルートボリュームサイズの変更は保持されません。	ルート (C:) ボリュームで使用可能なストレージを表示するには、次のコマンドを使用します。 `(Get-PSDrive -Name C).Free / 1MB` EBS ルートボリュームの変更の詳細については、「How」を参照してください。

データベース

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S3001	SQL Server 共有メモリプロトコル	SQL Server 共有メモリプロトコルが無効になっている	これを検証するには、[SQL Server Configuration Manager] > [SQL Server Network Configuration] >

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
		ため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、SQL Server Configuration Manager で共有メモリプロトコルを有効にします。	[Protocols for MSSQLSERVER] > [Shared Memory] が有効になっていることを確認します。プロトコルを有効にしてから、SQL Server プロセスを再起動します。
SP-S3002	サービスマスターキー	RDS オートメーションが、新しい SMK 生成の一部としてサービスマスターキー (SMK) のバックアップを取得できないため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。この問題を解決するには、サポートケースを作成します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス
SP-S3003	サービスマスターキー	サービスマスターキー (SMK) に関連するメタデータがないか不完全であるため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。この問題を解決するには、サポートケースを作成します。	

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S3004	DB エンジンのバージョンとエディション	RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これは、SQL Server の想定バージョンとインストールされているバージョンおよびエディションが一致しないためです。SQL Server のエディションの変更は、RDS Custom for SQL Server ではサポートされていません。また、RDS Custom EC2 インスタンスでの SQL Server バージョンの手動変更もサポートされていません。この問題を解決するには、サポートケースを作成します。	次のクエリを実行して、SQL バージョンを取得します。
			`select @@version`
			次の AWS CLI コマンドを実行して、RDS SQL エンジンのバージョンとエディションを取得します。
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep EngineVersion`
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep Engine`
			詳細については、「RDS Custom for SQL Server DB インスタンスの変更」および「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S3005	DB エンジンのエディション	現在の SQL Server エディションが、想定される SQL Server のエディション [%s] と一致しないため、RDS Custom DB インスタンスのステータスが [サポートされていない設定] に設定されています。SQL Server のエディションの変更は、RDS Custom for SQL Server ではサポートされていません。この問題を解決するには、サポートケースを作成します。	次のクエリを実行して、SQL エディションを取得します。
			
			Example
			
			`select @@version`
			
			次の AWS CLI コマンドを実行して、RDS SQL エンジンのエディションを取得します。
			
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep Engine`

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S3006	DB エンジンのバージョン	現在の SQL Server バージョンが、想定される SQL Server のバージョン [%s] と一致しないため、RDS Custom DB インスタンスのステータスが [サポートされていない設定] に設定されています。SQL Server のエディションの変更は、RDS Custom EC2 のインスタンスではサポートされていません。この問題を解決するには、サポートケースを作成します。SQL Server のバージョンを今後変更する場合は、AWS RDS コンソールから、または modify-db-instance CLI コマンドを使用してインスタンスを変更できます。	次のクエリを実行して、SQL バージョンを取得します。
			
			Example
			
			`select @@version`
			
			次の AWS CLI コマンドを実行して、RDS SQL エンジンのバージョンを取得します。
			
			`aws rds describe-db-instances \ --db-instance-identifier db-instance-name |grep EngineVersion`
			
			詳細については、「RDS Custom for SQL Server DB インスタンスの変更」および「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

イベントコード	設定領域	RDS イベントメッセージ	検証プロセス

SP-S3007	データベースファイルの場所	データベースファイルが D:\ ドライブの外部で設定されているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、ROW、LOG、FILESTREAM などのすべてのデータベースファイルが D:\ ドライブに格納されることを確認します。	次のクエリを実行して、デフォルトパスにないデータベースファイルの場所を一覧表示します。
			`USE master;`
			`SELECT physical_name as files_not_in_default_path`
			`FROM sys.master_files`
			`WHERE SUBSTRING(physical_name,1,3)!='D:\';`

SP-S3008	データベース数の制限を超えました	DB インスタンスでのデータベースの総数が上限の 5,000 を超えているため、RDS Custom DB インスタンスのステータスは [サポートされていない設定] に設定されています。これを解決するには、データベースの数をサポートされている上限未満に減らします。	データベースの総数を確認するには、次のコマンドを使用します。
			`SELECT COUNT(name) as databaseCount`
			`FROM sys.databases`
			`WHERE name not in ('tempdb','master','model','msdb','DWDiagnostics','DWConfiguration','DWQueue');`

RDS Custom for SQL Server の **Storage-Full** に関するトラブルシューティング

RDS Custom は、ルート (C:) ボリュームもモニタリングします。ルートボリュームで利用できるディスク容量が 500 MiB 未満の場合、RDS Custom for SQL Server DB インスタンスは `unsupported-configuration` 状態に移行します。`Event SP-S2010` の「RDS Custom for SQL Server DB でサポートされていない構成の修正」を参照してください。

RDS Custom for SQL Server の TDE が有効なデータベースの PENDING_RECOVERY 状態のトラブルシューティング

自動復号で問題が発生した場合、透過的データ暗号化 (TDE) が有効になっている SQL Server データベースは `PENDING_RECOVERY` 状態のままになることがあります。これは通常、復元の完了前にアカウントの RDS Custom マネージド S3 バケットに保存されているソース DB インスタンスサービスマスターキー (SMK) バックアップファイルが削除された場合に、DB インスタンスの復元後に発生します。

自動復号を有効にして TDE が有効なデータベースをオンラインにするには、パスワードでデータベースマスターキー (DMK) を開き、SMK を使用して DMK を暗号化する必要があります。

以下の SQL Server コマンドを参照してください。

```
-- Identify PENDING_RECOVERY TDE databases
USE MASTER;
GO
SELECT name, is_encrypted, state_desc FROM sys.databases;
GO

-- Open DMK using password
OPEN MASTER KEY DECRYPTION BY PASSWORD = '<password>';
GO

-- Encrypt DMK using SMK
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY;
GO

-- Close SMK
CLOSE MASTER KEY;
GO

-- Bring the TDE databases online
ALTER DATABASE <database_name> SET ONLINE;
GO
```

```
-- Verify TDE databases are now in ONLINE state
SELECT name, is_encrypted, state_desc FROM sys.databases;
GO
```

Amazon RDS on AWS Outposts

AWS Outposts での Amazon RDS により、RDS for SQL Server、RDS for MySQL、RDS for PostgreSQL および RDS for Oracle データベースが AWS Outposts Rack Gen1 および Gen2 環境に拡張されます。AWS Outposts により、パブリックな AWS リージョンと同じハードウェアを使用して、AWS のサービス、インフラストラクチャ、オペレーションモデルをオンプレミスに導入できます。出力の RDS を使用すると、オンプレミスで実行する必要があるビジネスアプリケーションの近くにマネージド DB インスタンスをプロビジョニングできます。AWS Outposts については、AWS Outposts ドキュメントおよび AWS Outposts 製品ページを参照してください。

同じ AWS マネジメントコンソール、AWS CLI、およびの RDS API を使用しながら、AWS クラウドで実行する RDS DB インスタンスの場合と同様の方法で、Outposts の DB インスタンスにあるオンプレミス RDS のプロビジョンと管理を行います。Outposts 上の RDS により、データベースのプロビジョニング、オペレーティングシステムとデータベースのパッチ適用、バックアップ、Amazon S3 による長期アーカイブなどのタスクが自動化されます。

出力の RDS は、DB インスタンスの自動バックアップをサポートしています。DB インスタンスをバックアップおよび復元するには、Outpost と AWS リージョン 間のネットワーク接続が必要です。Outpost からのすべての DB スナップショットとトランザクションログはお客様の AWS リージョン に保存されます。お客様の AWS リージョンの DB スナップショットから別の Outpost に DB インスタンスを復元できます。詳細については、「バックアップの概要」を参照してください。

出力の RDS は、DB インスタンスの自動メンテナンスとアップグレードをサポートしています。詳細については、「DB インスタンスのメンテナンス」を参照してください。

RDS on Outposts では、AWS KMS key を使用してDB インスタンスと DB スナップショットの保管時に、 をによる暗号化を使用します。保存時の暗号化の詳細については、「Amazon RDS リソースの暗号化」を参照してください。

デフォルトでは、Outposts サブネットの EC2 インスタンスは Amazon Route 53 DNS サービスを使用してドメイン名を IP アドレスを解決できます。Outpost と AWS リージョン 間のパスレイテンシーによっては、Route 53 での DNS 解決時間が長くなる場合があります。このような場合、オンプレミス環境でローカルにインストールされた DNS サーバーを使用できます。詳細については、AWS Outposts ユーザーガイドの DNS を参照してください。

AWS リージョン へのネットワーク接続が利用できない場合、DB インスタンスは引き続きローカルで実行されます。ローカル DNS サーバーをセカンダリサーバーとして設定することで、DNS 名前解決を使用して DB インスタンスに引き続きアクセスすることができます。ただし、新しい DB イン

スタンスを作成したり、既存の DB インスタンスを変更したりすることはできません。接続がない場合、自動バックアップは行われません。DB インスタンスに障害が発生した場合、接続が復元されるまで DB インスタンスは自動的に置き換えられません。できる限り早くネットワーク接続を復元することをお勧めします。

トピック

- AWS Outposts での Amazon RDS の前提条件
- AWS Outposts での Amazon RDS による Amazon RDS 機能のサポート
- AWS Outposts の Amazon RDS でサポートされている DB インスタンスクラス
- AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス
- AWS Outposts 上での Amazon RDS のマルチ AZ 配置の使用
- AWS Outposts の Amazon RDS での DB インスタンスの作成
- Amazon RDS on AWS Outposts でのリードレプリカの作成
- AWS Outposts 上の Amazon RDS での DB インスタンスの復元に関する考慮事項

AWS Outposts での Amazon RDS の前提条件

AWS Outposts 上で Amazon RDS を使用するための前提条件は次のとおりです。

- オンプレミスデータセンターに AWS Outposts をインストールします。詳細については、AWS Outposts サーバーインストールガイドの Installing an AWS Outposts server を参照してください。
- 出力の RDS で使用可能なサブネットが少なくとも 1 つあることを確認します。同じサブネットを他のワークロードに使用できます。
- Outpost と AWS リージョン間に信頼できるネットワーク接続が必要です。

AWS Outposts での Amazon RDS による Amazon RDS 機能のサポート

次の表では、Amazon RDS on AWS Outposts でサポートされる Amazon RDS 機能について説明します。

機能	サポート対象	コメント	詳細情報

DB インスタンスのプロビジョニング	はい	DB インスタンスは、RDS for SQL Server、RDS for MySQL、RDS for PostgreSQL、および RDS for Oracle DB エンジンについてのみ作成できます。以下バージョンがサポートされています。
		• Microsoft SQL Server:
		  • 16.00.4085.2.v1 以降の 2022 バージョン
		  • 15.00.4043.16.v1 以降の 2019 バージョン
		  • 14.00.3294.2.v1 以降の 2017 バージョン
		  • 13.00.5820.21.v1 以降の 2016 バージョン
		• MySQL 8.0 および 8.4 バージョン
		• すべての PostgreSQL 16、15、14、13 バージョンと PostgreSQL バージョン 12.5 以降の PostgreSQL 12 バージョン
			AWS Outposts の Amazon RDS での DB インスタンスの作成

機能	サポート対象	コメント	詳細情報
		• すべての Oracle バージョン	
Microsoft SQL Server Management Studio を使用して Microsoft SQL Server DB インスタンスに接続する	はい	一部の TLS バージョンと暗号化暗号は安全でない可能性があります。無効にするには、SQL Server のセキュリティプロトコルおよび暗号の設定の手順に従います。	SQL Server DB インスタンスへの接続
マスターユーザーパスワードの変更	はい	なし	Amazon RDS DB インスタンスを変更する
DB インスタンスの名前を変更する	はい	なし	Amazon RDS DB インスタンスを変更する
DB インスタンスを再起動する	はい	なし	DB インスタンスの再起動
DB インスタンスの停止	はい	なし	一時的に Amazon RDS DB インスタンスを停止する
DB インスタンスのスタート	はい	なし	以前に停止した Amazon RDS DB インスタンスを開始する

機能	サポート対象	コメント	詳細情報
マルチ AZ 配置	はい	マルチ AZ 配置は MySQL、PostgreSQL、および Oracle DB インスタンスでサポートされています。 マルチ AZ 配置はダイレクト VPC ルーティング (DVR) をサポートしていません。	AWS Outposts の Amazon RDS での DB インスタンスの作成 Amazon RDS でのマルチ AZ 配置の設定と管理
DB パラメータグループ	はい	なし	Amazon RDS のパラメータグループ
リードレプリカ	はい	リードレプリカは MySQL、PostgreSQL、および Oracle DB インスタンスでサポートされています。 リードレプリカはダイレクト VPC ルーティング (DVR) をサポートしていません。	Amazon RDS on AWS Outposts でのリードレプリカの作成
保管中の暗号化	はい	出力の RDS は、暗号化されていない DB インスタンスをサポートしていません。	Amazon RDS リソースの暗号化
AWS Identity and Access Management (IAM) データベース認証	いいえ	なし	MariaDB、MySQL、および PostgreSQL の IAM データベース認証

機能	サポート対象	コメント	詳細情報
IAM ロールと DB クラスターの関連付け	いいえ	なし	add-role-to-db-instance AWS CLI コマンド AddRoleToDBInstance RDS API オペレーション
Kerberos 認証	はい	なし	Kerberos 認証
Amazon RDS リソースのタグ付け	はい	なし	Amazon RDS リソースのタグ付け
オプショングループ	はい	次の RDS for Oracle オプションはサポートされていません。 • Amazon EFS の統合 • 「Amazon S3 統合」	オプショングループを使用する
メンテナンスウィンドウの変更	はい	なし	DB インスタンスのメンテナンス
自動マイナーバージョンアップグレード	はい	なし	マイナーエンジンバージョンの自動アップグレード
バックアップウィンドウの変更	はい	なし	バックアップの概要 Amazon RDS DB インスタンスを変更する
DB インスタンスクラスを変更する	はい	なし	Amazon RDS DB インスタンスを変更する

機能	サポート対象	コメント	詳細情報
割り当てられたストレージの変更	はい	なし	Amazon RDS DB インスタンスを変更する
ストレージのオートスケーリング	はい	なし	Amazon RDS ストレージの自動スケーリングによる容量の自動管理
手動および自動 DB インスタンススナップショット	はい	自動バックアップやマニュアルスナップショットは AWS リージョン に保存することができます。または、Amazon S3 on Outposts と Amazon EBS ローカルスナップショットが必要な Outpost にローカルに保存することもできます。 ローカルバックアップは MySQL、PostgreSQL、および Oracle DB インスタンスでサポートされています。 バックアップを Outpost に保存するには、Outposts に Amazon S3 が設定されていることを確認してください。 ローカルバックアップは、マルチ AZ インスタンスデプロイではサポートされていません。	AWS Outposts の Amazon RDS での DB インスタンスの作成 Amazon S3 on Outposts Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成

機能	サポート対象	コメント	詳細情報
DB スナップショットからの復元	はい	復元した DB インスタンスの自動バックアップとマニュアルスナップショットは、親 AWS リージョン または Outspot にローカル保存できます。	AWS Outposts 上の Amazon RDS での DB インスタンスの復元に関する考慮事項 DB インスタンスへの復元
Amazon S3 からの DB インスタンスの復元	いいえ	なし	Amazon RDS for MySQL DB インスタンスへのバックアップの復元
Amazon S3 へのスナップショットデータのエクスポート	いいえ	なし	Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート
ポイントインタイムリカバリ	はい	復元した DB インスタンスの自動バックアップとマニュアルスナップショットを親 AWS リージョン または Outpost にローカル保存できますが、例外が1つあります。	AWS Outposts 上の Amazon RDS での DB インスタンスの復元に関する考慮事項 Amazon RDS の DB インスタンスを特定の時点に復元する
拡張モニタリング	はい	なし	拡張モニタリングを使用した OS メトリクスのモニタリング
Amazon CloudWatch のモニタリング	はい	AWS リージョン のデータベースで使用可能なメトリクスのセットと同じセットを表示できます。	Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング

機能	サポート対象	コメント	詳細情報
CloudWatch Logs へのデータベースエンジンログの発行	はい	なし	Amazon CloudWatch Logs へのデータベースログの発行
イベントの通知	はい	なし	Amazon RDS イベント通知の操作
Amazon RDS Performance Insights	いいえ	なし	Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング
データベースログの表示またはダウンロード	いいえ	Outpost の RDS は、コンソールを使用したデータベースログの表示や、AWS CLIまたは RDS API を使用したデータベースログの取得をサポートしていません。 出力の RDS は、コンソールを使用したデータベースログのダウンロードや、AWS CLI または RDS API を使用したデータベースログのダウンロードをサポートしていません。	Amazon RDS ログファイルのモニタリング
Amazon RDS プロキシ	いいえ	なし	Amazon RDS Proxy
Amazon RDS for MySQL のストアドプロシージャ	はい	なし	RDS for MySQL ストアドプロシージャリファレンス

機能	サポート対象	コメント	詳細情報
RDS for MySQL の外部データベースを使用したレプリケーション	いいえ	なし	外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定
Amazon RDS for Microsoft SQL Server のネイティブバックアップおよびリストア	はい	なし	ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート

AWS Outposts の Amazon RDS でサポートされている DB インスタンスクラス

AWS Outposts の Amazon RDS では、次の DB インスタンスクラスがサポートされています。

- 汎用 DB インスタンスクラス
  - db.m7i.48xlarge
  - db.m7i.24xlarge
  - db.m7i.16xlarge
  - db.m7i.12xlarge
  - db.m7i.8xlarge
  - db.m7i.4xlarge
  - db.m7i.2xlarge
  - db.m7i.xlarge
  - db.m7i.large
  - db.m5.24xlarge
  - db.m5.16xlarge
  - db.m5.12xlarge

- db.m5.8xlarge
- db.m5.4xlarge
- db.m5.2xlarge
- db.m5.xlarge
- db.m5.large

- メモリ最適化 DB インスタンスクラス
  - db.r7i.48xlarge
  - db.r7i.24xlarge
  - db.r7i.16xlarge
  - db.r7i.12xlarge
  - db.r7i.8xlarge
  - db.r7i.4xlarge
  - db.r7i.2xlarge
  - db.r7i.xlarge
  - db.r7i.large
  - db.r5.24xlarge
  - db.r5.16xlarge
  - db.r5.12xlarge
  - db.r5.8xlarge
  - db.r5.4xlarge
  - db.r5.2xlarge
  - db.r5.xlarge
  - db.r5.large

Outpost の設定方法によっては、これらのクラスをすべて利用できない場合があります。例えば、Outpost の db.r5 クラスを購入していない場合は、Outposts で RDS と使用することはできません。

汎用 SSD ストレージのみが、Outposts DB インスタンスの RDS でサポートされています。DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

Amazon RDS が DB インスタンスのメンテナンスと復旧を管理するためには、Outpost のアクティブ容量が必要です。本稼働環境の DB インスタンスクラスごとに、N+1 の EC2 インスタンスを設定することをお勧めします。Outposts の RDS では、これらの EC2 インスタンスの追加容量を、メンテナンスと修復のオペレーションに使用できます。例えば、本稼働環境に 3 つの db.m5.large と 5 db.r5.xlarge DB インスタンスクラスがある場合、少なくとも 4 つの m5.large EC2 インスタンスと 6 つの r5.xlarge EC2 インスタンスを使用することをお勧めします。詳細については、AWS Outposts ユーザーガイドの AWS Outposts での回復力 を参照してください。

AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス

Amazon RDS on AWS Outposts は、オンプレミスネットワークに関して提供された情報を使用して、アドレスプールを作成します。このプールは、カスタマー所有の IP アドレスプール (CoIP プール) と呼ばれます。顧客所有の IP アドレス (CoIP) は、オンプレミスネットワークを介して Outpost サブネット内のリソースへのローカル接続または外部接続を提供します。CoIP の詳細については、AWS Outposts ユーザーガイドの顧客所有の IP アドレスを参照してください 。

各 出力の RDS DB インスタンスには、virtual private cloud (VPC) 内にトラフィック用のプライベート IP アドレスがあります。このプライベート IP アドレスにはパブリックアクセスができません。パブリックオプションを使用すると、DB インスタンスがプライベート IP アドレスだけでなく、パブリック IP アドレスも保持するかどうかを設定できます。接続にパブリック IP アドレスを使用すると、インターネット経由でルーティングされ、場合によってはレイテンシーが高くなることがあります。

出力の RDS では、これらのプライベート IP アドレスとパブリック IP アドレスを使用する代わりに、RDS on Outposts は、サブネットを通じて DB インスタンスの CoIP の使用をサポートします。Outposts 上の RDS で、DB インスタンスの CoIP を使用すると、DB インスタンスエンドポイントを使用して DB インスタンスに接続できます。Outposts の RDS は、VPC の内部と外部に対するすべての接続に CoIP を自動的に使用します。

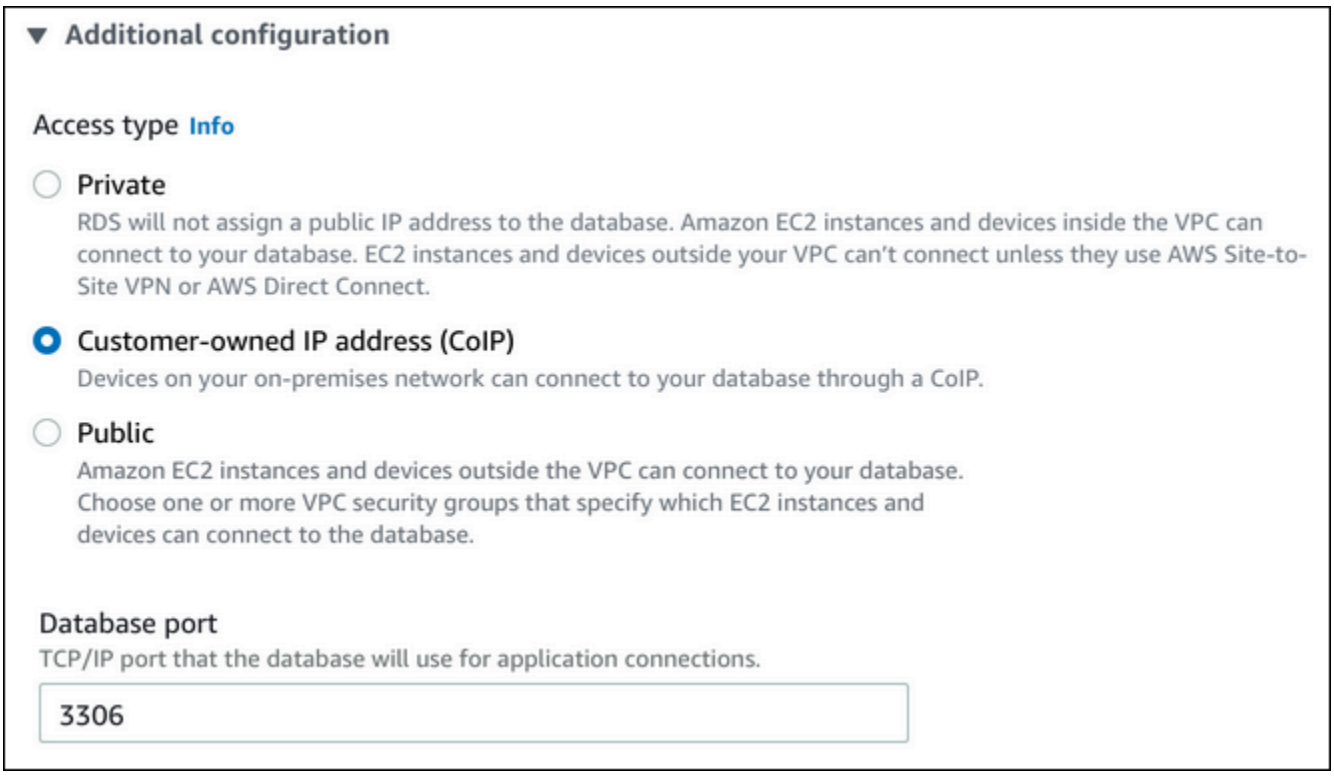
CoIP は、出力の RDS DB インスタンスに対して次の利点を提供します。

- 接続レイテンシーの短縮
- 強化されたセキュリティ

CoIP を使用する

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、 RDS on Outposts DB インスタンスの CoIP を有効または無効にできます。

- AWS マネジメントコンソール では、[Access type] (アクセスタイプ) の [Customer-owned IP address (CoIP)] (顧客所有の IP アドレス (CoIP)) 設定を使用して、CoIP を使用します。他の設定のいずれかを選択して無効にします。



- AWS CLI で、`--enable-customer-owned-ip | --no-enable-customer-owned-ip` オプションを使用します。
- RDS API で、`EnableCustomerOwnedIp` パラメータを使用します。

次のいずれかのアクションを実行すると、CoIP を有効または無効にできます。

- DB インスタンスを作成する

  詳細については、「AWS Outposts の Amazon RDS での DB インスタンスの作成」を参照してください。
- DB インスタンスの変更

  詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- リードレプリカの作成

  詳細については、「Amazon RDS on AWS Outposts でのリードレプリカの作成」を参照してください。
- スナップショットからの DB インスタンスの復元

  詳細については、「DB インスタンスへの復元」を参照してください。

- 特定の時点への DB インスタンスの復元

  詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

> **Note**
>
> DB インスタンスの CoIP を有効にしても、Amazon RDS が DB インスタンスに CoIP を割り当てられない場合があります。このような場合、DB インスタンスのステータスは、互換性のないネットワークに変更されます。DB インスタンスのステータスの詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

制限事項

出力の RDS DB インスタンスの CoIP サポートには、次の制限が適用されます。

- DB インスタンスで CoIP を使用している場合は、その DB インスタンスでパブリックアクセシビリティが無効になっていることを確認してください。
- VPC セキュリティグループのインバウンドルールに CoIP アドレスの範囲 (CIDR ブロック) が含まれていることを確認します。セキュリティグループの設定の詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。
- CoIP プールから DB インスタンスに CoIP を割り当てることはできません。DB インスタンスに CoIP を使用するとき、Amazon RDS は CoIP プールの CoIP を DB インスタンスに自動的に割り当てます。
- Outpost リソース (所有者) を所有する AWS アカウント アカウントを使用するか、同じ組織の他の AWS アカウント アカウント (コンシューマー) と次のリソースを共有する必要があります。
  - Outpost
  - DB インスタンスの VPC のローカルゲートウェイ (LGW) のルートテーブル
  - LGW ルートテーブルの CoIP プール

  詳細については、AWS Outposts ユーザーガイドの共有 AWS Outposts リソースを使用するを参照してください。

AWS Outposts 上での Amazon RDS のマルチ AZ 配置の使用

マルチ AZ 配置の場合、Amazon RDS はプライマリ DB インスタンスを 1 つのAWS Outpost に作成します。RDS は別の Outpost 上にあるスタンバイ DB インスタンスにデータを同期的にレプリケートします。

AWS Outposts のマルチ AZ 配置は、AWS リージョン のマルチ AZ 配置のように機能しますが、以下の違いがあります。

- 2 つ以上の Outposts 間のローカル接続が必要です。
- 顧客所有の IP (CoIP) プールが必要です。詳細については、「AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス」を参照してください。
- レプリケーションは、ローカルネットワークで実行されます。

AWS Outposts でのマルチ AZ は、RDS on Outposts 上で MySQL、PostgreSQL、および Oracle のすべてのサポートされているバージョンで使用できます。ローカルバックアップは、マルチ AZ 配置ではサポートされていません。詳細については、「AWS Outposts の Amazon RDS での DB インスタンスの作成」を参照してください。

責任共有モデルの使用

AWS は高可用性を実現する DB インスタンスを提供するために商業的に合理的な努力を行っていますが、その可用性では責任共有モデルが使用されます。RDS on Outposts の DB インスタンスをフェイルオーバーおよび修復を行う機能には、各 Outposts が AWS リージョン に接続されている必要があります。

また、RDS on Outposts では、プライマリ DB インスタンスをホストしている Outpost と、同期レプリケーションのためにスタンバイ DB インスタンスをホストしている Outpost との間の接続も必要です。この接続に影響があると、RDS on Outposts がフェイルオーバーを実行できなくなる可能性があります。

同期的データレプリケーションの結果、スタンダード DB インスタンスのデプロイのレイテンシーが高くなる可能性があります。プライマリ DB インスタンスをホストしている Outpost とスタンバイ DB インスタンスをホストしている Outpost の間の接続の帯域幅とレイテンシーは、直接レイテンシーに影響します。詳細については、「前提条件」を参照してください。

可用性の向上

可用性を改善するには、次のアクションをお勧めします。

- 基になるホストの問題が発生した場合に回復とフェイルオーバーを可能にするために、ミッションクリティカルなアプリケーション用に十分な追加容量を割り当てます。これは、DB サブネットグループ内のサブネットを含むすべての Outposts に適用されます。詳細については、「AWS Outposts の耐障害性」を参照してください。
- Outposts に冗長ネットワーク接続を提供します。
- Outposts を 3 つ以上使用してください。3 つ以上の Outposts を使用すると、Amazon RDS は DB インスタンスを回復できます。現在の Outpost に障害が発生した場合、RDS は、DB インスタンスを別の Outpost に移動させることで回復します。
- Outpost にデュアル電源と冗長ネットワーク接続を提供します。

ローカルネットワークに次のことをお勧めします。

- プライマリ DB インスタンスをホストしている Outpost とスタンバイ DB インスタンスをホストしている Outpost の間のラウンドトリップ時間 (RTT) レイテンシーは、書き込みレイテンシーに直接影響します。AWS Outposts 間の RTT レイテンシーを低い 1 桁ミリ秒に保ちます。5 ミリ秒以下をお勧めしますが、要件は異なる場合があります。

  ネットワークレイテンシーへの正味の影響は、`WriteLatency` の Amazon CloudWatch メトリクスで確認できます。詳細については、「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。
- Outposts 間の接続の可用性は、DB インスタンスの全体的な可用性に影響します。Outposts 間に冗長なネットワーク接続があります。

前提条件

RDS on Outposts でのマルチ AZ 配置には、次の前提条件があります。

- ローカル接続を介して接続され、AWS リージョン に異なるアベイラビリティーゾーンに接続されている Outposts が少なくとも 2 つある。
- DB サブネットグループに以下が含まれていることを確認します。
  - 任意の AWS リージョン にある 2 つ以上のアベイラビリティーゾーンにある 2 つ以上のサブネット。
  - Outposts 内のみのサブネット
  - 同じ仮想プライベートクラウド (VPC) 内の 2 つ以上の Outposts 内の 2 つ以上のサブネット。

- DB インスタンスの VPC をすべてのローカルゲートウェイルートテーブルに関連付けます。レプリケーションは、Outposts のローカルゲートウェイを使用してローカルネットワーク上で実行されるため、この関連付けが必要です。

  例えば、VPC では、Outpost-A にサブネット A、Outpost-B にサブネット B が含まれているとします。Outpost-A は LocalGateway-A (LGW-A)、Outpost-B は LocalGateway-B (LGW-B) を使用します。LGW-A には RouteTable-A があり、LGW-B には RouteTable-B があります。レプリケーショントラフィックには、RouteTable-A と RouteTable-B の両方を使用します。これを行うには、VPC を RouteTable-A と RouteTable-B の両方に関連付けます。

  関連付けの作成方法の詳細については、Amazon EC2 create-local-gateway-route-table-vpc-association AWS CLI コマンドを参照してください。
- Outposts が顧客所有の IP (CoIP) ルーティングを使用していることを確認します。また、各ルートテーブルには、少なくとも 1 つのアドレスプールが必要です。Amazon RDS は、データの同期のために、プライマリ DB インスタンスとスタンバイ DB インスタンスに対してそれぞれ追加の IP アドレスを割り当てます。
- RDS DB インスタンスを所有する AWS アカウント が、ローカルゲートウェイのルートテーブルおよび CoIP プールを所有していることを確認します。または、ローカルゲートウェイルートテーブルおよび CoIP プールへのアクセスを持つ Resource Access Manager 共有の一部であることを確認します。
- CoIP プール内の IP アドレスが、1 つの Outpost ローカルゲートウェイから別のゲートウェイにルーティングできることを確認します。
- VPC の CIDR ブロック (10.0.0.0/4 など) と CoIP プール CIDR ブロックにクラス E (240.0.0.0/4) からの IP アドレスが含まれていないことを確認します。RDS はこれらの IP アドレスを内部的に使用します。
- アウトバウンドおよび関連するインバウンドトラフィックが正しく設定されていることを確認します。

  RDS on Outposts は、プライマリ DB インスタンスとスタンバイ DB インスタンスの間に仮想プライベートネットワーク (VPN) 接続を確立します。これを正しく機能させるには、インターネットセキュリティアソシエーションおよびキー管理プロトコル (ISAKMP) のアウトバウンドおよび関連するインバウンドトラフィックがローカルネットワークで許可されている必要があります。UDP ポート 4500 を使用して、ユーザーデータグラムプロトコル (UDP) ポート 500 と IP セキュリティ (IPsec) ネットワークアドレス変換トラバーサル (NAT-T) を使用します。

CoIP の詳細については、このガイドの AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス と AWS Outposts ユーザーガイドの顧客所有の IP アドレスを参照してください。

Amazon EC2 許可に対する API オペレーションの使用

AWS Outposts で DB インスタンスに CoIP を使用するかどうかにかかわらず、RDS には CoIP プールリソースへのアクセスが必要です。RDS は、マルチ AZ 配置の代わりに、CoIP の次の EC2 許可 API オペレーションを呼び出すことができます。

- `CreateCoipPoolPermission` — RDS on Outposts のマルチ AZ DB インスタンスを作成する場合
- `DeleteCoipPoolPermission` — RDS on Outposts のマルチ AZ DB インスタンスを削除する場合

これらの API オペレーションは、許可で指定された CoIP プールから Elastic IP アドレスを割り当てる許可を内部 RDS アカウントに付与、または削除します。これらの IP アドレスは、`DescribeCoipPoolUsage` API オペレーションを使用して表示できます。CoIP の詳細については、AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス と AWS Outposts ユーザーガイドの顧客所有の IP アドレスを参照してください 。

RDS は、マルチ AZ 配置の代わりに、ローカルゲートウェイルートテーブルの次の EC2 許可 API オペレーションを呼び出すことができます。

- `CreateLocalGatewayRouteTablePermission` — RDS on Outposts のマルチ AZ DB インスタンスを作成する場合
- `DeleteLocalGatewayRouteTablePermission` — RDS on Outposts のマルチ AZ DB インスタンスを削除する場合

これらの API オペレーションは、内部 RDS アカウントに内部 RDS VPC をローカルゲートウェイルートテーブルに関連付ける許可を付与または削除します。`DescribeLocalGatewayRouteTableVpcAssociations` API オペレーションを使用して、これらのルートテーブルと VPC の関連付けを表示できます。

AWS Outposts の Amazon RDS での DB インスタンスの作成

AWS Outposts の Amazon RDS での DB インスタンスの作成は、AWS クラウドでの Amazon RDS DB インスタンスの作成に類似しています。ただし、Outpost に関連付けられている DB サブネットグループを必ず指定してください。

Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) は、AWS リージョン のすべてのアベイラビリティーゾーンをまたぐことができます。Outpost サブネットを追加することで、AWS リージョン 内の任意の VPC を Outpost に拡張できます。Outpost サブネットを VPC に追加するには、サブネットを作成するときに Outpost の Amazon リソースネーム (ARN) を指定します。

出力の RDS DB インスタンスを作成する前に、Outpost に関連付けられた 1 つのサブネットを含む DB サブネットグループを作成できます。出力の RDS DB インスタンスを作成するときに、この DB サブネットグループを指定します。DB インスタンスを作成するときに、新しい DB サブネットグループを作成することもできます。

AWS Outposts の設定については、AWS Outposts ユーザーガイドを参照してください。

コンソール

DB サブネットグループを作成する

Outpost に関連付けられた 1 つのサブネットを持つ DB サブネットグループを作成します。

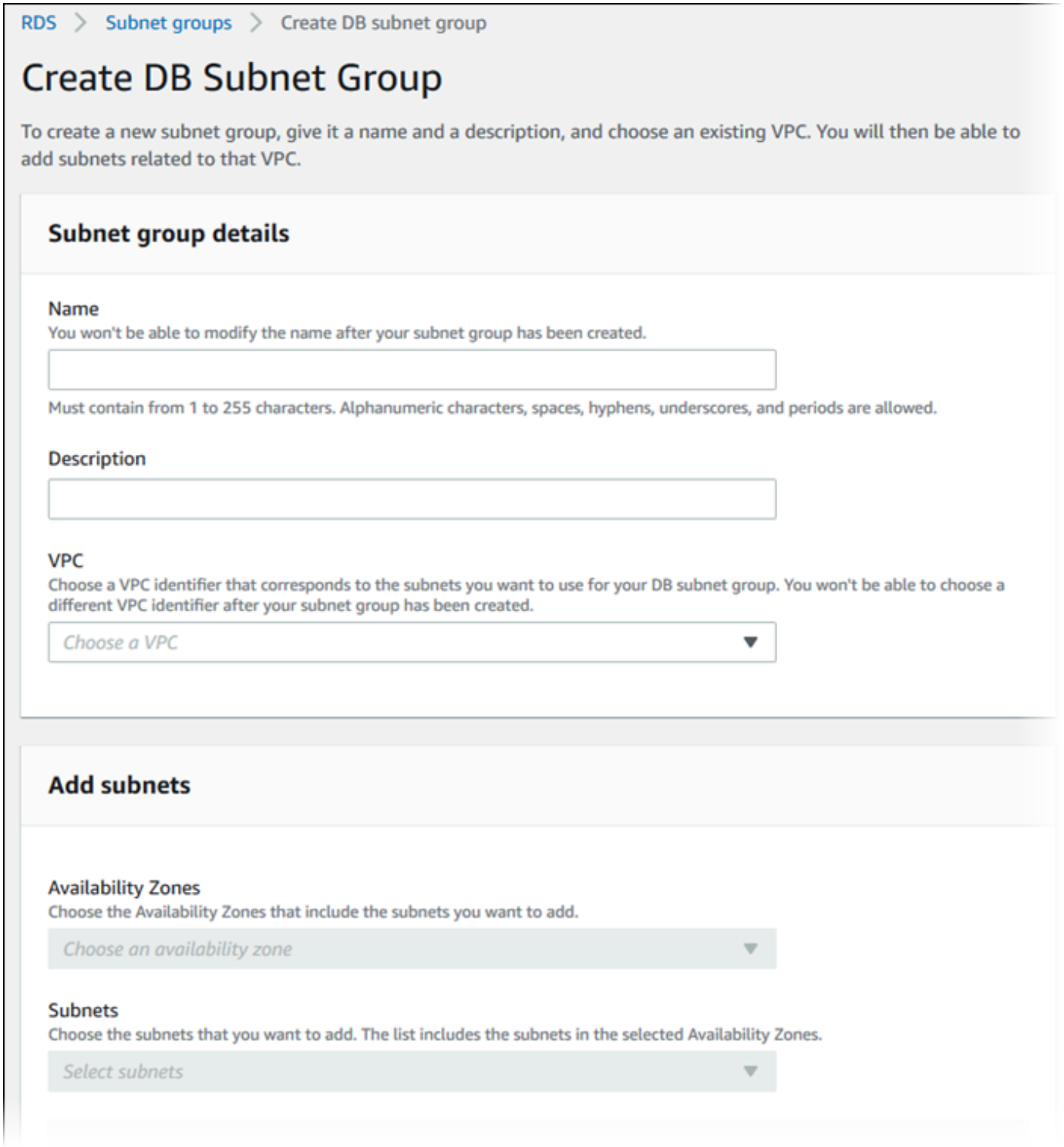
DB インスタンスを作成するときに、Outpost 用の新しい DB サブネットグループを作成することもできます。その場合、この手順をスキップします。

> **Note**
>
> AWS クラウド の DB サブネットグループを作成するには、少なくとも 2 つのサブネットを指定します。

Outpost の DB サブネットグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB サブネットグループを作成する AWS リージョンを選択します。
3. [サブネットグループ] を選択してから、[DB サブネットグループの作成] を選択します。

[DB サブネットグループの作成] ページが表示されます。



4. 「Name (名前)」では、DB サブネットグループの名前を選択します。
5. 「Description (説明)」 では、 DB サブネットグループの説明を選択します。
6. [VPC] では、 DB サブネットグループを作成する VPCを選択します。

7. [アベイラビリティーゾーン] で、Outpost のアベイラビリティーゾーンを選択します。
8. [サブネット] で、出力の RDS 用にサブネットを選択します。
9. [作成] を選択して、DB サブネットグループを作成します。

Outspot の RDS DB インスタンスページの作成

DB インスタンスを作成し、DB インスタンスの Outpost を選択します。

コンソールを使用して 出力の RDS DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する Outpost がアタッチされる AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。

   AWS マネジメントコンソール では、設定した利用可能な Outposts が検出され、[オンプレミス] オプションが [データベースの場所] セクションに表示されます。

   > **Note**
   >
   > Outposts を設定していない場合は、[データベースの場所] セクションが表示されないか、[オンプレミス作成方法の選択] セクションで [出力の RDS] オプションを使用できません。

5. データベース場所では、オンプレミスを選択します。
6. 「On-premises creation method (オンプレミスの作成方法」]では、「RDS on Outposts (Outposts 上の RDS)」を選択します。
7. Outposts 接続性の設定を指定します。この設定は、DB インスタンスの DB サブネットグループを含む VPC を使用する Outpost 用です。VPC は、Amazon VPC サービスに基づいている必要があります。

   a. 「Virtual Private Cloud (VPC)」では、 DB インスタンスの DB サブネットグループを含む VPC を選択します。
   b. VPC セキュリティグループでは、DB インスタンスの Amazon VPC セキュリティグループを選択します。

c. [DB subnet group] (DB サブネットグループ) では、DB インスタンスの DB サブネットグループを選択します。

Outpost に関連付けられている既存の DB サブネットグループを選択できます。例えば、DB サブネットグループを作成する の手順を実行した場合などです。

Outpost の新しい DB サブネットグループを作成することもできます。

8. マルチ AZ 配置の場合、[Create a standby instance (recommended for production usage)] (スタンバイインスタンスの作成 (本番環境での使用に推奨)) をクリックして、別の Outpost にスタンバイ DB インスタンスを作成します。

> **Note**
>
> このオプションは、Microsoft SQL Server では使用できません。
> マルチ AZ 配置を作成する場合は、バックアップを Outpost に保存できません。

9. 「バックアップ」 で、次の作業を行います。

a. 「Backup target (バックアップの対象)」では、以下のいずれかを選択します。

- AWS クラウド は、自動バックアップとマニュアルスナップショットを親 AWS リージョン に保存します。
- Outposts (オンプレミス) で、ローカルバックアップを作成します。

> **Note**
>
> バックアップを Outpost に保存するには、Outpost に Simple Storage Service (Amazon S3) 機能が必要です。詳細については、Amazon S3 on Outpostsを参照してください。
> ローカルバックアップは、マルチ AZ 配置やリードレプリカではサポートされていません。

b. 自動バックアップの有効化を選択して、DB インスタンスのポイントインタイムのスナップショットを作成します。

自動バックアップを有効にする場合は、バックアップ保持期間とバックアップウィンドウの値を選択するか、デフォルト値のままにしておくことができます。

10. 必要に応じて、他の DB インスタンスの設定を指定します。

DB インスタンスを作成する際の各設定の詳細については、「DB インスタンスの設定」を参照してください。

11. [データベースの作成] を選択します。

「Databases (データベース)」 ページが表示されます。DBインスタンスの作成中であることを示すバナーが表示され、「認証情報の詳細を表示」ボタンが表示されます。

DB インスタンスの詳細の表示

DB インスタンスの作成後、認証情報やその他の詳細を表示することができます。

インスタンスの詳細を表示するには

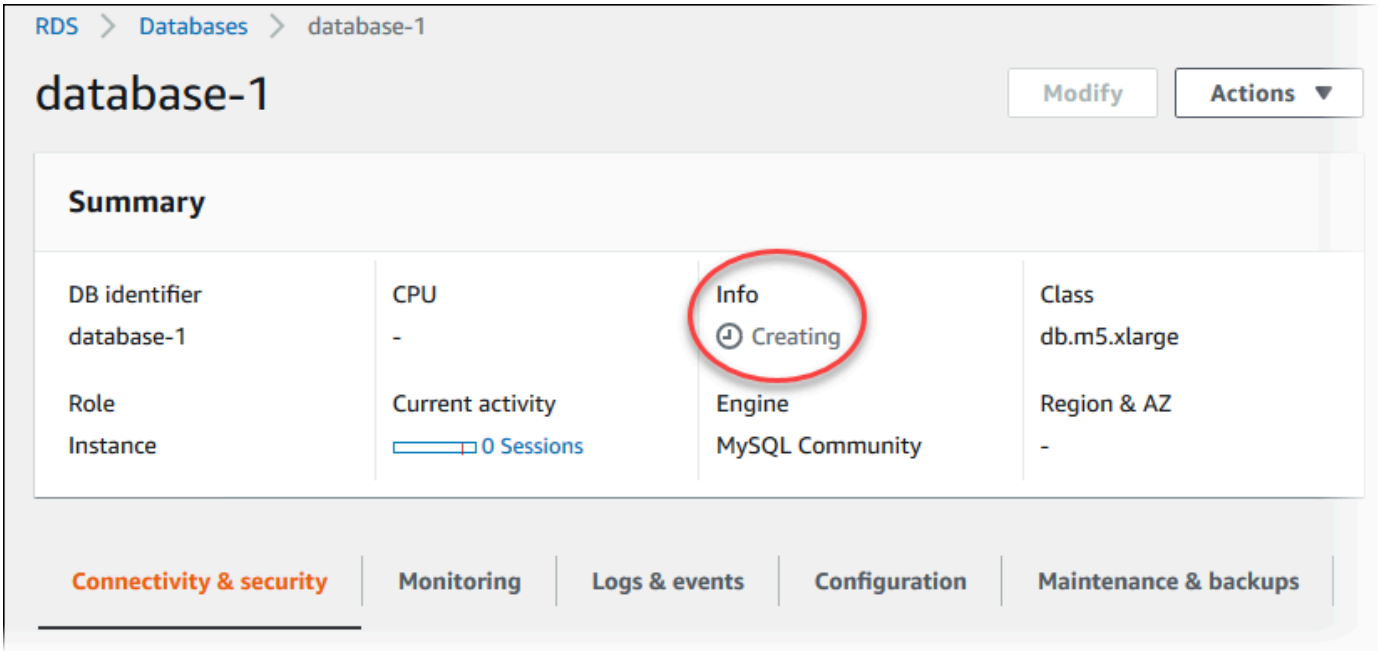
1. DB インスタンスのマスターユーザーネームおよびパスワードを表示するには、「認証情報の詳細の表示」 をデータベースページで選択します。

   これらの認証情報を使用して、マスターユーザーとして DB インスタンスに接続できます。

   > **Important**
   >
   > マスターユーザーのパスワードを再度表示することはできません。記録していない場合は、変更する必要がある場合があります。DB インスタンスが有効になった後にマスターユーザーのパスワードを変更するには、DB インスタンスを変更します。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

2. 「Databases (データベース)」 ページで、新しい DB インスタンスの名前を選択します。

   RDS コンソールに、新規の DB インスタンスの詳細が表示されます。DB インスタンスが作成されて使用できるようになるまで、DB インスタンスのステータスは [作成中] となります。ステータスが [Available] に変わると、DB インスタンスに接続できます。DB インスタンスクラスと割り当てられたストレージによっては、新しい DB インスタンスを使用できるようになるまで数分かかることがあります。



DB インスタンスが利用可能になると、AWS クラウドの RDS DB インスタンスと同じ方法で管理することができます。

AWS CLI

AWS CLIで Outpost に新しい DB インスタンスを作成する前に、まず Outpost で RDS が使用する DB サブネットグループを作成します。

Outpost の DB サブネットグループを作成するには

- db-subnet-groupsコマンドを使用します。`--subnet-ids` で、出力の RDS 用に Outpost のサブネットグループを指定します。

  Linux、macOS、Unix の場合:

```
aws rds create-db-subnet-group \
    --db-subnet-group-name myoutpostdbsubnetgr \
    --db-subnet-group-description "DB subnet group for RDS on Outposts" \
    --subnet-ids subnet-abc123
```

  Windows の場合:

```
aws rds create-db-subnet-group ^
    --db-subnet-group-name myoutpostdbsubnetgr ^
```

```
    --db-subnet-group-description "DB subnet group for RDS on Outposts" ^
    --subnet-ids subnet-abc123
```

AWS CLIを使用して Outspotの RDS DB インスタンスを作成するには

- create-db-instanceコマンドを使用します。Outpost のアベイラビリティーゾーン、Outpost に関連付けられた Amazon VPC セキュリティグループ、Outpost 用に作成した DB サブネットグループを指定します。以下のオプションを含めることができます。
  - `--db-instance-identifier`
  - `--db-instance-class`
  - `--engine` - データベースエンジンです。次のいずれかの値を使用します。
    - MySQL - `mysql` を指定します。
    - PostgreSQL - `postgres` を指定します。
    - Microsoft SQL Server - `sqlserver-ee`、`sqlserver-se`、または `sqlserver-web` を指定します。
    - Oracle - `oracle-se2`、`oracle-se2-cdb`、`oracle-ee`、または `oracle-ee-cdb` を指定します。
  - `--availability-zone`
  - `--vpc-security-group-ids`
  - `--db-subnet-group-name`
  - `--allocated-storage`
  - `--max-allocated-storage`
  - `--master-username`
  - `--master-user-password`
  - `--multi-az | --no-multi-az` — (オプション) スタンバイ DB インスタンスを別のアベイラビリティーゾーンに作成するかどうか。デフォルトは `--no-multi-az` です。

    `--multi-az` オプションは、SQL Server では使用できません。
  - `--backup-retention-period`
  - `--backup-target` - (オプション) 自動バックアップとマニュアルスナップショットを保存する場所です。次のいずれかの値を使用します。
    - `outposts` - Outpost でローカルに保存してください。

- region - 親 に保存してください。AWS リージョンこれは、デフォルト値です。

  --multi-az オプションを使用すると、--backup-target に outposts は使用できません。また、--backup-target に outposts を使用した場合、DB インスタンスにリードレプリカを設定することはできません。

- --storage-encrypted
- --kms-key-id

Example

次の例では、myoutpostdbinstanceという名前のMySQL DBインスタンスを作成し、バックアップを Outpost に保存しています。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier myoutpostdbinstance \
    --engine-version 8.0.17 \
    --db-instance-class db.m5.large \
    --engine mysql \
    --availability-zone us-east-1d \
    --vpc-security-group-ids outpost-sg \
    --db-subnet-group-name myoutpostdbsubnetgr \
    --allocated-storage 100 \
    --max-allocated-storage 1000 \
    --master-username masterawsuser \
    --manage-master-user-password \
    --backup-retention-period 3 \
    --backup-target outposts \
    --storage-encrypted \
    --kms-key-id mykey
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier myoutpostdbinstance ^
    --engine-version 8.0.17 ^
    --db-instance-class db.m5.large ^
    --engine mysql ^
    --availability-zone us-east-1d ^
    --vpc-security-group-ids outpost-sg ^
```

```
    --db-subnet-group-name myoutpostdbsubnetgr ^
    --allocated-storage 100 ^
    --max-allocated-storage 1000 ^
    --master-username masterawsuser ^
    --manage-master-user-password ^
    --backup-retention-period 3 ^
    --backup-target outposts ^
    --storage-encrypted ^
    --kms-key-id mykey
```

DB インスタンスを作成する際の各設定の詳細については、「DB インスタンスの設定」を参照してください。

RDS API

RDS API を使用して Outpost に新しい DB インスタンスを作成するには、まず CreateDBSubnetGroup オペレーションを呼び出して、出力の RDS 用に DB サブネットグループを作成します。SubnetIds で、出力の RDS 用に Outpost のサブネットグループを指定します。

次に、以下のパラメータを使用して CreateDBInstance オペレーションを呼び出します。Outpost のアベイラビリティーゾーン、Outpost に関連付けられた Amazon VPC セキュリティグループ、Outpost 用に作成した DB サブネットグループを指定します。

- AllocatedStorage
- AvailabilityZone
- BackupRetentionPeriod
- BackupTarget

  マルチ AZ DB インスタンスのデプロイを作成する場合、BackupTarget に outposts を使用することはできません。また、BackupTarget に outposts を使用した場合、DB インスタンスにリードレプリカを設定することはできません。
- DBInstanceClass
- DBInstanceIdentifier
- VpcSecurityGroupIds
- DBSubnetGroupName
- Engine
- EngineVersion
- MasterUsername

- MasterUserPassword
- MaxAllocatedStorage (オプション)
- MultiAZ (オプション)
- StorageEncrypted
- KmsKeyID

DB インスタンスを作成する際の各設定の詳細については、「DB インスタンスの設定」を参照してください。

Amazon RDS on AWS Outposts でのリードレプリカの作成

Amazon RDS on AWS Outposts では、DB エンジンの組み込みのレプリケーション機能を使用して、ソースの DB インスタンスからリードレプリカを作成できます。ソース DB インスタンスがプライマリ DB インスタンスになります。プライマリ DB インスタンスに対して行った更新は、リードレプリカに非同期的にコピーされます。読み込みクエリをアプリケーションからリードレプリカにルーティングすることにより、プライマリ DB インスタンスの負荷を軽減できます。リードレプリカを使うと、単一 DB インスタンスのキャパシティ制約にとらわれることなく伸縮自在にスケールアウトし、読み込み負荷の高いデータベースワークロードに対応できます。

RDS on Outposts DB インスタンスからリードレプリカを作成する場合、リードレプリカはお客様が所有する IP アドレス (CoIP) を使用します。詳細については、「AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス」を参照してください。

RDS on Outposts でのリードレプリカの制限事項は次のとおりです。

- RDS on Outposts DB インスタンスで RDS for SQL Server のリードレプリカを作成することはできません。
- RDS on Outposts では、クロスリージョンリードレプリカはサポートされません。
- RDS on Outposts では、カスケードリードレプリカはサポートされません。
- ソース RDS on Outposts DB インスタンスは、ローカルバックアップを持つことができません。ソース DB インスタンスのバックアップターゲットは、AWS リージョン である必要があります。
- リードレプリカにはお客様が所有する IP (CoIP) プールが必要です。詳細については、「AWS Outposts での Amazon RDS 用の顧客所有の IP アドレス」を参照してください。
- RDS on Outposts のリードレプリカは、ソース DB インスタンスと同じ仮想プライベートクラウド (VPC) でのみ作成できます。
- RDS on Outposts のリードレプリカは、ソース DB インスタンスと同じ VPC 内の同じ Outpost または別の Outpost に配置できます。
- AWS KMS External Key Store (XKS) で暗号化された DB インスタンスのリードレプリカを作成することはできません。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、RDS on Outposts DB インスタンスからリードレプリカを作成できます。リードレプリカの詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

コンソール

ソース DB インスタンスからリードレプリカを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. リードレプリカのソースとして使用する DB インスタンスを選択します。
4. [アクション] で [リードレプリカの作成] を選択します。
5. [DB インスタンス識別子] に、リードレプリカの名前を入力します。
6. Outposts 接続性の設定を指定します。この設定は、DB インスタンスの DB サブネットグループを含む仮想プライベートクラウド (VPC) を使用する Outpost 用です。VPC は、Amazon VPC サービスに基づいている必要があります。
7. DB インスタンスクラスを選択します。リードレプリカでもソース DB インスタンスと同等以上の DB インスタンスクラスとストレージタイプを使用することをお勧めします。
8. マルチ AZ 配置の場合、[Create a standby instance (recommended for production usage)] (スタンバイインスタンスの作成 (本番環境での使用に推奨)) をクリックして、別のアベイラビリティーゾーンにスタンバイ DB インスタンスを作成します。

   リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。
9. (オプション) [Connectivity] (接続性) で、[Subnet Group] (サブネットグループ) と [Availability Zone] (アベイラビリティーゾーン) の値を設定します。

   [Subnet Group] (サブネットグループ) と [Availability Zone] (アベイラビリティーゾーン) の値を両方設定すると、リードレプリカは DB サブネットグループのアベイラビリティーゾーンに関連付けられた Outpost に作成されます。

   [Subnet Group] (サブネットグループ) に値を指定し、[Availability Zone] (アベイラビリティーゾーン) に [No preference] (指定なし) を指定した場合、リードレプリカは DB サブネットグループ内のランダムな Outpost に作成されます。
10. [AWS KMS key] では、KMS キーの AWS KMS key 識別子を選択します。

    リードレプリカは、暗号化する必要があります。
11. 必要に応じて、他のオプションを選択します。
12. [Create read replica] を選択します。

リードレプリカが作成されると、RDS コンソールの [Databases] (データベース) ページに表示されます。[Role] (ロール) 列に [Replica] (レプリカ) が表示されます。

AWS CLI

ソース MySQL、PostgreSQL、または Oracle DB インスタンスからリードレプリカを作成するには、AWS CLI コマンド create-db-instance-read-replica を使用します。

`--db-subnet-group-name` および `--availability-zone` オプションを指定することで、リードレプリカを作成する場所を制御できます。

- `--db-subnet-group-name` および `--availability-zone` オプションの両方を指定すると、リードレプリカは DB サブネットグループのアベイラビリティーゾーンに関連付けられた Outpost に作成されます。
- `--db-subnet-group-name` オプションを指定し、`--availability-zone` オプションは指定しない場合、リードレプリカは DB サブネットグループ内のランダムな Outpost に作成されます。
- どちらのオプションも指定しない場合、リードレプリカはソース RDS on Outposts DB インスタンスと同じ Outpost に作成されます。

次の例では、レプリカを作成し、`--db-subnet-group-name` および `--availability-zone` オプションを含めてリードレプリカの場所を指定しています。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance \
    --availability-zone us-west-2a
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --availability-zone us-west-2a
```

RDS API

ソース MySQL、PostgreSQL、または Oracle DB インスタンスからリードレプリカを作成するには、次の必須パラメータを指定して Amazon RDS API CreateDBInstanceReadReplica オペレーションを呼び出します。

- `DBInstanceIdentifier`
- `SourceDBInstanceIdentifier`

`DBSubnetGroupName` および `AvailabilityZone` パラメータを指定することで、リードレプリカを作成する場所を制御できます。

- `DBSubnetGroupName` および `AvailabilityZone` パラメータの両方を指定すると、リードレプリカは DB サブネットグループのアベイラビリティーゾーンに関連付けられた Outpost に作成されます。
- `DBSubnetGroupName` パラメータを指定し、`AvailabilityZone` パラメータは指定しない場合、リードレプリカは DB サブネットグループ内のランダムな Outpost に作成されます。
- どちらのパラメータも指定しない場合、リードレプリカはソース RDS on Outposts DB インスタンスと同じ Outpost に作成されます。

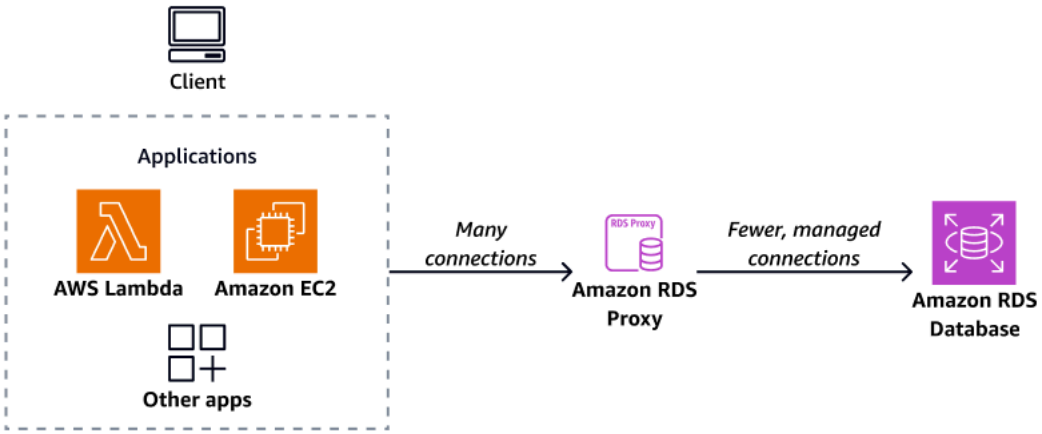
AWS Outposts 上の Amazon RDS での DB インスタンスの復元に関する考慮事項

Amazon RDS on AWS Outpostsで DB インスタンスを復元する場合、一般的に、復元した DB インスタンスの自動バックアップとマニュアルスナップショットのストレージ場所を選択することができます。

- マニュアル DB スナップショットから復元するとき、バックアップを親 AWS リージョン または Outpost のローカルに保存することができます。
- 自動バックアップ (ポイントインタイムリカバリ) から復元するとき、選択肢が少なくなります。
  - 親 AWS リージョン から復元する場合、バックアップは AWS リージョン または Outpost に保存できます。
  - Outpost から復元する場合、バックアップは Outpost にのみ保存できます。

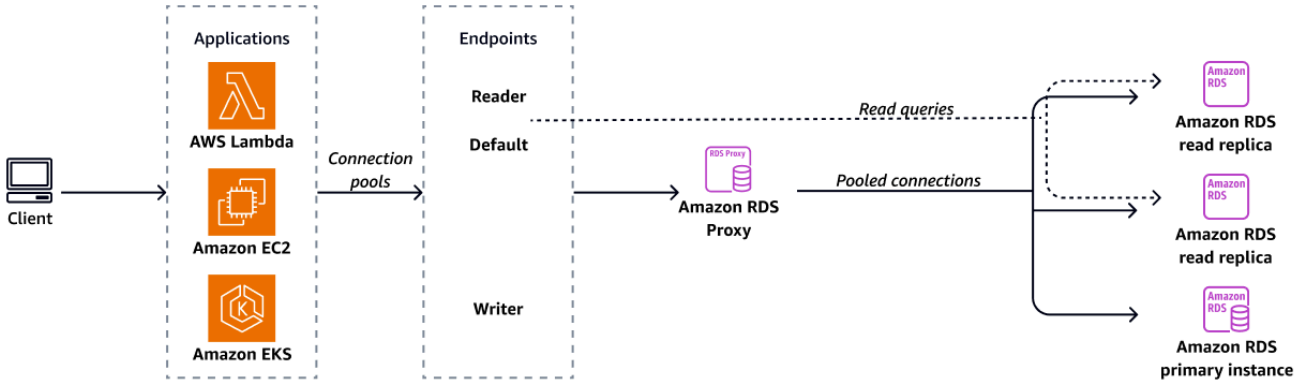
Amazon RDS Proxy

Amazon RDS Proxy を使用すると、アプリケーションでデータベース接続をプールおよび共有して、アプリケーションのスケーリング能力を向上させることができます。RDS Proxy は、アプリケーション接続を維持しながらスタンバイ DB インスタンスに自動的に接続することで、データベースの障害に対するアプリケーションの耐障害性を高めます。RDS Proxy を使用すると、プロキシに接続するクライアントに AWS Identity and Access Management (IAM) 認証を適用でき、プロキシは IAM データベース認証または AWS Secrets Manager に保存されている認証情報を使用してデータベースに接続できます。



RDS Proxy を使用すると、データベーストラフィックの予測不可能なサージを処理できます。そうでない場合、このようなサージは、接続のオーバーサブスクリプションや新しい接続の急速な作成による問題の原因となることがあります。RDS Proxy は、データベース接続プールを確立し、このプール内の接続を再利用します。この方法により、毎回新しいデータベース接続を開くことによるメモリと CPU のオーバーヘッドを回避することができます。オーバーサブスクリプションからデータベースを保護するために、データベース接続の作成数を制御できます。

RDS Proxy は、接続プールからアプリケーション接続をすぐに提供できない場合に、これらの接続の処理順序を決めたり、スロットリングを行ったりします。レイテンシーは増加する場合がありますが、データベースの障害や過負荷が突然発生することはなく、アプリケーションは継続してスケーリングされます。接続リクエスト数が指定した制限を超えると、RDS Proxy はアプリケーション接続を拒否 (負荷を削除) します。同時に、使用可能な容量で RDS が対応できる負荷に対して、予測可能なパフォーマンスを維持します。



認証情報を処理するオーバーヘッドを減らし、新しい接続ごとに安全な接続を確立できます。RDS Proxy は、この作業の一部をデータベースに代わって処理できます。

RDS Proxy には、RDS Proxy でサポートされているエンジンバージョンとの完全な互換性があります。ほとんどのアプリケーションでは、コードを変更せずに RDS Proxy を有効にすることができます。

トピック

- リージョンとバージョンの可用性
- RDS Proxy のクォータと制限事項
- RDS Proxy の使用場所の計画
- RDS Proxy の概念と用語
- RDS Proxy のスタート方法
- RDS Proxy の管理
- Amazon RDS Proxy エンドポイントの操作
- Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング
- RDS Proxy イベントの使用
- RDS Proxy のトラブルシューティング
- RDS Proxy の AWS CloudFormation での使用
- ブルー/グリーンデプロイでの RDS Proxy の使用
- RDS Proxy のベストプラクティス

リージョンとバージョンの可用性

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。RDS Proxy に関する Amazon RDS のバージョンとリージョンの可用性の詳細については、「Amazon RDS Proxy でサポートされているリージョンと DB エンジン」を参照してください。

RDS Proxy のクォータと制限事項

RDS Proxy には以下のクォータと制限が適用されます。

- 各 AWS アカウント ID は 20 個のプロキシに制限されています。アプリケーションがさらに多くのプロキシを必要とする場合、AWS マネジメントコンソール 内の Service Quotas ページから引き上げをリクエストします。Service Quotas ページで、Amazon Relational Database Service (Amazon RDS) を選択し、Proxies を見つけてクォータの引き上げをリクエストします。AWS は、クォータを自動的に引き上げたり、サポート によるリクエストのレビューを保留したりできます。
- 各プロキシには最大 200 個の Secrets Manager シークレットを関連付けることができるため、シークレットを使用するときに接続を最大 200 個の異なるユーザーアカウントに制限できます。
- 各プロキシにはデフォルトのエンドポイントがあり、プロキシの設定済みサブネットから選択された 2 つのアベイラビリティーゾーンにのみプロビジョニングされます。複数の AZ にサブネットが設定されている場合、そのうちの 2 つが選択されます。
- プロキシごとに最大 20 のその他のプロキシエンドポイントを追加できます。これらの追加のエンドポイントは、作成時に指定されたすべてのアベイラビリティーゾーンにプロビジョニングされます。これらのエンドポイントを作成、表示、変更、および削除できます。
- レプリケーション設定の RDS DB インスタンスの場合、リードレプリカではなく、ライター DB インスタンスにのみプロキシを関連付けることができます。
- RDS Proxy は、データベースと同じ仮想プライベートクラウド (VPC) 内に存在する必要があります。プロキシにはパブリックにアクセスできませんが、データベースにはパブリックにアクセスできます。例えば、ローカルホストでデータベースをプロトタイプ化する場合、プロキシへの接続を許可するために必要なネットワーク要件を設定しない限り、プロキシに接続することはできません。これは、ローカルホストがプロキシの VPC の外側にあるためです。
- `dedicated` に設定されたテナンシーを含む VPC では、RDS Proxy を使用できません。
- `Enforce Mode` が有効になっている暗号化コントロールを持つ VPC では RDS Proxy を使用できません。

- IPv6 エンドポイントネットワークタイプの場合は、IPv6 のみをサポートするように VPC とサブネットを設定します。IPv4 と IPv6 の両方のターゲット接続ネットワークタイプで、デュアルスタックモードをサポートするように VPC とサブネットを設定します。
- IAM 認証が有効になっている RDS DB インスタンスで RDS Proxy を使用する場合、プロキシは、IAM 認証または Secrets Manager に保存されている認証情報を使用してデータベースに接続できます。プロキシに接続するクライアントは、IAM 認証情報を使用して認証する必要があります。設定手順の詳細については、「RDS Proxy のデータベース認証情報の設定」および「RDS Proxy の IAM 認証の設定」を参照してください。
- SSL ホスト名の検証を使用するときは、カスタム DNS で RDS Proxy を使用することができません。
- 各プロキシは、1 つのターゲット DB インスタンスに関連付けることができます。ただし、同じ DB インスタンスに複数のプロキシを関連付けることができます。
- ステートメントのテキストサイズが 16 KB を超える場合、プロキシはセッションを現在の接続に固定します。
- 特定のリージョンには、プロキシを作成する際に考慮すべきアベイラビリティーゾーン (AZ) の制限があります。米国東部 (バージニア北部) リージョンは、`use1-az3` アベイラビリティーゾーンで RDS Proxy サポートしていません。米国西部 (北カリフォルニア) リージョンは、`usw1-az2` アベイラビリティーゾーンで RDS プロキシをサポートしていません。プロキシの作成時にサブネットを選択するときは、上記のアベイラビリティーゾーンのサブネットを選択しないようにしてください。
- 現在、RDS プロキシはグローバル条件コンテキストキーをサポートしていません。

  グローバル条件コンテキストキーの詳細については、「IAM ユーザーガイド」の「AWS グローバル条件コンテキストキー」を参照してください。
- RDS Custom for SQL Server で RDS Proxy を使用することはできません。
- データベースパラメータグループの変更をプロキシに反映するには、変更をすぐに適用することを選択した場合でも、インスタンスの再起動が必要です。クラスターレベルのパラメータには、クラスター全体の再起動が必要です。
- プロキシターゲットを登録すると、プロキシによって `rdsproxyadmin` DB ユーザーが自動的に作成されます。これは、プロキシ機能に不可欠な保護されたユーザーです。`rdsproxyadmin` ユーザーの改ざんは、いかなる権限においても、避けてください。`rdsproxyadmin` ユーザーやそのアクセス許可を削除または変更すると、アプリケーションへのプロキシが完全に利用できなくなる可能性があります。

RDS for MariaDB バージョン 12.3 以降、データベースエンジンはこの保護を適用します。`rdsproxyadmin` に対して、`CREATE`、`DROP`、`RENAME`、`GRANT`、`REVOKE`、`SET PASSWORD` を試行すると、エラーが返されます。詳細については、「Amazon RDS での MariaDB のセキュリティ」を参照してください。

以下のセクションに各 DB エンジンに関するその他の制限事項については、以下のセクションを参照してください。

- RDS for MariaDB の追加の制限事項
- RDS for Microsoft SQL Server の追加の制限事項
- RDS for MySQL の追加の制限事項
- RDS for PostgreSQL の追加の制限事項

RDS for MariaDB の追加の制限事項

RDS for MariaDB データベースを使用した RDS Proxy には、以下の追加制限が適用されます。

- 現在、すべてのプロキシはポート 3306 で MariaDB をリッスンします。プロキシは引き続き、データベース設定で指定したポートを使用してデータベースに接続します。
- RDS Proxy は、Amazon EC2 インスタンスのセルフマネージド MariaDB データベースでは使用できません。
- DB パラメータグループが `1` に設定された `read_only` パラメータを含む RDS for MariaDB DB インスタンスでは、RDS Proxy を使用できません。
- RDS Proxy は MariaDB 圧縮モードをサポートしていません。例えば、`mysql` コマンドの `--compress` オプションや `-C` オプションで使用される圧縮はサポートされていません。
- 一部の SQL ステートメントと関数は、固定させずに接続状態を変更できます。固定の最新の動作については、「RDS Proxy の固定の回避」を参照してください。
- RDS Proxy では、MariaDB `auth_ed25519` プラグインはサポートされていません。
- RDS Proxy では、MariaDB データベースの Transport Layer Security (TLS) バージョン 1.3 はサポートされていません。
- RDS Proxy が同じデータベース接続を再利用して別のクエリを実行すると、`GET DIAGNOSTIC` コマンドを処理するデータベース接続が不正確な情報を返すことがあります。これは、RDS プロキシがデータベース接続を多重化する場合に発生する可能性があります。詳細については、「RDS Proxy の概念」を参照してください。

- RDS Proxy は現在、MariaDB で `ClientPasswordAuthType` の `caching_sha2_password` オプションをサポートしていません。

> **Important**
>
> MariaDB データベースに関連付けられているプロキシの場合、初期化クエリで設定パラメータ `sql_auto_is_null` を `true` または 0 以外の値に設定しないでください。その場合、アプリケーションの動作が正常でなくなる場合があります。

RDS for Microsoft SQL Server の追加の制限事項

Microsoft SQL Server データベースを使用した RDS Proxy には、以下の追加制限が適用されます。

- プロキシ用に作成する必要がある Secrets Manager シークレットの数は、DB インスタンスが使用する照合順序によって異なります。例えば、DB インスタンスが大文字と小文字を区別する照合順序を使用しているとします。アプリケーションが「Admin」と「admin」の両方を受け入れる場合、プロキシには 2 つの別々のシークレットが必要です。SQL Server Agent の詳細については、Microsoft SQL Server のドキュメントを参照してください。
- RDS Proxy は、Active Directory を使用する接続をサポートしていません。
- トークンプロパティをサポートしていないクライアントでは IAM 認証を使用できません。詳細については、「Microsoft SQL Server への接続に関する考慮事項」を参照してください。
- `@@IDENTITY`、`@@ROWCOUNT`、および `SCOPE_IDENTITY` の結果は必ずしも正確ではありません。回避策として、同じセッションステートメントでそれらの値を取得して、正しい情報が返されることを確認します。
- 接続で複数のアクティブな結果セット (MARS) が使用されている場合、RDS プロキシは初期化クエリを実行しません。MARS の詳細については、 Microsoft SQL Server のドキュメントを参照してください。
- 現在、RDS Proxy は、メジャーバージョンの SQL Server 2022 で実行する RDS for SQL Server DB インスタンスをサポートしていません。
- RDS Proxy は、メジャーバージョンの SQL Server 2014 で実行される RDS for SQL Server DB インスタンスをサポートしていません。
- RDS Proxy は、1 つの TLS レコードで複数のレスポンスメッセージを処理できないクライアントアプリケーションをサポートしていません。
- RDS Proxy は、RDS for SQL Server のエンドツーエンドの IAM 認証をサポートしていません。

RDS for MySQL の追加の制限事項

RDS for MySQL データベースを使用した RDS Proxy には、以下の追加制限が適用されます。

- RDS Proxy の `caching_sha2_password` 認証のサポートには、安全な (TLS) 接続が必要です。
- RDS Proxy の`caching_sha2_password` のサポートには、特定の go-sql ドライバーバージョンとの互換性の問題があることが知られています。
- MySQL 8.4 C ドライバーを使用した場合、パラメータ数がプリペアドステートメントのプレースホルダー数を超えると、`mysql_stmt_bind_named_param` API が不正な形式のパケットを生成する可能性があります。これにより、誤ったレスポンスが生じます。MySQL リードレプリカの詳細については、MySQL バグレポートを参照してください。
- 現在、すべてのプロキシはポート 3306 で MySQL をリッスンします。プロキシは引き続き、データベース設定で指定したポートを使用してデータベースに接続します。
- RDS Proxy は、EC2 インスタンスのセルフマネージド MySQL データベースでは使用できません。
- DB パラメータグループが `1` に設定された `read_only` パラメータを含む RDS for MySQL DB インスタンスでは、RDS Proxy を使用できません。
- RDS Proxy は MySQL の圧縮モードをサポートしていません。例えば、`mysql` コマンドの `--compress` オプションや `-C` オプションで使用される圧縮はサポートされていません。
- RDS プロキシが同じデータベース接続を再利用して別のクエリを実行すると、`GET DIAGNOSTIC` コマンドを処理するデータベース接続が不正確な情報を返すことがあります。これは、RDS プロキシがデータベース接続を多重化する場合に発生する可能性があります。
- `SET LOCAL` など、一部の SQL ステートメントと関数は、ピニングを起こさずに接続状態を変更できます。固定の最新の動作については、「RDS Proxy の固定の回避」を参照してください。
- マルチステートメントクエリでの `ROW_COUNT()` 関数の使用はサポートされていません。
- RDS Proxy は、1 つの TLS レコードで複数のレスポンスメッセージを処理できないクライアントアプリケーションをサポートしていません。
- RDS Proxy は MySQL デュアルパスワードをサポートしていません。
- セッション状態変数を設定するように RDS DB パラメータグループの `init_connect` パラメータを設定すると、RDS Proxy が期待どおりに動作しない場合があります。代わりに、プロキシを使用してデータベースに接続する際に、セッション初期化ステートメントを実行するように、プロキシの初期化クエリを設定します。

> **Important**
>
> MySQL データベースに関連付けられているプロキシの場合、初期化クエリで設定パラメータ `sql_auto_is_null` を `true` または 0 以外の値に設定しないでください。その場合、アプリケーションの動作が正常でなくなる場合があります。

RDS for PostgreSQL の追加の制限事項

RDS for PostgreSQL データベースを使用した RDS Proxy には、以下の追加制限が適用されます。

- RDS Proxy は PostgreSQL のセッション固定フィルターをサポートしていません。
- 現在、すべてのプロキシはポート 5432 で PostgreSQL をリッスンします。
- PostgreSQL の場合、RDS Proxy は現在、`CancelRequest` を発行してクライアントからのクエリのキャンセルをサポートしていません。これは例えば、インタラクティブな psql セッションで長時間実行されるクエリを、Ctrl+C を使用してキャンセルする場合に相当します。
- PostgreSQL 関数 lastval の結果は必ずしも正確ではありません。回避策としては、INSERT ステートメントを `RETURNING` 句と共に使用します。
- RDS Proxy は現在、ストリーミングレプリケーションモードをサポートしていません。
- RDS Proxy は直接 SSL ネゴシエーションモードをサポートしていません。
- RDS Proxy は、PostgreSQL メッセージングプロトコルのバージョン 3.0 のみをサポートしています。
- RDS for PostgreSQL 16 では、`scram_iterations` 値の変更は、プロキシとデータベース間の認証プロセスにのみ影響します。具体的には、`ClientPasswordAuthType` を `scram-sha-256` に設定した場合、`scram_iterations` 値に加えられたカスタマイズは、クライアントとプロキシ間のパスワード認証に影響しません。代わりに、クライアントとプロキシ間のパスワード認証の反復値は 4096 に固定されます。
- RDS Proxy が機能するには、デフォルトの `postgres` データベースが RDS for PostgreSQL インスタンスに存在する必要があります。アプリケーションが異なるデータベースを使用している場合でも、このデータベースを削除しないでください。
- `ALTER ROLE` を使用して `SET ROLE` でユーザーロールを変更した場合、そのユーザーによるプロキシへの後続の接続で固定が発生した場合に、このロール設定は使用されないことがあります。これを防ぐには、プロキシを使用するときにプロキシの初期化クエリで `SET ROLE` を使用します。詳細については、「Amazon RDS のプロキシの作成」の「初期化クエリ」を参照してください。

> **Important**
>
> PostgreSQL データベースを使用する既存のプロキシでは、SCRAM のみを使用するようにデータベース認証を変更すると、プロキシが最大 60 秒間使用できなくなります。この問題を回避するには、以下のいずれかの方法で対応します。
>
> - データベースが SCRAM と MD5 認証の両方を許可していることを確認します。
> - SCRAM 認証のみを使用するには、新しいプロキシを作成し、アプリケーショントラフィックを新しいプロキシに移行してから、以前にデータベースに関連付けられていたプロキシを削除します。

RDS Proxy の使用場所の計画

RDS Proxy を使用することで最大の利点を得られる DB インスタンス、クラスター、およびアプリケーションを判別できます。そのためには、以下の要因を考慮します。

- 「接続が多すぎます」エラーが頻繁に発生する DB インスタンスは、プロキシと関連付けることをお勧めします。これは多くの場合、ConnectionAttempts CloudWatch メトリクスの値が高いことが特徴です。プロキシを使用すると、アプリケーションは多数のクライアント接続を開くことができます。一方、プロキシは DB インスタンスへの長続きする接続の数をより少なく管理します。
- より小さい AWS インスタンスクラス (T2 や T3 など) を使用する DB インスタンスの場合、プロキシを使用すると、メモリ不足状態を回避できます。また、接続を確立するための CPU オーバーヘッドを削減することもできます。メモリ不足状態は、多数の接続を処理するときに発生する場合があります。
- Amazon CloudWatch の特定のメトリクスをモニタリングして、DB インスタンスが特定のタイプの制限に近づいているかどうかを判断できます。これらの制限は、接続数や接続管理関連のメモリに適用されます。また、特定の CloudWatch メトリクスをモニタリングすることで、DB インスタンスが、多数の存続期間の短い接続を処理しているかどうかも判断できます。このような接続を開いたり閉じたりすると、データベースにパフォーマンスのオーバーヘッドが生じる可能性があります。モニタリングするメトリクスの詳細については、「Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。
- AWS Lambda 関数も、プロキシの使用に適しています。これらの関数で頻繁に行う短いデータベース接続は、RDS Proxy の接続プールを使用することで利点を得られます。Lambda アプリケーションコードでデータベース認証情報を管理する代わりに、Lambda 機能に設定済みの IAM 認証を利用できます。

- 通常、多数のデータベース接続を開いたり閉じたりし、また、接続プーリング機構が組み込まれていないアプリケーションは、プロキシの使用に適しています。
- 長期にわたって多数の接続を開いたままにするアプリケーションは、通常、プロキシの使用に適しています。SaaS (サービスとしてのソフトウェア) や e コマースなどの業界のアプリケーションは、接続を開いたままにしておくことで、データベースリクエストのレイテンシーを最小化できる場合があります。RDS Proxy を使用すると、アプリケーションは、DB インスタンスに直接接続する場合よりも多くの接続を開いたままにできます。
- IAM 認証や Secrets Manager は設定が複雑であるという理由で、すべての DB インスタンスには導入されていない場合があります。プロキシは、特定のアプリケーションのクライアント接続に対して認証ポリシーを適用できます。Lambda アプリケーションコードでデータベース認証情報を管理する代わりに、Lambda 機能に設定済みの IAM 認証を利用できます。
- RDS Proxy は、データベース障害に対してアプリケーションの回復力と透過性を高めるのに役立ちます。RDS Proxy はドメインネームシステム (DNS) キャッシュをバイパスすることで、Amazon RDS マルチ AZ DB インスタンス のフェイルオーバー時間を最大 66% 短縮できます。また、RDS Proxy は、アプリケーション接続を維持したまま、新しいデータベースインスタンスにトラフィックを自動的にルーティングします。これにより、アプリケーションに対して透明性の高いフェイルオーバーを行うことができます。
- ブルー/グリーンデプロイを使用してデータベースを更新する場合、RDS Proxy を使用してスイッチオーバーのダウンタイムを最小限に抑えることができます。スイッチオーバー中、RDS Proxy は移行を把握し、グリーン環境が新しい本番環境になると接続をグリーン環境に自動的にリダイレクトするため、DNS 伝播の遅延がなくなります。

RDS Proxy の概念と用語

RDS Proxy を使用すると、Amazon RDS DB インスタンス の接続管理をシンプルにできます。

RDS Proxy は、クライアントアプリケーションとデータベースとの間のネットワークトラフィックを処理します。これを行うアクティブな方法として、まず、データベースプロトコルを確認します。次に、その動作をアプリケーションからの SQL オペレーションとデータベースからの結果セットに基づいて調整します。

RDS Proxy により、データベースの接続管理から発生するメモリと CPU のオーバーヘッドが減ります。アプリケーション接続を同時に開く数が多いほど、データベースに必要なメモリと CPU リソースが少なくなります。また、アプリケーションの接続を閉じてから長いアイドル状態の後でアプリケーションを再度開くためのロジックは不要です。同様に、データベース問題の発生時に接続を再確立するために必要なアプリケーションロジックも減ります。

RDS Proxy のインフラストラクチャは可用性が高く、複数のアベイラビリティーゾーン (AZ) にデプロイできます。RDS Proxy の計算、メモリ、およびストレージは、RDS DB インスタンスから独立しています。この分離によって、データベースサーバーのオーバーヘッドが減り、そのリソースをデータベースワークロードの処理に集中させることができます。RDS Proxy コンピューティングリソースはサーバーレスであり、データベースのワークロードに基づいて自動的にスケーリングされます。

トピック

- RDS Proxy の概念
- 接続プーリング
- RDS Proxy のセキュリティ
- フェイルオーバー
- 取引

RDS Proxy の概念

RDS Proxy は、接続プールや以降のセクションで説明するその他の機能を実行するインフラストラクチャを処理します。プロキシは、RDS コンソールの [プロキシ] ページに表示されます。

各プロキシは、単一の RDS DB インスタンスへの接続を処理します。プロキシは、RDS のマルチ AZ DB インスタンスまたはクラスター で、現在のライターインスタンスを自動的に判別します。

プロキシで開いたままにしてデータベースアプリケーションで使用できるようにした接続は、接続プールを形成します。

デフォルトでは、RDS Proxy はセッション内の各トランザクションの後で接続を再利用できます。このトランザクションレベルの再利用は、多重化と呼ばれます。RDS Proxy が接続を接続プールから一時的に削除して再利用する場合、そのオペレーションは、接続の借用と呼ばれます。この再利用が安全であれば、RDS Proxy はその接続を接続プールに返します。

場合によっては、RDS Proxy は、現在のセッションの外でデータベース接続を再利用しても安全であると判断できません。このような場合、セッションが終了するまで、セッションは同じ接続で維持されます。このフォールバック動作は、固定と呼ばれます。

プロキシにはデフォルトのエンドポイントがあります。Amazon RDS DB インスタンスを使用する場合は、このエンドポイントに接続します。インスタンス に直接接続する読み取り/書き込みエンドポイントに接続する代わりに、この操作を行います。RDS DB クラスターの場合、追加の読み取り/書

き込みエンドポイントと読み取り専用エンドポイントを作成することもできます。詳細については、「プロキシエンドポイントの概要」を参照してください。

例えば、接続プールを使用しなくても、読み取り/書き込み接続に、引き続きクラスターエンドポイントを使用できます。負荷分散された読み取り専用接続に、引き続きリーダーエンドポイントを使用できます。クラスター内の特定の DB インスタンスの診断やトラブルシューティングに、引き続きインスタンスエンドポイントを使用できます。他の AWS のサービス (AWS Lambda など) を使用して RDS データベースに接続している場合、プロキシエンドポイントを使用するには、接続設定を変更します。例えば、RDS Proxy 機能を利用してプロキシエンドポイントを通じてデータベースにアクセスすることを Lambda 関数に許可するように指定します。

プロキシごとにターゲットグループがあります。このターゲットグループは、プロキシが接続できる RDS DB インスタンスを指します。プロキシに関連付けられた RDS DB インスタンスは、そのプロキシのターゲットと呼ばれます。コンソールでプロキシを作成すると、RDS Proxy によって自動的に、対応するターゲットグループも作成され、関連するターゲットが登録されます。

エンジンファミリーは、同じ DB プロトコルを使用する関連データベースエンジンのセットです。作成するプロキシごとにエンジンファミリーを選択します。

接続プーリング

各プロキシは、関連付けられた RDS データベース のライターインスタンスとリーダーインスタンスに対して個別に接続プーリングを実行します。接続プーリングは、接続の開閉や、多数の接続を同時に開いたままにすることに伴うオーバーヘッドを削減するための最適化方法です。このオーバーヘッドには、新しい各接続を処理するために必要なメモリも含まれます。また、各接続を閉じて新しい接続を開くため、CPU のオーバーヘッドを伴います。例えば、TLS/SSL (Transport Layer Security/Secure Sockets Layer) のハンドシェイク、認証、ネゴシエーション機能などがあります。接続プーリングは、アプリケーションロジックを簡素化します。同時に開いている接続の数を最小限に抑えるためのアプリケーションコードを記述する必要はありません。

各プロキシは、接続の多重化も実行します。これは接続の再利用とも呼ばれます。多重化により、RDS Proxy は 1 つの基となるデータベース接続を使用して 1 つのトランザクションのすべてのオペレーションを実行します。RDS は、次のトランザクションに別の接続を使用できます。プロキシへの接続は同時に多数を開くことができます。プロキシから DB インスタンスやクラスターへの接続の数はより少なく保持されます。これにより、データベースサーバーでの接続に伴うメモリオーバーヘッドがさらに最小化されます。「接続が多すぎます」エラーが発生する可能性も減ります。

RDS Proxy のセキュリティ

RDS Proxy は、TLS/SSL や AWS Identity and Access Management (IAM) などの既存の RDS セキュリティメカニズムを使用します。これらのセキュリティ機能の概要については、「Amazon RDS でのセキュリティ」を参照してください。また、RDS が認証、認可、その他のセキュリティ領域でどのように機能するかについても、よく理解する必要があります。

RDS Proxy は、クライアントアプリケーションおよび基となるデータベースの間に別のセキュリティ層を追加します。例えば、基になる DB インスタンスが古いバージョンの TLS をサポートしている場合でも、TLS 1.3 を使用してプロキシに接続できます。プロキシがデータベースユーザーとパスワードによる認証方法を使用してデータベースに接続する場合でも、IAM ロールを使用してプロキシに接続できます。この方法により、DB インスタンス自体の高額な移行作業を必要とせずに、データベースアプリケーションに強力な認証要件を適用できます。

RDS Proxy では、次の認証方法を使用できます。

- データベースの認証情報
- 標準 IAM 認証
- エンドツーエンド IAM 認証

RDS Proxy での IAM の使用

RDS Proxy には、IAM 認証の 2 つの方法があります。

- 標準 IAM 認証: プロキシが Secrets Manager に保存されている認証情報を使用してデータベースに接続している間、プロキシへの接続に IAM 認証を適用します。これにより、データベースでパスワードによるネイティブ認証方法が使用されている場合でも、データベースへのアクセスに IAM 認証を適用できます。プロキシは Secrets Manager からデータベース認証情報を取得し、アプリケーションに代わってデータベースへの認証を処理します。
- エンドツーエンド IAM 認証: アプリケーションからプロキシ経由でデータベースに直接接続するための IAM 認証を適用します。エンドツーエンド IAM 認証は、セキュリティ設定を簡素化し、Secrets Manager でのデータベース認証情報管理を回避します。この追加のセキュリティレイヤーは、クライアントアプリケーションからデータベースへの IAM ベースのアクセスコントロールを適用します。

標準 IAM 認証を使用するには、認証に Secrets Manager シークレットを使用するようにプロキシを設定し、クライアント接続の IAM 認証を有効にします。アプリケーションは IAM を使用してプロキ

シを認証し、プロキシは Secrets Manager から取得した認証情報を使用してデータベースを認証します。

エンドツーエンドの IAM 認証を使用するには、プロキシを作成または変更時にデフォルトの認証スキームを設定するときに、IAM 認証を使用するようにプロキシを設定します。

エンドツーエンド IAM 認証では、プロキシに関連付けられた IAM ロールを更新して、`rds-db:connect` アクセス許可を付与する必要があります。エンドツーエンドの IAM 認証により、Secrets Manager シークレットを介して個々のデータベースユーザーをプロキシに登録する必要がなくなります。

RDS Proxy での TLS/SSL の使用

TLS/SSL プロトコルを使用して RDS Proxy に接続できます。

> **Note**
>
> RDS Proxy は AWS Certificate Manager (ACM) の証明書を使用します。RDS Proxy を使用している場合は、Amazon RDS 証明書をダウンロードしたり、RDS Proxy 接続を使用するアプリケーションを更新したりする必要はありません。

プロキシとデータベース間のすべての接続に TLS を適用するには、AWS マネジメントコンソールでプロキシを作成または変更するときに [Transport Layer Security が必要] 設定を指定できます。

RDS Proxy により、セッションでクライアントと RDS Proxy エンドポイント間でも TLS/SSL が必ず使用されるようにもできます。そのためには RDS Proxy で、クライアント側の要件を指定します。SSL セッション可変は、RDS Proxy を使用したデータベースへの SSL 接続には設定されません。

- RDS for MySQL の場合、`mysql` コマンドを実行するときに、`--ssl-mode` パラメータを使用してクライアント側の要件を指定します。
- Amazon RDS PostgreSQL の場合、`psql` コマンドを実行するときに、`conninfo` 文字列の一部として `sslmode=require` を指定します。

RDS Proxy は、TLS プロトコルバージョン 1.0、1.1、1.2、および 1.3 をサポートしています。プロキシに接続するには、基になるデータベースで使用しているものよりも高いバージョンの TLS を使用します。

デフォルトでは、クライアントプログラムは RDS Proxy との暗号化された接続を確立します。さらに制御する場合は、`--ssl-mode` オプションを使用できます。RDS Proxy は、クライアント側のすべての SSL モードをサポートします。

クライアントの SSL モードは次のとおりです。

PREFERRED

SSL は初期の選択肢ですが、必須ではありません。

DISABLED

SSL は許可されていません。

REQUIRED

SSL を強制します。

VERIFY_CA

SSL を義務化し、認証機関 (CA) を検証します。

VERIFY_IDENTITY

SSL を義務化し、CA と CA ホスト名を検証します。

`--ssl-mode`、`VERIFY_CA`、または `VERIFY_IDENTITY` でクライアントを使用する場合は、CA を指す `--ssl-ca` を `.pem` 形式で指定します。`.pem` ファイルを使用するには、Amazon Trust Services からすべてのルート CA PEM をダウンロードし、1 つの `.pem` ファイルに配置します。

RDS Proxy は、ドメインとそのサブドメインの両方に適用されるワイルドカード証明書を使用します。`mysql` クライアントを使用して SSL モード `VERIFY_IDENTITY` で接続する場合、現時点では、MySQL 8.0 互換の `mysql` コマンドを使用する必要があります。

RDS Proxy でのカスタム暗号スイートのサポート

TLS 暗号スイートは、クライアントとサーバーの接続を保護するために使用される、暗号化アルゴリズムの標準化された組み合わせです。それらは、暗号化とデータの整合性がどのように適用されるかを決定します。TLS 1.2 以前では、キー交換と認証方法も規定されています。最終的な暗号スイートは、TLS ハンドシェイク中にネゴシエートされます。

RDS/Aurora MySQL と MariaDB

カスタム暗号スイートは、以下を使用してパラメータグループを介して設定できます。

- `ssl_cipher` – TLS 1.2 接続用
- `tls_ciphersuites` – TLS 1.3 接続用

エンジンでサポートされている暗号スイートについては、Aurora MySQL の暗号スイートの設定についてのページを参照してください。

RDS/Aurora PostgreSQL

カスタム暗号スイートは、以下を使用してパラメータグループを介して設定できます。

- `ssl_ciphers` – TLS 1.2 接続用。Aurora PostgreSQL 17 では、TLS 1.3 暗号スイートを含めることもできます。
- `ssl_tls13_ciphers` – TLS 1.3 接続用

エンジンでサポートされている暗号スイートについては、RDS PostgreSQL 用の暗号スイートの設定およびAurora PostgreSQL 用の暗号スイートの設定についてのページを参照してください。

RDS Proxy の動作

RDS Proxy を使用する場合、データベースに設定された暗号スイート設定が、RDS Proxy とデータベース間の接続に適用されます。

RDS Proxy は、アプリケーションと RDS Proxy 間の接続に対しても、同じ許可リストに登録された暗号スイートを適用します。これにより、アプリケーションからプロキシ、データベースまで、カスタム暗号をエンドツーエンドで適用できるため、環境全体で一貫したセキュリティ標準を維持できます。

RDS Proxy は、TLS 接続用の暗号スイートの特定のサブセットのみをサポートします。データベースが追加の暗号スイートをサポートしている場合でも、RDS Proxy では次の暗号スイートのみがサポートされています。プロキシへのクライアント接続に他の暗号スイートを適用すると、接続の設定が失敗します。

RDS Proxy でサポートされている暗号スイート

#	TLS バージョン	OpenSSL 形式の暗号	IANA 形式の暗号
1	TLS 1.0、TLS 1.1、TLS 1.2	ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

#	TLS バージョン	OpenSSL 形式の暗号	IANA 形式の暗号
2	TLS 1.0、TLS 1.1、TLS 1.2	DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
3	TLS 1.0、TLS 1.1、TLS 1.2	ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
4	TLS 1.0、TLS 1.1、TLS 1.2	DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
5	TLS 1.2	ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
6	TLS 1.2	DHE-RSA-AES256-GCM-SHA384	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
7	TLS 1.2	DHE-RSA-AES256-SHA256	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
8	TLS 1.2	ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
9	TLS 1.2	DHE-RSA-AES128-GCM-SHA256	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
10	TLS 1.2	DHE-RSA-AES128-SHA256	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
11	TLS 1.2	DHE-RSA-CHACHA20-POLY1305	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
12	TLS 1.3	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
13	TLS 1.3	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256

フェイルオーバー

フェイルオーバーは、元のデータベースインスタンスが使用できなくなったときに、別のインスタンスに置き換える高可用性機能です。フェイルオーバーは、データベースインスタンスの問題が原因で発生することがあります。また、通常のメンテナンス手順の一環として、データベースのアップグレード時などに発生することもあります。フェイルオーバーは、マルチ AZ 設定の RDS DB インスタンスに適用されます。

プロキシを介して接続すると、データベースのフェイルオーバーに対してアプリケーションの耐障害性が高くなります。元の DB インスタンスが使用できなくなると、RDS Proxy はアイドル状態のアプリケーション接続を切断せずにスタンバイデータベースに接続します。これにより、フェイルオーバープロセスが高速化および簡素化されます。これは、通常の再起動やデータベース問題に伴う停止よりも、アプリケーションの停止が短くなります。

RDS Proxy を使用していない場合は、フェイルオーバーによって短時間の停止が発生します。停止中は、フェイルオーバー中のデータベースに対して書き込み操作を実行できません。既存のデータベース接続はすべて中断されるため、アプリケーションで接続を再度開く必要があります。利用できなくなった DB インスタンスの代わりに読み取り専用 DB インスタンスが昇格すると、新しい接続および書き込みオペレーションでデータベースが使用可能になります。

DB フェイルオーバー中、RDS Proxy は引き続き同じ IP アドレスで接続を受け入れ、接続を自動的に新しいプライマリ DB インスタンスに転送します。RDS Proxy を介して接続しているクライアントは、以下の影響を受けません。

- フェイルオーバー時のドメインネームシステム (DNS) 伝播の遅延。
- ローカル DNS キャッシュ。
- 接続タイムアウト。
- どの DB インスタンスが現在のライターであるかの不確実性。
- 接続を閉じずに使用不能になった以前のライターからのクエリ応答の待機。

アプリケーション独自の接続プールがある場合は、RDS Proxy を経由することで、ほとんどの接続がフェイルオーバーやその他の中断時にも存続します。トランザクションや SQL ステートメントの途中の接続のみがキャンセルされます。RDS Proxy は、すぐに新しい接続を受け入れます。データベースライターが使用できない場合、RDS Proxy は着信リクエストをキューに入れます。

アプリケーション独自の接続プールがない場合は、RDS Proxy を使用することで、接続速度が高速になり、開いたままの接続の数を増やすことができます。データベースからの頻繁な再接続に伴う高

額なオーバーヘッドがオフロードされます。そのために、RDS Proxy 接続プールに保持されているデータベース接続を再利用します。このアプローチは、設定コストが大きい TLS 接続で特に重要です。

取引

1 つのトランザクション内のすべてのステートメントは、常に同じ基となるデータベース接続を使用します。このトランザクションが終了すると、この接続は別のセッションで使用可能になります。トランザクションを粒度の単位として使用すると、次のような結果になります。

- RDS for MySQL `autocommit` 設定がオンのときは、各ステートメントが終わるたびに接続の再利用が発生することがあります。
- 逆に、`autocommit` 設定が無効になっていると、セッションで最初に発行したステートメントによって新しいトランザクションが開始されます。例えば、`SELECT`、`INSERT`、`UPDATE` などのデータ操作言語 (DML) ステートメントのシーケンスを入力したとします。この場合、`COMMIT` または `ROLLBACK` を発行するか、またはトランザクションが終了するまで、接続の再利用は起こりません。
- データ定義言語 (DDL) ステートメントを入力すると、そのステートメントの完了後にトランザクションが終了します。

RDS Proxy は、データベースクライアントアプリケーションで使用されているネットワークプロトコルを介してトランザクションが終了したことを検出します。トランザクションの検出は、SQL ステートメントのテキスト内にある `COMMIT` や `ROLLBACK` などのキーワードに依存しません。

場合によっては、セッションを別の接続に移動することが実用的でないようなデータベースリクエストが RDS Proxy で検出されることがあります。このような場合、セッションの残りの部分では、その接続の多重化がオフになります。セッションに対して多重化が実用的であることを RDS Proxy で確信できない場合は、同じルールが適用されます。このオペレーションは固定と呼ばれます。固定を検出して最小化する方法については、「RDS Proxy の固定の回避」を参照してください。

RDS Proxy のスタート方法

以下のページの情報を使用して、Amazon RDS Proxy のセットアップと管理を行い、関連するセキュリティオプションを設定します。セキュリティオプションは、各プロキシにアクセスできるユーザーと、各プロキシから DB インスタンスに接続する方法を制御します。

RDS プロキシを初めて使用する場合は、提示されている順序でページに従うことをお勧めします。

トピック

- RDS Proxy のネットワーク前提条件の設定
- RDS Proxy のデータベース認証情報の設定
- RDS Proxy の IAM 認証の設定
- Amazon RDS のプロキシの作成
- プロキシの表示
- RDS Proxy を介したデータベースへの接続

RDS Proxy のネットワーク前提条件の設定

RDS Proxy を使用するには、RDS DB インスタンスと RDS Proxy の間に、共通の仮想プライベートクラウド (VPC) が必要です。この VPC には、異なるアベイラビリティーゾーンにあるサブネットが 2 つ以上必要です。アカウントは、これらのサブネットを所有することも、他のアカウントと共有することもできます。VPC 共有の詳細については、共有 VPC の操作を参照してください。

IPv6 をサポートするには、追加のネットワーク設定が必要です。

- IPv6 エンドポイントネットワークタイプ – VPC とサブネットは IPv6 をサポートするように設定する必要があります。これには、VPC とサブネットに IPv6 CIDR ブロックを割り当てることが含まれます。
- デュアルスタックエンドポイントネットワークタイプ – VPC とサブネットは IPv4 と IPv6 の両方のアドレス指定をサポートしている必要があります。
- IPv6 ターゲット接続ネットワークタイプ – プロキシからの IPv6 接続をサポートするには、データベースをデュアルスタックモードに設定する必要があります。

Amazon EC2、Lambda、Amazon ECS などのクライアントアプリケーションリソースは、プロキシと同じ VPC に置くことができます。または、プロキシとは別の VPC に置くこともできます。RDS DB インスタンス に正常に接続できた場合は、既に必要なネットワークリソースが存在します。

トピック

- サブネット情報の取得
- IP アドレス容量の計画

サブネット情報の取得

プロキシを作成するには、プロキシが動作するサブネットと VPC を指定する必要があります。次の Linux の例は、AWS アカウント が所有する VPC とサブネットを調べる AWS CLI コマンドを示しています。特に、CLI を使用してプロキシを作成するときは、サブネット ID をパラメータとして渡します。

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

次の Linux の例は、特定の RDS DB インスタンスに対応するサブネット ID を決定する AWS CLI コマンドを示しています。DB インスタンスの VPC ID を見つけます。VPC を調べて、そのサブネットを見つけます。次の Linux の例は、その方法を示しています。

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]|[0]|[0]|[Subnets]|[0]|[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]|[0]|[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
```

```
subnet_id_4
subnet_id_5
subnet_id_6
```

IP アドレス容量の計画

RDS Proxy は、登録されている DB インスタンスのサイズと数に基づき、必要に応じて自動的に容量を調整します。特定の操作には、登録されたデータベースのサイズの増加や内部 RDS Proxy メンテナンス操作など、プロキシ容量の追加が必要になる場合もあります。これらのオペレーション中、プロキシは追加の容量をプロビジョニングするため、より多くの IP アドレスを必要とする場合があります。これらの追加アドレスによって、ワークロードに影響を与えずにプロキシを拡張できます。サブネットに空き IP アドレスがない場合、プロキシのスケールアップはできません。そのため、クエリの待ち時間が長くなったり、クライアント接続が失敗したりする可能性があります。サブネットに十分な空きの IP アドレスがないと、RDS はイベント RDS-EVENT-0243 を通じて通知します。このイベントのフィールドの詳細については、「RDS Proxy イベントの使用」を参照してください。

> **Note**
>
> RDS Proxy は、VPC 内の各プロキシで 215 個を超える IP アドレスを使用しません。

DB インスタンスクラスのサイズに基づいて、サブネット内に次の最少数の空き IP アドレスをプロキシ用として予約します。

DB インスタンスクラス	IP アドレスの最小値
db.*.xlarge 以下	10
db.*.2xlarge	15
db.*.4xlarge	25
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

これらの推奨 IP アドレス数は、デフォルトのエンドポイントのみを使用するプロキシの推定値です。エンドポイントまたはリードレプリカを追加したプロキシには、さらに多くの空き IP アドレスが必要になる場合があります。エンドポイントを追加するたびに、追加で 3 つの IP アドレスを予約することをお勧めします。各リードレプリカに、そのリードレプリカのサイズに基づき、表に指定された IP アドレスを追加で予約することをお勧めします。

RDS Proxy のデータベース認証情報の設定

Amazon RDS の RDS Proxy は、AWS Secrets Manager を使用してデータベース認証情報を安全に保存および管理します。アプリケーションに認証情報を埋め込む代わりに、必要な認証の詳細を含む Secrets Manager シークレットにプロキシを関連付けます。RDS DB インスタンスで、プロキシの接続先のデータベースユーザーアカウントごとに個別の Secrets Manager シークレットを作成します。

または、エンドツーエンドの IAM 認証を使用するように RDS Proxy を設定することもできます。これにより、データベース認証情報を Secrets Manager に保存する必要がなくなります。RDS Proxy は、クライアントからプロキシへの接続とプロキシからデータベースへの接続の両方に IAM 認証を使用します。これにより、シークレットやパスワードの管理を必要としない、完全に統合された IAM ベースの認証ソリューションが提供されます。新しい IAM DB ユーザーを追加する方法の詳細については、「IAM 認証を使用したデータベースアカウントの作成」を参照してください。

トピック

- RDS Proxy で使用するシークレットの作成

RDS Proxy で使用するシークレットの作成

プロキシを作成する前に、まずデータベース認証情報を保存するシークレットを少なくとも 1 つ作成する必要があります。

コンソール

シークレットを作成する

1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [Amazon RDS データベースの認証情報] を選択します。

4. ユーザー名とパスワードを入力します。入力する認証情報は、関連付けられた RDS データベースに存在するデータベースユーザーの認証情報と一致する必要があります。RDS Proxy は、これらの認証情報を使用して、アプリケーションに代わってデータベースへの接続を認証および確立します。

   不一致がある場合は、データベースパスワードと一致するようにシークレットを更新できます。シークレットを更新するまで、そのシークレットを使用してプロキシ経由で接続しようとすると失敗しますが、他の有効なシークレットを使用した接続は引き続き機能します。

   > **Note**
   >
   > RDS for SQL Server の場合、RDS Proxy は DB インスタンスの照合設定に関係なく、Secrets Manager で大文字と小文字を区別するシークレットが必要です。アプリケーションで「Admin」や「admin」などの大文字と小文字が異なるユーザー名が許可されている場合は、それぞれに個別のシークレットを作成する必要があります。RDS Proxy は、クライアントとプロキシ間の大文字と小文字を区別しないユーザー名認証をサポートしていません。
   > SQL Server Agent の詳細については、Microsoft SQL Server のドキュメントを参照してください。

5. [データベース] で、シークレットがアクセスする Amazon RDS データベースを選択します。
6. シークレットの他の設定を入力し、[保存] を選択します。詳細な手順については、「AWS Secrets Manager ユーザーガイド」の「Create an AWS Secrets Manager secret」を参照してください。

AWS CLI

AWS CLI を通じてプロキシを作成する場合、対応するシークレットの Amazon リソースネーム (ARN) を指定します。プロキシがアクセスできるすべての DB ユーザーアカウントに対し、同じように操作します。AWS マネジメントコンソール では、シークレットをそのわかりやすい名前を使用して選択します。

- RDS Proxy で使用する Secrets Manager シークレットを作成するには、create-secret コマンドを使用します。

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
```

```
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- Secrets Manager シークレットを暗号化するカスタムキーを作成することもできます。次のコマンドはキーの例を作成します。

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam::account_id:role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
```

```
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
        "Resource":"*"
      }
    ]
}'
```

例えば、以下のコマンドは、2 つのデータベースユーザーの Secrets Manager シークレットを作成します。

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

カスタム AWS KMS キーで暗号化されたシークレットを作成するには、以下のコマンドを使用します。

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

AWS アカウントが所有するシークレットを表示するには、list-secrets コマンドを使用します。

```
aws secretsmanager list-secrets
```

CLI を使用してプロキシを作成する場合、1 つ以上のシークレットの Amazon リソースネーム (ARN) を `--auth` パラメータに渡します。次の例は、AWS アカウントが所有する各シークレットの名前と ARN のみを含むレポートを準備する方法を示しています。この例では、`--output table` バージョン 2 で使用可能な AWS CLI パラメータを使用します。AWS CLI バージョン 1 を使用している場合は、代わりに `--output text` を使用します。

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

シークレットに適切な形式の正しい認証情報が含まれていることを確認するには、get-secret-value コマンドを使用します。*your_secret_name* をシークレットの短縮名または ARN に置き換えます。

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

出力には、次のような JSON でエンコードした値を持つ行が含まれます。

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

RDS Proxy の IAM 認証の設定

Amazon RDS で RDS Proxy の AWS Identity and Access Management (IAM) 認証を設定するには、必要なアクセス許可を付与する IAM ポリシーを作成して設定します。

このトピックでは、必要な IAM ポリシーの作成や IAM ロールへのアタッチなど、RDS Proxy の IAM 認証を設定する手順について説明します。

> **Tip**
>
> この手順は、独自の IAM ロールを作成する場合にのみ必要です。それ以外の場合は、プロキシの設定時に RDS が自動的に必要なロールを作成するため、これらのステップをスキップできます。

前提条件

RDS Proxy の IAM 認証を設定する前に、以下があることを確認してください。

- AWS Secrets Manager – データベース認証情報を含む少なくとも 1 つの保存済みシークレット。シークレットの作成手順については、「the section called “データベース認証情報の設定”」を参照してください。

  これはエンドツーエンドの IAM 認証を使用している場合は必要ありません。
- IAM アクセス許可 – AWS Secrets Manager で IAM ポリシー、ロール、シークレットを作成および管理するためのアクセス許可を持つ IAM ロールまたはユーザー。

エンドツーエンドの IAM 認証用の IAM ポリシーの作成

エンドツーエンドの IAM 認証を使用する場合、RDS Proxy は Secrets Manager から認証情報を取得するのではなく、IAM 認証を使用してデータベースに接続します。これには、プロキシで使用するデータベースアカウントの `rds-db:connect` アクセス許可を持つ IAM ロールを設定する必要があります。

IAM を使用して RDS Proxy をデータベースに対して認証するには、必要なデータベース接続許可を付与するポリシーを持つ IAM ロールを作成します。

コンソール

プロキシを使用してエンドツーエンドの IAM 認証用のロールを作成するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール https://console.aws.amazon.com/iam/ を開きます。
2. ロールに対するアクセス許可ポリシーを作成します。一般的な手順については、「IAM ポリシーを作成する (コンソール)」を参照してください。

   このポリシーを JSON エディタに貼り付け、以下の変更を行います。

   - 自分のアカウント ID に置き換えます。
   - `us-east-2` をプロキシが存在する必要のある場所に置き換えます。
   - データベースリソース ID とユーザー名を、使用するものに置き換えます。リソース ID の形式は、RDS インスタンスと Aurora clusters で異なります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}
```

3. ロールを作成し、アクセス許可ポリシーをそのロールにアタッチします。一般的な手順については、「AWS サービスにアクセス許可を委任するロールを作成する」を参照してください。

   [信頼されたエンティティタイプ] で、[AWS サービス] を選択します。[ユースケース] で、[RDS] を選択し、ユースケースの [RDS – ロールをデータベースに追加する] を選択します。

4. [アクセス許可ポリシー] で、作成したポリシーを選択します。
5. [信頼されたエンティティを選択] で、ロールの次の信頼ポリシーを入力します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

AWS CLI を使用してロールを作成するには、次のリクエストを送信します。

```
aws iam create-role \
  --role-name my_e2e_iam_role_name \
```

```
  --assume-role-policy-document '{"Version":"2012-10-17",
     "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

ポリシーをロールにアタッチします。

```
aws iam put-role-policy \
  --role-name my_e2e_iam_role_name \
  --policy-name e2e_iam_db_connect_policy \
  --policy-document '{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}'
```

エンドツーエンドの IAM 認証用に設定された IAM ロールとアクセス許可を使用して、DefaultAuthScheme を IAM_AUTH に設定してプロキシを作成できるようになりました。このプロキシは、Secrets Manager シークレットを必要とせずに、IAM を使用してデータベースに直接認証します。手順については、「the section called “プロキシの作成”」を参照してください。

エンドツーエンドの IAM 認証を使用する場合は、「IAM 認証を使用したデータベースアカウントの作成」で説明されているように、データベースユーザーが IAM 認証用に設定されていることを確認します。

Secrets Manager アクセス用の IAM ポリシーの作成

RDS Proxy が Secrets Manager からデータベース認証情報を取得できるようにするには、必要なアクセス許可を付与するポリシーを持つ IAM ロールを作成します。

コンソール

プロキシで使用するシークレットにアクセスするためのロールを作成するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール https://console.aws.amazon.com/iam/ を開きます。
2. ロールに対するアクセス許可ポリシーを作成します。一般的な手順については、「IAM ポリシーを作成する (コンソール)」を参照してください。

   このポリシーを JSON エディタに貼り付け、以下の変更を行います。

   - 自分のアカウント ID に置き換えます。
   - `us-east-2` をプロキシが存在するリージョンに置き換えます。
   - シークレット名を、作成したシークレット名に置き換えます。詳細については、「Specifying KMS keys in IAM policy statements」を参照してください。
   - Secrets Manager シークレットの暗号化に使用した KMS キー ID を、デフォルトキーまたは独自のキーに置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_1",
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/key_id",
            "Condition": {
                "StringEquals": {
```

```
                    "kms:ViaService": "secretsmanager.us-
east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. ロールを作成し、アクセス許可ポリシーをそのロールにアタッチします。一般的な手順については、「AWS サービスにアクセス許可を委任するロールを作成する」を参照してください。

   [信頼されたエンティティタイプ] で、[AWS サービス] を選択します。[ユースケース] で、[RDS] を選択し、ユースケースの [RDS – ロールをデータベースに追加する] を選択します。

4. [アクセス許可ポリシー] で、作成したポリシーを選択します。
5. [信頼されたエンティティを選択] で、ロールの次の信頼ポリシーを入力します。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

AWS CLI を使用してロールを作成するには、次のリクエストを送信します。

```
aws iam create-role \
  --role-name my_role_name \
```

```
  --assume-role-policy-document '{"Version": "2012-10-17",
      "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

ポリシーをロールにアタッチします。

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

IAM ロールとアクセス許可を設定したら、プロキシを作成してこのロールに関連付けることができます。これにより、プロキシは AWS Secrets Manager からデータベース認証情報を安全に取得し、アプリケーションの IAM 認証を有効にすることができます。手順については、「the section called “プロキシの作成”」を参照してください。

Amazon RDS のプロキシの作成

プロキシは、RDS for MariaDB、RDS for Microsoft SQL サーバー、RDS for MySQL、RDS for PostgreSQL DB の各インスタンスに関連付けることができます。

コンソール

プロキシを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. [Create proxy (プロキシの作成)] を選択します。
4. プロキシについて次の設定を行います。

設定	説明
エンジンファミリー	データベースとの間で送受信されるネットワークトラフィックを解釈するときに、プロキシが認識するデータベースネットワークプロトコル。 **Note** RDS for PostgreSQL を使用するには、`postgres` データベースを必ずインスタンス内に保持してください。「削除された `postgres` データベースのトラブルシューティング」を参照してください。
プロキシ識別子	AWS アカウント ID と現在の AWS リージョン内で一意の名前。
アイドルクライアントの接続タイムアウト	プロキシは、設定された期間アイドル状態のままである場合、クライアント接続を閉じます。デフォルトでは 1,800 秒 (30 分) です。接続がアイドル状態であるとは、前のリクエストの完了後、新しいリクエストが指定時間内にアプリケーションから送信されない場合のことです。プロキシは、基となるデータベー

設定	説明
	ス接続を開いたままにして接続プールに返すため、その接続を新しいクライアント接続に利用できます。 古い接続を事前に削除するには、アイドルクライアントの接続タイムアウトを短くします。ワークロード急増時の接続コストを最小限に抑えるには、タイムアウトを長くします。
データベース	このプロキシを介してアクセスする RDS DB インスタンス。このリストには、互換性のあるデータベースエンジン、エンジンバージョン、および他の設定を持つ DB インスタンスとクラスターのみが含まれます。リストが空の場合は、RDS Proxy と互換性のある新しい DB インスタンスまたはクラスターを作成します。これを行うには、「Amazon RDS DB インスタンスの作成」の手順に従います。次に、プロキシの作成をもう一度試します。
接続プールの最大接続数	RDS Proxy が使用できる `max_connections` 制限の割合を定義する 1 ~ 100 の値。この DB インスタンスまたはクラスターで 1 つのプロキシのみを使用する場合は、この値を 100 に設定します。RDS Proxy がこの設定を使用する方法の詳細については、「the section called “MaxConnectionsPercent”」を参照してください。

設定	説明
セッション固定フィルタ	検出された特定のセッション状態が RDS Proxy によって固定されるのを防ぎます。これにより、接続を多重化するためのデフォルトの安全対策がバイパスされます。現在、PostgreSQL はこの設定をサポートしておらず、使用可能なオプションは EXCLUDE_VARIABLE_SETS のみです。これを有効にすると、ある接続のセッション変数が他の接続に影響を与える可能性があるため、クエリが現在のトランザクションの外部で設定されたセッション変数に依存する場合は、エラーや正確性の問題が発生します。このオプションは、アプリケーションがデータベース接続を安全に共有できることをあらかじめ確認した上で使用してください。 以下のパターンは安全だと考えられます。 • 有効なセッション変数値に変更がない SET ステートメント。つまり、セッション変数に変更はありません。 • セッション変数の値を変更し、同じトランザクションでステートメントを実行します。 詳細については、「RDS Proxy の固定の回避」を参照してください。
接続借用タイムアウト	プロキシが使用可能なデータベース接続をすべて使用することが予想される場合は、プロキシがタイムアウトエラーを返すまでの待機時間を設定します。指定できる値は最大 5 分です。この設定は、プロキシが接続の最大数に達していて、すべての接続が使用されている場合にのみ適用されます。

設定	説明
初期化クエリ	(オプション) 初期化クエリを追加するか、現在のクエリを変更します。新しい各データベース接続を開くときに実行するプロキシ用の 1 つ以上の SQL ステートメントを指定できます。設定は通常、各接続の設定が同一であることを確認するために、SET ステートメントとともに使用されます。追加するクエリが有効であることを確認します。1 つの SET ステートメントに複数の変数を含めるには、カンマ区切りを使用します。例えば、次のようになります。 `SET variable1 =value1, variable2 =value2` 複数のステートメントの場合は、セミコロンをセパレータとして使用します。 **Important** 初期化クエリは、ターゲットグループ設定の一部としてアクセスできるため、認証や暗号化方法によって保護されません。プロキシターゲットグループ設定を表示または管理するアクセス権を持つユーザーは、初期化クエリを表示できます。このオプションには、パスワードや有効期間の長い暗号化キーなどの機密データを追加しないでください。

設定	説明
AWS Identity and Access Management (IAM) ロール	Secrets Manager シークレットにアクセスする権限を持つ IAM ロール。シークレットは、プロキシが使用できるデータベースユーザーアカウントの認証情報を表します。または、AWS マネジメントコンソールを使用して新しい IAM ロールを作成することもできます。 **Note** アカウントでロールマネージャーが有効になっている場合、Amazon RDS は自動的にロールをアタッチします。また、ここで説明するロールオプションは [カスタマイズ] オプションに置き換えられます。別のロールを使用するには、[カスタマイズ] を選択します。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。
Secrets Manager のシークレット	プロキシを使用できるデータベースユーザーアカウントの認証情報を表す Secrets Manager シークレットを作成または選択します。 [デフォルト認証スキーム] が [なし] に設定されている場合、このフィールドは必須です。[デフォルト認証スキーム] が [IAM 認証] に設定されている場合、このフィールドはオプションになり、コンソールでそのようにマークされます。 ドロップダウンから 1 つ以上のシークレットを選択するか、[新しいシークレットを作成する] リンクを使用して、新しいシークレットを作成できます。

設定	説明
クライアント認証タイプ	プロキシがクライアントからの接続に使用する認証のタイプ。選択した内容は、このプロキシに関連付けるすべての Secrets Manager シークレットに適用されます。シークレットごとに異なるクライアント認証タイプを指定する必要がある場合は、代わりに AWS CLI または API を使用してプロキシを作成します。このオプションは、クライアント接続が認証にデータベース認証情報を使用する場合にのみ指定します。
IAM 認証	プロキシへの接続の IAM 認証に [必須]、[許可済み]、または [許可されていません] を指定します。[許可済み] オプションは、RDS for SQL Server のプロキシに限り有効です。選択した内容は、このプロキシに関連付けるすべての Secrets Manager シークレットに適用されます。シークレットごとに異なる IAM 認証を指定する必要がある場合は、代わりに AWS CLI または API を使用してプロキシを作成します。
デフォルトの認証スキーム	プロキシへのクライアント接続およびプロキシから基になるデータベースへの接続に、プロキシが使用するデフォルトの認証のタイプを選択します。次のオプションがあります。 • [なし] (デフォルト) - プロキシは Secrets Manager シークレットからデータベース認証情報を取得します。 • [IAM 認証] - プロキシは IAM 認証を使用してデータベースに接続し、end-to-end IAM 認証を有効にします。 [IAM 認証] を選択すると、ターゲットグループ設定のデータベースに対して IAM データベース認証を有効にするように促す情報アラートが表示されます。 **Note** このオプションは、MySQL、PostgreSQL、および MariaDB エンジンファミリーでのみサポートされています。

設定	説明
IAM 認証用のデータベースアカウント	このフィールドは、[デフォルト認証スキーム] が [IAM 認証] に設定され、[Identity and Access Management (IAM) ロール] が [IAM ロールを作成] に設定されている場合にのみ表示されます。 IAM 認証で使用するプロキシのデータベースユーザーアカウントに名前を付けます。これは必須のフィールドです。次の方法で複数のアカウントを指定します。 • データベースユーザー名を入力してタグとして追加する • 特定のデータベースユーザー名を使用する (例: `db_user`、`jane_doe`) • 複数のユーザーにワイルドカードパターンを使用する (例: `db_test_*`) 各アカウントは、X アイコンをクリックして削除できるリムーバブルタグとして表示されます。コンソールはこれらの値を使用して、IAM ロールポリシーに適切な `rds-db:connect` アクセス許可を作成します。
Transport Layer Security が必要	すべてのクライアント接続に TLS/SSL を適用します。クライアント接続が暗号化されてるかどうかに関係なく、プロキシは基となるデータベースへの接続に同じ暗号化設定を使用します。

設定	説明
ターゲット接続ネットワークタイプ	プロキシがターゲットデータベースへの接続に使用する IP バージョン。次のオプションから選択します。 • IPv4 – プロキシは IPv4 アドレスを使用してデータベースに接続します。 • IPv6 – プロキシは IPv6 アドレスを使用してデータベースに接続します。 デフォルトは IPv4 です。IPv6 を使用するには、データベースがデュアルスタックモードをサポートしている必要があります。デュアルスタックモードは、ターゲット接続では使用できません。
エンドポイントネットワークタイプ	クライアントがプロキシへの接続に使用するプロキシエンドポイントの IP バージョン。次のオプションから選択します。 • IPv4 – プロキシエンドポイントは IPv4 アドレスのみを使用します。 • IPv6 – プロキシエンドポイントは IPv6 アドレスのみを使用します。 • デュアルスタック – プロキシエンドポイントは IPv4 アドレスと IPv6 アドレスの両方をサポートします。 デフォルトは IPv4 です。IPv6 またはデュアルスタックを使用するには、選択したネットワークタイプをサポートするように VPC とサブネットを設定する必要があります。
サブネット –	このフィールドには、VPC に関連付けられたすべてのサブネットがあらかじめ入力されています。プロキシに不要なサブネットは削除できますが、少なくとも 2 つのサブネットを残しておく必要があります。IPv6 またはデュアルスタックのエンドポイントネットワークタイプの場合は、選択したサブネットが選択したネットワークタイプをサポートしていることを確認します。

設定	説明
VPC セキュリティグループ	既存の VPC セキュリティグループを選択するか、AWS マネジメントコンソールから新しい VPC セキュリティグループを作成します。アプリケーションがプロキシにアクセスできるようにするインバウンドルールと、データベースターゲットからのトラフィックを許可するアウトバウンドルールを設定します。 **Note** セキュリティグループは、プロキシからデータベースへの接続を許可する必要があります。セキュリティグループは、アプリケーションからプロキシへのイングレスとプロキシからデータベースへのエグレスの両方で機能します。たとえば、データベースとプロキシの両方に同じセキュリティグループを使用する場合は、そのセキュリティグループ内のリソースが相互に通信できることを確認してください。 共有 VPC を使用する際には、VPC のデフォルトのセキュリティグループや、別のアカウントに関連付けられたセキュリティグループを使用することは避けてください。代わりに、自分のアカウントに属しているセキュリティグループを選択します。存在しない場合は、作成してください。詳細については、「共有 VPC の操作」を参照してください。 RDS は、高可用性を確保するために複数のアベイラビリティーゾーンにプロキシをデプロイします。クロス AZ 通信を有効にするには、プロキシサブネットのネットワークアクセスコントロールリスト (ACL) で、エンジンポートでのエグレスとすべてのポートでのイングレスを許可する必要があります。ネットワーク ACL の詳細については、「ネットワーク ACL を使用してサブネットへのトラフィックを制御する」を参照してください。プロキシとターゲットのネットワーク ACL が同じである場合は、ソースを VPC CIDR に設定した、TCP プロトコルイングレスルールを追加する必要があります。また、[送信先] を

設定	説明
	VPC CIDR に設定して、エンジンポート固有の TCP プロトコル Egress ルールも追加する必要があります。
拡張ログ記録をアクティブ化	プロキシの互換性やパフォーマンスの問題のトラブルシューティングを行うには、この設定を有効にします。有効にすると、RDS Proxy は、SQL の動作やプロキシ接続のパフォーマンスとスケーラビリティをデバッグするのに役立つ詳細なパフォーマンス情報をログに記録します。 この設定はデバッグの目的でのみ有効にし、ログ内の機密情報を保護するための適切なセキュリティ対策が講じられていることを確認してください。オーバーヘッドを最小限に抑えるため、この設定は有効にしてから 24 時間後に自動的に無効になります。これは、特定の問題のトラブルシューティングを行うために一時的に使用してください。

5. [Create proxy (プロキシの作成)] を選択します。

AWS CLI

AWS CLI を使用してプロキシを作成するには、以下の必須パラメータを指定して create-db-proxy コマンドを呼び出します。

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--vpc-subnet-ids`

`--engine-family` 値では、大文字と小文字が区別されます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-proxy \
    --db-proxy-name proxy_name \
```

```
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--default-auth-scheme { NONE | IAM_AUTH }] \
    [--auth ProxyAuthenticationConfig_JSON_string] \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] \
    [--target-connection-network-type { IPV4 | IPV6 }] \
    [--tags comma_separated_list]
```

Windows の場合:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--default-auth-scheme { NONE | IAM_AUTH }] ^
    [--auth ProxyAuthenticationConfig_JSON_string] ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] ^
    [--target-connection-network-type { IPV4 | IPV6 }] ^
    [--tags comma_separated_list]
```

以下は、--auth オプションの JSON 値の例です。この例では、各シークレットに異なるクライアント認証タイプを適用します。

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },
```

```
  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

`--endpoint-network-type` パラメータは、クライアントがプロキシへの接続に使用するプロキシエンドポイントの IP バージョンを指定します。次の値を指定できます。

- `IPV4` – プロキシエンドポイントは IPv4 アドレスのみを使用します (デフォルト)。
- `IPV6` – プロキシエンドポイントは IPv6 アドレスのみを使用します。
- `DUAL` – プロキシエンドポイントは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。

`--target-connection-network-type` パラメータは、プロキシがターゲットデータベースへの接続に使用する IP バージョンを指定します。次の値を指定できます。

- `IPV4` – プロキシは IPv4 アドレス (デフォルト) を使用してデータベースに接続します。
- `IPV6` – プロキシは IPv6 アドレスを使用してデータベースに接続します。

IPv6 またはデュアルスタックのエンドポイントネットワークタイプを使用するには、選択したネットワークタイプをサポートするように VPC とサブネットを設定する必要があります。IPv6 ターゲット接続ネットワークタイプを使用するには、データベースがデュアルスタックモードをサポートしている必要があります。

> **Tip**
>
> `--vpc-subnet-ids` パラメータに使用するサブネット ID がまだわからない場合は、「RDS Proxy のネットワーク前提条件の設定」を使用して、ID を検索する方法の例を参照してください。

> **Note**
>
> セキュリティグループは、プロキシの接続先のデータベースへのアクセスを許可する必要があります。同じセキュリティグループが、アプリケーションからプロキシへのイングレスと、プロキシからデータベースへのエグレスに使用されます。例えば、データベースとプロキシに同じセキュリティグループを使用するとします。この場合は必ず、そのセキュリティグループ内のリソースが同じセキュリティグループ内の他のリソースと通信できるように指定してください。
>
> 共有 VPC を使用する場合、VPC のデフォルトのセキュリティグループや、別のアカウントに属しているセキュリティグループを使用することはできません。自分のアカウントに属しているセキュリティグループを選択します。存在しない場合は、作成します。この制限事項の詳細については、共有 VPC の操作を参照してください。

プロキシに適切な関連付けを作成するには、register-db-proxy-targets コマンドも使用します。ターゲットグループ名 `default` を指定します。RDS Proxy は、各プロキシを作成するときに、この名前のターゲットグループを自動的に作成します。

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

RDS API

RDS Proxy を作成するには、Amazon RDS API オペレーション CreateDBProxy を呼び出します。AuthConfig データ構造でパラメータを渡します。

RDS Proxy は、各プロキシを作成するときに、`default` という名前のターゲットグループを自動的に作成します。RegisterDBProxyTargets 関数を呼び出して、このターゲットグループに RDS DB インスタンスを関連付けます。

> **Important**
>
> デフォルトの認証スキームに [IAM 認証] を選択する場合。
>
> - プロキシが正常に接続する前に、ターゲットデータベースインスタンスまたはクラスターで IAM データベース認証を有効にする必要があります。
> - [IAM ロールを作成] を選択した場合、[IAM 認証用のデータベースアカウント] が必要です。
> - 既存の IAM ロールを選択した場合、コンソールはデータベース接続許可でロールを自動的に更新しません。ロールに必要な `rds-db:connect` アクセス許可があることを確認します。

プロキシの表示

1 つ以上の RDS プロキシを作成したら、AWS マネジメントコンソール、AWS CLI、または RDS API でそれらを表示および管理できます。設定の詳細の確認、パフォーマンスのモニタリング、必要に応じて変更または削除するプロキシの決定が可能です。

データベースアプリケーションがプロキシを介してトラフィックをルーティングできるようにするには、接続文字列にプロキシエンドポイントを指定する必要があります。

コンソール

コンソールでプロキシを表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. プロキシ名を選択して詳細を表示します。
4. 詳細ページの [ターゲットグループ] セクションに、プロキシが特定の RDS DB インスタンスにどのようにリンクされているかが表示されます。デフォルトのターゲットグループページに移動すると、プロキシの作成時に定義された設定など、この関連付けの詳細を表示できます。この設定には、最大接続数の割合 (%)、接続借用タイムアウト、エンジンファミリー、セッション固定フィルタが含まれます。

CLI

CLI を使用してプロキシを表示するには、describe-db-proxies コマンドを使用します。デフォルトでは、このリクエストは AWS アカウントが所有するすべてのプロキシを返します。単一のプロキシの詳細を表示するには、 `--db-proxy-name` パラメータで名前を指定します。

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

プロキシに関連付けられた他の情報を表示するには、以下のコマンドを使用します。

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

プロキシに関連付けられている情報の詳細を表示するには、次のコマンドのシーケンスを使用します。

1. プロキシのリストを取得するには、describe-db-proxies を実行します。
2. プロキシが使用できる接続の最大割合などの接続パラメータを表示するには、describe-db-proxy-target-groups `--db-proxy-name` を実行します。プロキシの名前をパラメータ値として使用します。
3. 返されたターゲットグループに関連付けられている RDS DB インスタンスの詳細を表示するには、describe-db-proxy-targets を実行します。

RDS API

RDS API を使用してプロキシを表示するには、DescribeDBProxies オペレーションを使用します。DBProxy データ型の値が返されます。

プロキシの接続設定の詳細を表示するには、この戻り値のプロキシ識別子を DescribeDBProxyTargetGroups オペレーションで使用します。DBProxyTargetGroup データ型の値が返されます。

プロキシに関連付けられている RDS インスタンスや Aurora DB クラスターを表示するには、DescribeDBProxyTargets オペレーションを使用します。DBProxyTarget データ型の値が返されます。

RDS Proxy を介したデータベースへの接続

プロキシを介して、またはデータベースに接続することによって RDS DB インスタンスに接続する方法は、一般に同じです。詳細については、「プロキシエンドポイントの概要」を参照してください。

トピック

- データベース認証情報を使用したデータベースへの接続
- IAM 認証を使用したデータベースへの接続
- Microsoft SQL Server への接続に関する考慮事項
- PostgreSQL への接続に関する考慮事項

データベース認証情報を使用したデータベースへの接続

データベース認証情報を使用してプロキシに接続するには、次の手順に従います。

1. プロキシエンドポイントを見つけます。AWS マネジメントコンソール では、プロキシの詳細ページで対応するエンドポイントを見つけることができます。AWS CLI では、describe-db-proxies コマンドを使用できます。以下の例のように指定します。

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
```

```
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. クライアントアプリケーションの接続文字列で、そのエンドポイントをホストパラメータとして指定します。例えば、`mysql -h` オプションまたは `psql -h` オプションの値としてプロキシエンドポイントを指定します。
3. 通常と同じようにデータベースのユーザー名とパスワードを指定します。

IAM 認証を使用したデータベースへの接続

RDS Proxy で IAM 認証を使用する場合、クライアントとプロキシ間の認証には 2 つのオプションがあります。

- 通常のユーザー名とパスワードで認証するようにデータベースユーザーを設定します。RDS Proxy は、Secrets Manager からユーザー名とパスワードの認証情報を取得します。RDS Proxy から基礎となるデータベースへの接続は IAM を経由しません。
- また、エンドツーエンドの IAM 認証を使用することもできます。これは、データベース認証情報を必要とせずに IAM を使用してプロキシを介してデータベースに接続します。

IAM 認証を使用して RDS Proxy に接続するには、RDS DB インスタンスとの IAM 認証と同じ一般的な接続手順で行います。IAM の使用に関する一般情報については、「Amazon RDS でのセキュリティ」を参照してください。エンドツーエンドの IAM 認証を使用している場合は、DB ユーザーに IAM 認証プラグインを提供します。「IAM 認証を使用したデータベースアカウントの作成」を参照してください。

RDS Proxy での IAM の使用方法の主な違いは以下のとおりです。

- 標準の IAM 認証では、データベースユーザーはデータベース内で通常の認証情報を持ちます。これらのユーザー名とパスワードを含む Secrets Manager シークレットを設定し、RDS Proxy に Secrets Manager からの認証情報の取得を認可します。IAM 認証がクライアントプログラムとプロキシ間の接続に適用されます。その後、プロキシは、Secrets Manager から取得したユーザー名とパスワードの認証情報を使用して、データベースに対して認証します。
- エンドツーエンドの IAM 認証では、データベース認証情報用に Secrets Manager シークレットを設定する必要はありません。IAM 認証は、クライアントからプロキシへの接続とプロキシからデータベースへの接続に適用されます。

- インスタンス、クラスター、またはリーダーの各エンドポイントの代わりに、プロキシエンドポイントを指定します。プロキシエンドポイントの詳細については、「IAM 認証を使用した DB インスタンスへの接続」を参照してください。
- IAM 認証を使用してプロキシに接続する場合は、Transport Layer Security (TLS) / Secure Sockets Layer (SSL) を使用していることを確認してください。

IAM ポリシーを変更することで、特定のユーザーにプロキシへのアクセスを許可できます。以下に例を示します。

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

> **Tip**
>
> RDS Proxy 接続の IAM 認証を設定するときは、接続の問題を回避するため、以下の重要なガイドラインに従ってください。
>
> - 同じデータベースユーザーまたは `rds_iam` ロールの一般的なパスワード認証を維持しながら、ロールを許可しないでください。
> - クライアントが IAM 認証を使用して RDS Proxy に接続する間、RDS Proxy は常に Secrets Manager を介したパスワード認証を使用してデータベースに接続することに注意してください。
> - 接続の終了や再接続が頻繁に発生する場合は、ユーザーまたはロールから既存の `rds_iam` 許可をすべて削除し、パスワード認証のみを使用します。
> - パスワードポリシーが SCRAM-SHA-256 の安全な文字要件を満たしていることを確認します。
>
> 同じデータベースユーザーの IAM 認証方法とパスワード認証方法が混在すると、接続が不安定になる可能性があります。

Microsoft SQL Server への接続に関する考慮事項

IAM 認証を使用してプロキシに接続する場合、パスワードフィールドは使用しません。代わりに、データベースドライバーの種類ごとに適切なトークンプロパティをトークンフィールドに指定します。例えば、JDBC の `accessToken` プロパティや ODBC の `sql_copt_ss_access_token` プロ

パティを使用します。または、.NET SqlClient ドライバーの AccessToken プロパティを使用します。トークンプロパティをサポートしていないクライアントでは IAM 認証を使用できません。

プロキシがデータベース接続を共有できない条件もあるため、代わりにクライアントアプリケーションからプロキシへの接続を専用のデータベース接続に固定します。これらの条件の詳細については、「RDS Proxy の固定の回避」を参照してください。

PostgreSQL への接続に関する考慮事項

RDS Proxy に接続するための新しい PostgreSQL データベースユーザーを作成する場合は、データベースに対するユーザー CONNECT 権限を付与してください。これがないと、ユーザーは接続を確立できません。詳細については、「the section called “RDS Proxy を使用する場合に PostgreSQL データベースに新しいデータベースユーザーを追加する”」を参照してください。

クライアントが PostgreSQL データベースへの接続をスタートする際は起動メッセージを送信します。このメッセージには、パラメータ名と値の文字列のペアが含まれています。詳細については、PostgreSQL ドキュメントの「PostgreSQL Message Formats」で StartupMessage を参照してください。

RDS Proxy を介して接続する場合、起動メッセージには、現在認識されている以下のパラメータを含めることができます。

- user
- database

起動メッセージには、以下の追加のランタイムパラメータを含めることもできます。

- application_name
- client_encoding
- DateStyle
- TimeZone
- extra_float_digits
- search_path

PostgreSQL メッセージの詳細については、PostgreSQL ドキュメントの「Frontend/Backend Protocol」を参照してください。

PostgreSQL では、JDBC を使用する場合、ピニングを回避するために以下の操作をお勧めします。

- 固定を回避するために、JDBC 接続パラメータ `assumeMinServerVersion` を `9.0` 以上に設定します。これにより、JDBC ドライバーが `SET extra_float_digits = 3` を実行するとき、接続の始動時に余分なラウンドトリップを実行しなくなります。
- 固定を回避するために、JDBC 接続パラメータ `ApplicationName` を *`any/your-application-name`* に設定します。これを行うと、JDBC ドライバーが `SET application_name = "PostgreSQL JDBC Driver"` を実行するとき、接続のスタートアップ中に余分なラウンドトリップを実行しなくなります。JDBC パラメータは `ApplicationName` ですが、PostgreSQL `StartupMessage` パラメータは `application_name` です。

詳細については、「RDS Proxy の固定の回避」を参照してください。JDBC を使用した接続の詳細については、PostgreSQL ドキュメントの「Connecting to the Database」を参照してください。

RDS Proxy の管理

このセクションでは、RDS Proxy の操作と設定を管理する方法について説明します。以下の手順は、アプリケーションがデータベース接続を最も効率的に使用し、接続を最大限に再利用するのに役立ちます。接続の再利用率を高めるほど、CPU とメモリのオーバーヘッドを減らすことができます。これにより、アプリケーションのレイテンシーを減らし、データベースのより多くのリソースをアプリケーションからのリクエストの処理に集中させることができます。

トピック

- RDS Proxy の変更
- RDS Proxy の使用時に新しいデータベースユーザーを追加する
- RDS Proxy の標準 IAM 認証からエンドツーエンドの IAM 認証への移行
- RDS Proxy 接続に関する考慮事項
- RDS Proxy の固定の回避
- RDS Proxy の削除

RDS Proxy の変更

プロキシの作成後に、プロキシに関連付けられた固有の設定を変更できます。これを行うには、プロキシ自体、プロキシに関連付けられているターゲットグループ、またはその両方を変更します。各プロキシには、ターゲットグループが関連付けられています。

AWS マネジメントコンソール

> **Important**
>
> クライアント認証タイプと IAM 認証フィールドの値は、このプロキシに関連付けられているすべての Secrets Manager シークレットに適用されます。シークレットごとに異なる値を指定するには、代わりに AWS CLI または API を使用してプロキシを変更します。

プロキシの設定を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. プロキシのリストで、設定を変更するプロキシを選択するか、その詳細ページに移動します。
4. [アクション]、[変更] の順に選択します。
5. 変更するプロパティを入力または選択します。次を変更できます。

   - プロキシ識別子 - プロキシの名前を変更するには、新しい識別子を入力します。
   - アイドル状態のクライアント接続のタイムアウト - アイドル状態のクライアント接続のタイムアウトの時間を入力します。
   - IAM ロール - Secrets Manager からシークレットを取得するために使用する IAM ロールを変更します。

     > **Note**
     >
     > [デフォルト認証スキーム] を [IAM 認証] に設定した場合、新しい IAM ロールを作成することはできません。

   - Secrets Manager シークレット - Secrets Manager シークレットを追加または削除します。これらのシークレットは、データベースのユーザー名とパスワードに対応します。
   - クライアント認証タイプ – プロキシへのクライアント接続の認証タイプを変更します。
   - IAM 認証 - プロキシへの接続に IAM 認証をすることを要求または禁止します。
   - デフォルト認証スキーム – プロキシへのクライアント接続とプロキシから基になるデータベースへの接続にプロキシが使用するデフォルトの認証スキームを変更します。

- Transport Layer Security が必要 - Transport Layer Security (TLS) の要件を有効または無効にします。
- VPC セキュリティグループ - プロキシで使用する VPC セキュリティグループを追加または削除します。
- 拡張されたログ記録を有効にする - 拡張ログ記録を有効または無効にします。

6. [変更] を選択します。

変更する設定がリストにない場合は、以下の手順を使用して、プロキシのターゲットグループを更新します。プロキシに関連付けられているターゲットグループは、物理データベース接続に関連する設定を制御します。プロキシごとに `default` という名前の 1 つのターゲットグループが関連付けられています。このターゲットグループはプロキシと共に自動的に作成されます。デフォルトのターゲットグループの名前を変更することはできません。

ターゲットグループは、プロキシの詳細ページからのみ変更できます。[プロキシ] ページのリストから変更することはできません。

プロキシのターゲットグループの設定を変更するには

1. [プロキシ] ページから、プロキシの詳細ページに移動します。
2. [ターゲットグループ] で、`default` リンクを選択します。現在、すべてのプロキシには `default` という名前のターゲットグループが 1 つ あります。
3. [デフォルト] ターゲットグループの詳細ページで、[変更] を選択します。
4. 変更できるプロパティに対して新しい設定を選択します。

   - データベース - 別の RDS DB インスタンスまたはクラスター を選択します。
   - 接続プールの最大接続数 - プロキシで使用できる最大接続数の割合 (%) を調整できます。
   - セッション固定フィルター - (オプション) セッション固定フィルタを選択します。これにより、クライアント接続間でデータベース接続を多重化するというデフォルトの安全対策が回避されます。現在、設定は PostgreSQL についてはサポートされていません。唯一の選択肢は `EXCLUDE_VARIABLE_SETS` です。

     この設定を有効にすると、ある接続のセッション変数が他の接続に影響を与える可能性があります。これにより、クエリが現在のトランザクション以外に設定されたセッション変数値に依存している場合、エラーや正確性の問題が発生する可能性があります。アプリケーションがクライアント接続間でデータベース接続を共有しても安全であることを確認した後に、このオプションの使用を検討してください。

以下のパターンは安全だと考えられます。

- 有効なセッション変数値に変更がない、つまりセッション変数に変更がない SET ステートメント。
- セッション変数の値を変更し、同じトランザクションでステートメントを実行します。

詳細については、「RDS Proxy の固定の回避」を参照してください。

- 接続借用タイムアウト - 接続借用タイムアウト間隔を調整します。この設定は、プロキシで最大数の接続が既に使用されている場合に適用されます。この設定により、プロキシがタイムアウトエラーを返す前に、接続が使用可能になるまで待つ時間が決まります。
- 初期化クエリ。(オプション) 初期化クエリを追加するか、現在のクエリを変更します。新しい各データベース接続を開くときに実行するプロキシ用の 1 つ以上の SQL ステートメントを指定できます。設定は通常、各接続の設定が同一であることを確認するために、SET ステートメントとともに使用されます。追加するクエリが有効であることを確認します。1 つの SET ステートメントに複数の変数を含めるには、カンマ区切りを使用します。例:

```
SET variable1=value1, variable2=value2
```

複数のステートメントの場合は、セミコロンをセパレータとして使用します。

ターゲットグループ識別子やデータベースエンジンなどの特定のプロパティは変更できません。

5. [Modify target group (ターゲットグループの変更)] を選択します。

AWS CLI

AWS CLI を使用してプロキシを変更するには、modify-db-proxy、modify-db-proxy-target-group、deregister-db-proxy-targets、register-db-proxy-targets の各コマンドを使用します。

`modify-db-proxy` コマンドを使用すると、次のようなプロパティを変更できます。

- プロキシで使用する一連の Secrets Manager シークレット。
- TLS が必要かどうか。
- アイドルクライアントのタイムアウト。
- デバッグ用に SQL ステートメントからの追加情報をログに記録するかどうか。
- Secrets Manager シークレットの取得に使用する IAM ロール。
- プロキシで使用するセキュリティグループ。

- プロキシに関連付けられたデフォルトの認証スキーム。

次の例は、既存のプロキシの名前を変更する方法を示しています。

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

接続関連の設定を変更したり、ターゲットグループの名前を変更したりするには、modify-db-proxy-target-group コマンドを使用します。現在、すべてのプロキシには default という名前のターゲットグループが 1 つ あります。このターゲットグループを使用する場合、プロキシの名前とターゲットグループ名 (default) を指定します。デフォルトのターゲットグループの名前を変更することはできません。

次の例は、初期にプロキシの MaxIdleConnectionsPercent 設定をチェックし、次にターゲットグループを使用して設定を変更する方法を示しています。

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
                "ConnectionBorrowTimeout": 120,
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
```

```
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

deregister-db-proxy-targets コマンドと register-db-proxy-targets コマンドでは、ターゲットグループを通じてプロキシが関連付けられている RDS DB インスタンス を変更します。現在、各プロキシが接続できる RDS DB インスタンスは 1 つです。ターゲットグループは、マルチ AZ 設定のすべての RDS DB インスタンスの接続の詳細を追跡します。

次の例では、cluster-56-2020-02-25-1399 という名前の Aurora MySQL クラスターに関連付けられているプロキシを初期に使用します。次に、このプロキシを変更して provisioned-cluster という名前の別のクラスターに接続できるようにします。

RDS DB インスタンスを使用する場合は、--db-instance-identifier オプションを指定します。

次の例では、Aurora MySQL プロキシを変更します。Aurora PostgreSQL プロキシにはポート 5432 があります。

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
```

```
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}

aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
```

```
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

RDS API

RDS API を使用してプロキシを変更するには、ModifyDBProxy、ModifyDBProxyTargetGroup、DeregisterDBProxyTargets、RegisterDBProxyTargets の各オペレーションを使用します。

`ModifyDBProxy` では、次のようなプロパティを変更できます。

- プロキシで使用する一連の Secrets Manager シークレット。
- TLS が必要かどうか。
- アイドルクライアントのタイムアウト。
- デバッグ用に SQL ステートメントからの追加情報をログに記録するかどうか。
- Secrets Manager シークレットの取得に使用する IAM ロール。
- プロキシで使用するセキュリティグループ。

`ModifyDBProxyTargetGroup` では、接続関連の設定を変更できます。現在、すべてのプロキシには `default` という名前のターゲットグループが 1 つ あります。このターゲットグループを使用する場合、プロキシの名前とターゲットグループ名 (`default`) を指定します。デフォルトのターゲットグループの名前を変更することはできません。

`DeregisterDBProxyTargets` および `RegisterDBProxyTargets` では、ターゲットグループを通じてプロキシが関連付けられる RDS DB インスタンスを変更します。現在、各プロキシが接続できる RDS DB インスタンス は 1 つです。ターゲットグループは、マルチ AZ 設定の RDS DB インスタンス の接続の詳細を追跡します。

RDS Proxy の使用時に新しいデータベースユーザーを追加する

状況に応じて、プロキシに関連付けられている RDS DB インスタンス に新しいデータベースユーザーを追加できます。Secrets Manager シークレットで標準認証を使用しているか、エンドツーエンドの IAM 認証を使用しているかに応じて続行します。

標準の IAM 認証を使用している場合は、以下の手順に従います。

1. 「RDS Proxy のデータベース認証情報の設定」で説明している手順を使用して、新しい Secrets Manager シークレットを作成します。
2. IAM ロールを更新して、RDS Proxy に新しい Secrets Manager シークレットへのアクセスを許可します。そのためには、IAM ロールポリシーのリソースセクションを更新します。
3. RDS Proxy を変更して、[Secrets Manager のシークレット] に新しい Secrets Manager のシークレットを追加します。
4. 既存のユーザーを新しいユーザーに置き換える場合は、プロキシで既存のユーザーの Secrets Manager シークレットに保存されている認証情報を更新します。

エンドツーエンドの IAM 認証を使用している場合は、データベースユーザーを作成し、IAM アクセス許可を設定する必要があります。これを行うには、次のステップを実行します。

1. 認証に使用する IAM ユーザーまたはロール名と一致する新しいデータベースユーザーをデータベースに作成します。
2. データベースユーザーがデータベースで IAM 認証プラグインを使用して設定されていることを確認します。「IAM 認証を使用したデータベースアカウントの作成」を参照してください。
3. 「エンドツーエンドの IAM 認証用の IAM ポリシーの作成」で説明されているように、IAM ユーザーまたはロールに `rds-db:connect` アクセス許可を付与するように IAM ポリシーを更新します。
4. デフォルトの認証スキームとして IAM 認証を使用するようにプロキシが設定されていることを確認します。

エンドツーエンドの IAM 認証では、クライアントからプロキシ、プロキシからデータベースへの認証に IAM 認証情報が使用されるため、Secrets Manager シークレットでデータベース認証情報を管理する必要はありません。

RDS Proxy を使用する場合に PostgreSQL データベースに新しいデータベースユーザーを追加する

PostgreSQL データベースに新しいユーザーを追加するとき、次のコマンドを実行する必要がある場合は、次のコマンドを実行します。

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

ユーザーがターゲットデータベース上の接続をモニタリングできるように、rdsproxyadmin ユーザーに CONNECT 権限を付与します。

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

上記のコマンドで rdsproxyadmin をデータベースユーザーに変更することで、他のターゲットデータベースユーザーにヘルスチェックの実行を許可することもできます。

RDS Proxy を使用する場合のデータベースユーザーのパスワードの変更

状況に応じて、プロキシに関連付けられている RDS DB インスタンス のデータベースユーザーのパスワードを変更できます。その場合は、対応する Secrets Manager シークレットを新しいパスワードで更新します。

エンドツーエンドの IAM 認証を使用している場合は、Secrets Manager シークレットのパスワードを更新する必要はありません。

RDS Proxy の標準 IAM 認証からエンドツーエンドの IAM 認証への移行

現在 RDS Proxy の標準 IAM 認証を使用している場合、クライアントは IAM を使用してプロキシを認証し、プロキシはシークレットを使用してデータベースに接続するため、クライアントからプロキシへの接続とプロキシからデータベースへの接続の両方が IAM 認証を使用するエンドツーエンドの IAM 認証に移行できます。

エンドツーエンド IAM 認証に移行するには

1. RDS Proxy IAM ロールのアクセス許可を更新する

   Secrets Manager と rds:db-connect アクセス許可の両方を含む更新されたプロキシアクセス許可ポリシーを作成します。

   ```
   # Create updated proxy permission policy
   ```

```
cat > updated-proxy-policy.json # EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetSecretsValue",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-1234f"
      ]
    },
    {
      "Sid": "RdsDBConnect",
      "Action": [
        "rds-db:connect"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:rds-db:us-east-1:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe"
      ]
    }
  ]
}
```

ロールポリシーのプロキシを更新します。

```
aws iam put-role-policy \
            --role-name RDSProxyRole \
            --policy-name UpdatedProxyPermissions \
            --policy-document file://updated-proxy-policy.json
```

2. RDS Proxy を変更してエンドツーエンドの IAM 認証を有効にする

```
aws rds modify-db-proxy \
  --db-proxy-name my-database-proxy \
  --default-auth-scheme IAM_AUTH \
  --region us-east-1
```

移行中のダウンタイムがゼロであることを確認する前に、RDS Proxy のステータスが [利用可能] で、DefaultAuthScheme が IAM_AUTH であることを確認します。

```
aws rds describe-db-proxies --db-proxy-name my-database-proxy --region us-east-1
```

正常な出力:

```
{
  "DBProxies": [
    {
      "DBProxyName": "my-database-proxy",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123456789abcdef",
      "Status": "available",
      ...
      "DefaultAuthScheme": "IAM_AUTH"
    }
  ]
}
```

3. データベースで IAM 認証を有効にする

```
aws rds modify-db-cluster \
  --db-cluster-identifier my-database-cluster \
  --enable-iam-database-authentication \
  --region us-east-1
```

4. IAM 認証用のデータベースユーザーを設定する

RDS for PostgreSQL の場合:

```
GRANT rds_iam TO jane_doe;
```

RDS for MariaDB および RDS for MySQL の場合:

```
ALTER USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

5. クライアントアプリケーションコードを変更する必要はありません。接続プロセスは同じままです。

RDS for PostgreSQL の場合:

```
# Generate authentication token
export PGPASSWORD=$(aws rds generate-db-auth-token \
  --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 \
  --port 5432 \
  --username jane_doe \
  --region us-east-1)

# Connect to database through proxy
psql "host=my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 port=5432 user=jane_doe dbname=postgres password=$PGPASSWORD sslmode=require
 sslrootcert=us-east-1-bundle.pem"
```

RDS for MariaDB および RDS for MySQL の場合:

```
# Generate authentication token
export MYSQL_PWD=$(aws rds generate-db-auth-token \
  --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 \
  --port 3306 \
  --username jane_doe \
  --region us-east-1)

# Connect to database through proxy
mysql -h my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com \
  -P 3306 \
  -u jane_doe \
  --ssl-ca=us-east-1-bundle.pem \
  --enable-cleartext-plugin
```

RDS Proxy 接続に関する考慮事項

接続設定の構成

RDS Proxy の接続プーリングを調整するには、以下の設定を変更します。

- IdleClientTimeout
- MaxConnectionsPercent

- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

プロキシがクライアント接続を閉じるまでの間、接続がアイドル状態を継続できる時間を指定できます。デフォルトは 1,800 秒 (30 分) です。

クライアント接続がアイドル状態と見なされるのは、前のリクエストの完了後、新しいリクエストが指定時間内にアプリケーションから送信されない場合です。基となるデータベース接続は開いたままで、接続プールに返されるため、新しいクライアント接続で再利用できます。プロキシで古い接続を事前に削除する場合は、クライアント接続でのアイドル状態のタイムアウトを短くすることを検討してください。ワークロードが頻繁にプロキシとの接続を確立する場合には、接続コストを節約するために、クライアント接続でのアイドル状態のタイムアウトを長くすることを検討してください。

この設定は、RDS コンソール内の [Idle client connection timeout] (アイドルクライアントの接続タイムアウト) フィールドと、AWS CLI および API の `IdleClientTimeout` 設定で行います。RDS コンソールで [Idle client connection timeout] (アイドルクライアントの接続タイムアウト) フィールドの値を変更する方法については、「AWS マネジメントコンソール」を参照してください。`IdleClientTimeout` 設定の値を変更するには、CLI コマンドの modify-db-proxy または API の ModifyDBProxy オペレーションを参照してください。

MaxConnectionsPercent

RDS Proxy がターゲットデータベースに対して確立できる接続の数を制限できます。上限は、データベースで使用可能な最大接続数に対する割合 (%) で指定します。この設定には、RDS Proxy コンソールの [接続プールの最大接続数] フィールド、または AWS CLI と API の `MaxConnectionsPercent` パラメータを使用します。

`MaxConnectionsPercent` 値はターゲットグループが使用する RDS DB インスタンス の `max_connections` 設定に対するパーセンテージで表されます。プロキシは、これらの接続のすべてを事前に作成するわけではありません。この設定では、ワークロードが必要とするときに、プロキシがこれらの接続を確立できます。

例えば、登録済みのデータベースターゲットの `max_connections` が 1000 に設定され、`MaxConnectionsPercent` が 95 に設定されている場合、RDS Proxy はそのデータベースターゲットへの同時接続の上限として 950 接続を設定します。

ワークロードが許容されるデータベース接続の最大数に達したときによく見られる副作用として、全体的なクエリ待ち時間が増加し、`DatabaseConnectionsBorrowLatency` メトリクスも増加しま

す。DatabaseConnections メトリクスと MaxDatabaseConnectionsAllowed メトリクスを比較することで、現在使用中のデータベース接続数と許可されているデータベース接続の合計数を監視できます。

このパラメータを設定する場合は、次のベストプラクティスに注意してください。

- ワークロードパターンの変化に備えて、接続に十分な余裕を持たせてください。このパラメータは、最近監視した最大使用量より少なくとも 30% 高く設定することをお勧めします。RDS Proxy はデータベース接続クォータを複数のノードに再配分するため、内部容量の変更に伴い、借用レイテンシーの増加を避けるために、少なくとも 30% の余裕を持たせて接続を追加することが必要になる場合があります。
- RDS Proxy は、高速フェイルオーバー、トラフィックルーティング、内部オペレーションをサポートするために、アクティブモニタリング用に一定数の接続を予約します。MaxDatabaseConnectionsAllowed メトリクスには、これらの予約済み接続は含まれません。これはワークロードの処理に使用できる接続の数を表し、MaxConnectionsPercent 設定から算出された値よりも小さい場合があります。

  MaxConnectionsPercent の最小推奨値

  - db.t3.small: 30
  - db.t3.medium またはそれ以上: 20

RDS コンソールの [Connection pool maximum connections] (接続プールの最大接続数) フィールドの値を変更する方法については、「AWS マネジメントコンソール」を参照してください。MaxConnectionsPercent 設定での値の変更については、CLI コマンドの「modify-db-proxy-target-group」、または API オペレーションの「ModifyDBProxyTargetGroup」を参照してください。

データベース接続での上限の詳細については、「データベース接続の最大数」を参照してください。

MaxIdleConnectionsPercent

RDS Proxy が接続プール内にアイドル状態で保持できる、データベース接続の数を制御できます。デフォルトでは、RDS Proxy は、接続に対するアクティビティが 5 分間なかった場合に、プール内のデータベース接続をアイドル状態とみなします。

MaxIdleConnectionsPercent 値は、RDS DB インスタンスターゲットグループの max_connections 設定のパーセンテージとして表されます。そのデフォルト値は MaxConnectionsPercent の 50% で、上限は MaxConnectionsPercent の値で指定します。例

えば、MaxConnectionsPercent が 80 の場合、MaxIdleConnectionsPercent のデフォルト値は 40 です。MaxConnectionsPercent の値が指定されていない場合、RDS for SQL Server では MaxIdleConnectionsPercent は 5 であり、その他すべてのエンジンのデフォルト値は 50 です。

値を大きくすると、プロキシではアイドル状態のデータベース接続の大部分を開いたままにします。値を小さくすると、プロキシではアイドル状態のデータベース接続の大部分を閉じます。ワークロードが予測できない場合は、MaxIdleConnectionsPercent には大きな値を設定するように検討してください。これにより、RDS Proxy では多数の新しいデータベース接続を開くことなく、アクティビティの急増に対応できるようになります。

この設定には、AWS CLI と API における DBProxyTargetGroup の MaxIdleConnectionsPercent 設定を使用します。MaxIdleConnectionsPercent 設定での値の変更については、CLI コマンドの「modify-db-proxy-target-group」、または API オペレーションの「ModifyDBProxyTargetGroup」を参照してください。

データベース接続での上限の詳細については、「データベース接続の最大数」を参照してください。

ConnectionBorrowTimeout

RDS Proxy がタイムアウトエラーを返す前に、接続プール内のデータベース接続が使用可能になるまで待つ時間を指定できます。デフォルト値は 120 秒です。この設定値は、接続数が最大値に達し、接続プールで利用可能な接続がなくなった場合に適用されます。また、例えば、フェイルオーバー操作が進行中であるなどの理由で、リクエストを処理するために使用できる適切なデータベースインスタンスがない場合にも適用されます。この設定を使用すると、アプリケーションコードでクエリタイムアウトを変更しなくても、アプリケーションに最適な待機期間を設定できます。

この設定には、RDS Proxy コンソールの [接続借用タイムアウト] フィールド、または AWS CLI と API における DBProxyTargetGroup の ConnectionBorrowTimeout 設定を使用します。RDS コンソールの [Connection borrow timeout] (接続借用タイムアウト) フィールドの値を変更する方法については、「AWS マネジメントコンソール」を参照してください。ConnectionBorrowTimeout 設定での値の変更については、CLI コマンドの「modify-db-proxy-target-group」、または API オペレーションの「ModifyDBProxyTargetGroup」を参照してください。

クライアント接続とデータベース接続

アプリケーションから RDS Proxy への接続は、クライアント接続と呼ばれます。プロキシからデータベースへの接続はデータベース接続です。RDS Proxy を使用する場合、クライアント接続はプロキシで終了し、データベース接続は RDS Proxy 内で管理されます。

アプリケーション側の接続プーリングは、アプリケーションと RDS Proxy 間で繰り返し接続を確立する回数が減るというメリットを提供します。

アプリケーション側の接続プールを実装する前に、以下の設定について考慮してください。

- クライアント接続の最大有効期間: RDS Proxy では、クライアント接続の最大有効期間は 24 時間です。この値は設定できません。クライアント接続が予期せず切断されないように、プールは最大接続時間を 24 時間未満に設定してください。
- クライアント接続アイドルタイムアウト: RDS Proxy は、クライアント接続の最大アイドル時間を適用します。予期せぬ接続の切断を避けるため、RDS Proxy のクライアント接続アイドルタイムアウト設定よりも低い値のアイドル接続タイムアウトをプールに設定します。

アプリケーション側の接続プールに設定されるクライアント接続の最大数は、RDS Proxy の max_connections 設定に制限される必要はありません。

クライアント接続プールにより、クライアント接続の時間が長くなります。接続にピニングが発生した場合、クライアント接続をプールすると多重化の効率が低下する可能性があります。ピニングされているものの、アプリケーション側の接続プールでアイドル状態のクライアント接続は、引き続きデータベース接続を保持し、そのデータベース接続が他のクライアント接続で再利用されるのを防ぎます。プロキシログを確認して、接続にピニングが発生していないかチェックしてください。

> **Note**
>
> RDS Proxy は、データベース接続が使用されなくなった 24 時間後にその接続を閉じます。プロキシは、アイドル状態の最大接続数の設定値に関係なく、このアクションを実行します。

RDS Proxy の固定の回避

データベースリクエストが以前のリクエストの状態情報に依存しない場合、多重化の効率が高まります。その場合、RDS Proxy は、各トランザクションの終了時に接続を再利用できます。このような状態情報の例には、SET ステートメントや SELECT ステートメントで変更できるほとんどの可変や設定パラメータが含まれます。クライアント接続の SQL トランザクションでは、デフォルトで、基となるデータベース接続を多重化できます。

プロキシへの接続は、固定と呼ばれる状態に入る場合があります。接続が固定されると、以降の各トランザクションは、セッションが終了するまで、同じ基になるデータベース接続を使用します。

また、他のクライアント接続は、セッションが終了するまでそのデータベース接続を再利用できません。クライアント接続がドロップされると、セッションは終了します。

RDS Proxy は、他のセッションに不適切なセッション状態の変化を検出すると、クライアント接続を特定の DB 接続に自動的に固定します。固定により、接続の再利用の有効性が低下します。すべての接続やほぼすべての接続で固定が発生する場合は、固定が発生する状態を減らすようにアプリケーションコードやワークロードを変更します。

例えば、アプリケーションによってセッションの変数や設定パラメータが変更されたとします。この場合、後続のステートメントは変更後の変数やパラメータが有効かどうかによって変わります。したがって、RDS Proxy はセッションの可変や構成設定の変更リクエストを処理する場合、そのセッションを DB 接続に固定します。これにより、セッション状態は、同じセッション内の後続のすべてのトランザクションで有効になります。

一部のデータベースエンジンでは、設定可能なすべてのパラメータにこのルールが適用されるわけではありません。RDS Proxy は、特定のステートメントと変数を追跡します。したがって、これらの変更時に RDS Proxy はセッションを固定しません。この場合、RDS Proxy は、これらの設定の値が同じである他のセッションでのみ、接続を再利用します。RDS Proxy がデータベースエンジンで追跡する内容の詳細については、以下を参照してください。

- RDS Proxy が RDS for SQL Server データベースに対して追跡する内容
- RDS Proxy が RDS for MariaDB および RDS for MySQL データベースに対して追跡する内容

RDS Proxy が RDS for SQL Server データベースに対して追跡する内容

RDS Proxy が追跡する SQL Server のステートメントは以下のとおりです。

- `USE`
- `SET ANSI_NULLS`
- `SET ANSI_PADDING`
- `SET ANSI_WARNINGS`
- `SET ARITHABORT`
- `SET CONCAT_NULL_YIELDS_NULL`
- `SET CURSOR_CLOSE_ON_COMMIT`
- `SET DATEFIRST`
- `SET DATEFORMAT`

- SET LANGUAGE
- SET LOCK_TIMEOUT
- SET NUMERIC_ROUNDABORT
- SET QUOTED_IDENTIFIER
- SET TEXTSIZE
- SET TRANSACTION ISOLATION LEVEL

RDS Proxy が RDS for MariaDB および RDS for MySQL データベースに対して追跡する内容

RDS Proxy が追跡する MariaDB および MySQL のステートメントは以下のとおりです。

- DROP DATABASE
- DROP SCHEMA
- 使用

RDS Proxy が追跡する MySQL および MariaDB の変数は以下のとおりです。

- AUTOCOMMIT
- AUTO_INCREMENT_INCREMENT
- CHARACTER SET (or CHAR SET)
- CHARACTER_SET_CLIENT
- CHARACTER_SET_DATABASE
- CHARACTER_SET_FILESYSTEM
- CHARACTER_SET_CONNECTION
- CHARACTER_SET_RESULTS
- CHARACTER_SET_SERVER
- COLLATION_CONNECTION
- COLLATION_DATABASE
- COLLATION_SERVER
- INTERACTIVE_TIMEOUT
- NAMES

- NET_WRITE_TIMEOUT
- QUERY_CACHE_TYPE
- SESSION_TRACK_SCHEMA
- SQL_MODE
- TIME_ZONE
- TRANSACTION_ISOLATION (or TX_ISOLATION)
- TRANSACTION_READ_ONLY (or TX_READ_ONLY)
- WAIT_TIMEOUT

> **Note**
>
> RDS Proxy は、セッションスコープで設定すると、TRANSACTION_ISOLATION 変数と TRANSACTION_READ_ONLY 変数の変更を追跡します。ただし、次のトランザクションスコープで設定すると、RDS Proxy は接続を固定します。この動作は、SET ステートメントと SET TRANSACTION ステートメントのどちらを使用してこれらの値を設定する場合でも適用されます。

固定を最小化する

RDS Proxy のパフォーマンスチューニングでは、固定を最小化してトランザクションレベルの接続の再利用 (多重化) を最大化します。

以下の方法で、固定を最小化できます。

- 固定の原因となる可能性のある不要なデータベースリクエストを避けます。
- すべての接続間で可変と構成設定を一貫して設定します。これにより、これらの特定の設定を持つ接続が後続のセッションで再利用される可能性が高くなります。

  ただし、PostgreSQL では、可変の設定によりセッションの固定が発生します。
- MySQL エンジンファミリデータベースの場合、セッション固定フィルターをプロキシに適用します。特定の種類のオペレーションがアプリケーションの正常な動作に影響しないことがわかっている場合、これらのオペレーションを除外してセッションの固定を起こさないように指定できます。
- Amazon CloudWatch メトリクス DatabaseConnectionsCurrentlySessionPinned をモニタリングして、固定が発生する頻度を確認します。このメトリクスおよび他の CloudWatch メトリ

クスの詳細については、「Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。

- SET ステートメントを使用して各クライアント接続を同等に初期化する場合、トランザクションレベルの多重化を維持したまま、この初期化を実行できます。この場合、初期セッション状態を設定するステートメントを、プロキシが使用する初期化クエリに移動します。このプロパティは、セミコロンで区切られた 1 つ以上の SQL ステートメントを含む文字列です。

  例えば、特定の設定パラメータを設定する初期化クエリをプロキシに定義できます。これにより、RDS Proxy でプロキシの新しい接続を設定するたびに、これらの設定が適用されます。トランザクションレベルの多重化を妨げないように、アプリケーションコードから対応する SET ステートメントを削除できます。

  プロキシでの固定の発生回数のメトリクスについては、「Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。

すべてのエンジンファミリーで固定が発生する条件

以下の場合は、多重化が予期しない動作をもたらす可能性があるため、プロキシはセッションを現在の接続に固定します。

- ステートメントのテキストサイズが 16 KB を超える場合、プロキシはセッションを固定します。

RDS for Microsoft SQL で固定が発生する条件

RDS for SQL サーバーでは、次の操作でも固定が発生します。

- 複数のアクティブな結果セット (MARS) の使用。MARS の詳細については、SQL Server のドキュメントを参照してください。
- 分散トランザクションコーディネーター (DTC) 通信の使用。
- 一時テーブル、トランザクション、カーソル、準備済みステートメントの作成。
- 次の SET ステートメントを使用してください。
  - `SET ANSI_DEFAULTS`
  - `SET ANSI_NULL_DFLT`
  - `SET ARITHIGNORE`
  - `SET DEADLOCK_PRIORITY`
  - `SET FIPS_FLAGGER`

- SET FMTONLY
- SET FORCEPLAN
- SET IDENTITY_INSERT
- SET NOCOUNT
- SET NOEXEC
- SET OFFSETS
- SET PARSEONLY
- SET QUERY_GOVERNOR_COST_LIMIT
- SET REMOTE_PROC_TRANSACTIONS
- SET ROWCOUNT
- SET SHOWPLAN_ALL、SHOWPLAN_TEXT、および SHOWPLAN_XML
- SET STATISTICS
- SET XACT_ABORT

RDS for MariaDB と RDS for MySQL で固定が発生する条件

MariaDB と MySQL の場合、次の操作に伴ってピニングも発生します。

- 明示的なテーブルロックステートメントである LOCK TABLE、LOCK TABLES、または FLUSH TABLES WITH READ LOCK が原因でプロキシによるセッションの固定が発生します。
- GET_LOCK を使用して名前付きロックを作成するプロキシはセッションを固定します。
- ユーザー変数またはシステム変数を設定すると (一部例外あり)、セッションはプロキシに固定されます。これにより接続の再利用が大幅に制限される場合は、固定されないように SET オペレーションを設定できます。これを行うには、セッション固定フィルターのプロパティを調整します。詳細については、「Amazon RDS のプロキシの作成」および「RDS Proxy の変更」を参照してください。
- 一時テーブルを作成した場合、プロキシはセッションを固定します。これにより、トランザクションの境界に関係なく、一時テーブルの内容がセッション全体で保持されます。
- 関数 ROW_COUNT および FOUND_ROWS を呼び出すと、固定が発生する場合があります。
- プリペアドステートメントの場合、プロキシはセッションを固定します。このルールは、プリペアドステートメントで SQL テキストを使用するか、バイナリプロトコルを使用するかに関係なく、適用されます。
- SET LOCAL を使用する場合、RDS Proxy は接続を固定しません。

- ストアドプロシージャやストアド関数を呼び出しても、固定は発生しません。RDS Proxy は、このような呼び出しに伴うセッション状態の変更を検出しません。トランザクション間でのセッション状態の維持を活用している場合は、アプリケーションがストアド ルーチン内でセッション状態を変更しないようにする必要があります。例えば、RDS プロキシは現在、すべてのトランザクションにわたって永続化される一時テーブルを作成するストアドプロシージャと互換性がありません。
- MySQL (構文 /*! ... */) または MariaDB (構文 /*M! ... */) の実行可能なコメントを含むクエリでは、ピン留めが発生します。RDS Proxy は、これらのコメントに埋め込まれた SQL を解析してセッション状態の変化を追跡することはできません。

アプリケーションの動作に関する専門知識がある場合は、特定のアプリケーションステートメントについて固定動作をスキップできます。これを行うには、プロキシの作成時に [セッションの固定フィルタ] オプションを選択します。現在、セッションの可変や構成設定の定義により発生するセッションの固定を回避できます。

RDS for PostgreSQL で固定が発生する条件

PostgreSQL の場合、次の操作に伴って固定も発生します。

- SET コマンドの使用
- PREPARE コマンド、DISCARD コマンド、DEALLOCATE コマンド、または EXECUTE コマンドを使用したプリペアドステートメントの管理
- 一時シーケンス、テーブル、またはビューの作成
- カーソルの宣言
- セッション状態の破棄
- 通知チャネルでのリッスン
- auto_explain などのライブラリモジュールのロード
- nextval や setval などの関数を使用したシーケンスの操作
- pg_advisory_lock や pg_try_advisory_lock などの関数を使用したロックの操作

> **Note**
>
> RDS プロキシは、トランザクションレベルのアドバイザリロック (特に pg_advisory_xact_lock、pg_advisory_xact_lock_shared、pg_try_advisory_xact_l をピン留めしません。

- パラメータの設定、またはパラメータのデフォルトへのリセット。具体的には、SET コマンドと set_config コマンドを使用して、セッション変数にデフォルト値を割り当てます。
- ストアドプロシージャやストアド関数を呼び出しても、固定は発生しません。RDS Proxy は、このような呼び出しに伴うセッション状態の変更を検出しません。トランザクション間でのセッション状態の維持を活用している場合は、アプリケーションがストアド ルーチン内でセッション状態を変更しないようにする必要があります。例えば、RDS プロキシは現在、すべてのトランザクションにわたって永続化される一時テーブルを作成するストアドプロシージャと互換性がありません。
- セッション状態の破棄。使用している接続プールライブラリで DISCARD ALL クエリをリセットクエリとして設定した場合、RDS Proxy は解放時にクライアント接続を固定します。これに伴って、DISCARD ALL コマンドがセッション状態の管理を妨げる場合があり、プロキシの多重化効率が低下し、予期しない結果につながる可能性があります。

RDS Proxy の削除

不要になったプロキシは削除できます。または、プロキシに関連付けられている DB インスタンスやクラスターがサービスから外された場合に、プロキシを削除できます。

AWS マネジメントコンソール

プロキシを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. リストから削除するプロキシを選択します。
4. [Delete Proxy (プロキシの削除)] を選択します。

AWS CLI

DB プロキシを削除するには、AWS CLI コマンド delete-db-proxy を使用します。該当する関連付けを削除するには、deregister-db-proxy-targets コマンドも使用します。

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
```

```
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

RDS API

DB プロキシを削除するには、Amazon RDS API 関数 DeleteDBProxy を呼び出します。プロキシからターゲットの登録を解除するには、DeregisterDBProxyTargets を呼び出します。

Amazon RDS Proxy エンドポイントの操作

RDS Proxy エンドポイントは、データベース接続を管理するための柔軟で効率的な方法を提供し、スケーラビリティ、可用性、セキュリティを向上させます。プロキシエンドポイントを使用すると、次のことが可能になります。

- モニタリングとトラブルシューティングの簡素化 – 複数のエンドポイントを使用して、さまざまなアプリケーションからの接続を個別に追跡し、管理します。

トピック

- プロキシエンドポイントの概要
- プロキシエンドポイントの制限
- マルチ AZ DB クラスターのプロキシエンドポイント
- VPC 間の RDS データベースへのアクセス
- プロキシエンドポイントの作成
- プロキシエンドポイントの表示
- プロキシエンドポイントの変更
- プロキシエンドポイントの削除

プロキシエンドポイントの概要

RDS Proxy エンドポイントを使用する際は、 RDS インスタンスエンドポイントと同じ種類の手順に従います。RDS エンドポイントに詳しくない場合は、「MySQL データベースエンジンを実行している DB インスタンスへの接続」および「PostgreSQL データベースエンジンを実行している DB インスタンスへの接続」で詳細を確認してください。

プロキシエンドポイントを作成する際には、プロキシの仮想プライベートクラウド (VPC) とは異なる VPC にエンドポイントを関連付けることができます。これにより、組織内の別のアプリケーションで使用される VPC など、別の VPC からプロキシに接続できます。

プロキシエンドポイントに関連付けられた制限の詳細については、「プロキシエンドポイントの制限」を参照してください。

RDS Proxy ログでは、各エントリの前に、関連付けられたプロキシエンドポイントの名前が付けられます。これは、ユーザー定義のエンドポイントに指定した名前、またはプロキシのデフォルトの読み取り/書き込みエンドポイントを表す特別な名前 `default` にすることができます。

各プロキシのエンドポイントには、独自の CloudWatch メトリクスのセットがあります。すべてのプロキシエンドポイント、特定のエンドポイント、あるいはプロキシのすべての読み取り/書き込みエンドポイントまたは読み取り専用エンドポイントのメトリクスをモニタリングします。詳細については、「Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。

プロキシエンドポイントは、関連付けられたプロキシと同じ認証メカニズムを使用します。RDS Proxy は、関連付けられたプロキシのプロパティと整合させて、ユーザー定義のエンドポイントのアクセス許可と認可を自動的に設定します。

プロキシエンドポイントの制限

RDS Proxy エンドポイントには以下の制限があります。

- RDS プロキシのデフォルトエンドポイントは変更できません。
- プロキシのユーザー定義エンドポイントの最大数は 20 です。したがって、プロキシには最大 21 個のエンドポイント (デフォルトのエンドポイントとユーザーが作成する 20) を持つことができます。
- 追加のエンドポイントをプロキシに関連付けると、RDS Proxy は、クラスター内のどの DB インスタンスをエンドポイントごとに使用するかを自動的に決定します。
- IPv6 またはデュアルスタックのエンドポイントネットワークタイプの場合、VPC とサブネットは選択したネットワークタイプをサポートするように設定する必要があります。

プロキシを作成すると、RDS はアプリケーションとデータベース間の安全な通信のために VPC エンドポイントを自動的に作成します。VPC エンドポイントが表示され、Amazon VPC コンソールからアクセスできます。

新しいプロキシエンドポイントを追加すると、AWS PrivateLink インターフェイスエンドポイントがプロビジョニングされます。プロキシに 1 つ以上のエンドポイントを追加すると、追加料金が発生します。詳細については、「RDS の料金」を参照してください。

マルチ AZ DB クラスターのプロキシエンドポイント

デフォルトでは、RDS Proxy を マルチ AZ DB クラスターで使用するときに接続するエンドポイントには読み取り/書き込み機能があります。そのため、このエンドポイントではすべてのリクエストをクラスターのライターインスタンスに送信します。これらの接続はすべて、ライターインスタンスの `max_connections` 値にカウントされます。プロキシが マルチ AZ DB クラスターに関連付けられている場合は、そのプロキシ用に追加の読み取り/書き込みエンドポイントまたは読み取り専用エンドポイントを作成できます。

プロキシで読み取り専用エンドポイントを使用すると、読み取り専用クエリを実行できます。これは、マルチ AZ DB クラスターにリーダーエンドポイントを使用するのと同じ方法です。そうすることで、1 つ以上のリーダー DB インスタンスを持つ マルチ AZ DB クラスターの読み取りスケーラビリティを活用するのに役立ちます。読み取り専用エンドポイントを使用し、必要に応じて マルチ AZ DB クラスターにリーダー DB インスタンスを追加することで、より多くの同時クエリを実行し、より多くの同時接続を確立できます。このリーダーエンドポイントは、クエリを多用するアプリケーションの読み取りスケーラビリティを向上させるのに役立ちます。また、リーダーエンドポイントは、クラスター内のリーダー DB インスタンスが使用できなくなった場合に接続の可用性を向上させるのに役立ちます。

マルチ AZ DB クラスターのリーダーエンドポイント

RDS Proxy では、リーダーエンドポイントを作成して使用できます。ただし、これらのエンドポイントは、マルチ AZ DBクラスターに関連付けられたプロキシに対してのみ機能します。RDS CLI または API を使用する場合、値が `TargetRole` で、`READ_ONLY` の属性が表示される場合があります。プロキシのターゲットを RDS DB インスタンスから マルチ AZ DB クラスターに変更することで、これらのプロキシを利用できます。

RDS Proxy を マルチ AZ DB クラスターで使用する場合は、リーダーエンドポイントと呼ばれる読み取り専用エンドポイントを作成して接続できます。

リーダーエンドポイントがアプリケーションの可用性をどのように支援するか

場合によっては、クラスター内のリーダーインスタンスが使用できなくなることがあります。それが発生した場合、DB プロキシのリーダーエンドポイントを使用する接続は、マルチ AZ DB クラスターエンドポイントを使用する接続よりも迅速に回復できます。RDS Proxy は、クラスター内の利

用可能なリーダーインスタンスのみに接続をルーティングします。インスタンスが使用できなくなると、DNS キャッシュによる遅延はありません。

接続が多重化されている場合、RDS Proxy はアプリケーションを中断せずに、後続のクエリを別のリーダーインスタンスに転送します。リーダーインスタンスが使用できない状態になると、そのインスタンスエンドポイントへのすべてのクライアント接続は閉じられます。

接続が固定されている場合、接続の次のクエリはエラーを返します。ただし、アプリケーションはすぐに同じプロキシエンドポイントに再接続できます。RDS Proxy は、`available` 状態の別のリーダー DB インスタンスに接続をルーティングします。手動で再接続する場合、RDS Proxy は古いリーダーインスタンスと新しいリーダーインスタンス間のレプリケーションラグをチェックしません。

マルチ AZ DB クラスターに使用可能なリーダーインスタンスがない場合は、RDS Proxy はリーダーエンドポイントへの接続を試みます。接続借用タイムアウト期間内にリーダーインスタンスが使用可能いならなかった場合、接続試行は失敗します。リーダーインスタンスが利用可能になると、接続試行は成功します。

リーダーエンドポイントがクエリスケーラビリティにどのように役立つか

プロキシのリーダーエンドポイントは、次の方法でマルチ AZ DB クラスタークエリのスケーラビリティを支援します。

- 実用的な場合は、RDS Proxy は特定のリーダーエンドポイント接続を使用するすべてのクエリの問題に同じリーダー DB インスタンスを使用します。これにより、同じテーブルに対する一連の関連クエリが、特定の DB インスタンスでキャッシング、プランの最適化などを活用できます。
- リーダー DB インスタンスが使用できなくなった場合、アプリケーションへの影響は、セッションが多重化されているか固定されているかによって異なります。セッションが多重化されている場合、RDS Proxy は、後続のクエリを、ユーザー側でアクションを実行せずに別のリーダー DB インスタンスにルーティングします。セッションが固定されている場合、アプリケーションはエラーを受け取り、再接続する必要があります。リーダーエンドポイントにすぐに再接続でき、RDS Proxy は、接続を利用可能なリーダー DB インスタンスにルーティングします。プロキシセッションの多重化および固定の詳細については、「RDS Proxy の概念」を参照してください。

VPC 間の RDS データベースへのアクセス

デフォルトでは、RDS テクノロジースタックのコンポーネントはすべて同じ Amazon VPC にあります。例えば、Amazon EC2 インスタンスで実行されているアプリケーションが Amazon RDS DB イ

ンスタンスに接続するとします。この場合、アプリケーションサーバーとデータベースは両方とも同じ VPC 内に存在する必要があります。

RDS Proxy により、EC2 インスタンスなど、別の VPC のリソースから 1 つの VPC の Amazon RDS DB インスタンスへのアクセスを設定できます。例えば、組織に、同じデータベースリソースにアクセスする複数のアプリケーションがあるとします。各アプリケーションは独自の VPC 内にある場合があります。

クロス VPC アクセスを有効にするには、プロキシの新しいエンドポイントを作成します。プロキシ自体は、 Amazon RDS DB インスタンスと同じ VPC に存在します。ただし、クロス VPC エンドポイントは、EC2 インスタンスなどの他のリソースとともに、他の VPC に存在します。クロス VPC エンドポイントは、EC2 および他のリソースと同じ VPC のサブネットおよびセキュリティグループに関連付けられます。このような関連付けにより、VPC の制限によりデータベースにアクセスできないアプリケーションからエンドポイントに接続できます。

次のステップでは、RDS Proxy を使用して VPC 間エンドポイントを作成してアクセスする方法について説明します。

1. 2 つの VPC を作成するか、 RDS の作業に既に使用している 2 つの VPC を選択します。各 VPC には、インターネットゲートウェイ、ルートテーブル、サブネット、セキュリティグループなど、独自のネットワークリソースが関連付けられている必要があります。VPC が 1 つしかない場合は、Amazon RDS のスタート方法 で別の VPC をセットアップして正常に RDS を使用できるようにするステップをご覧ください。Amazon EC2 コンソールで既存の VPC を調べて、相互に接続するリソースの種類を確認することもできます。
2. 接続する Amazon RDS DB インスタンスに関連付けられた DB プロキシを作成します。「Amazon RDS のプロキシの作成」 の手順に従います。
3. RDS コンソールのプロキシの [詳細] ページの [プロキシエンドポイント] セクションで、[エンドポイントの作成] を選択します。「プロキシエンドポイントの作成」 の手順に従います。
4. クロス VPC エンドポイントを読み取り/書き込み可能にするか、読み取り専用にするかを選択します。
5. Amazon RDS DB インスタンスと同じ VPC のデフォルトを受け入れる代わりに、別の VPC を選択します。この VPC は、プロキシが存在する VPC と同じ AWS リージョンに存在する必要があります。
6. これで、Amazon RDS DB インスタンスと同じ VPC からサブネットとセキュリティグループのデフォルトを受け入れるのではなく、新しい選択を行います。選択した VPC のサブネットとセキュリティグループに基づいて、これらを作成します。

7. Secrets Manager シークレットの設定を変更する必要はありません。各エンドポイントがどの VPC にあるかに関係なく、プロキシのすべてのエンドポイントで同じ認証情報が機能します。同様に、IAM 認証を使用する場合、エンドポイントが異なる VPC にある場合でも、IAM 設定とアクセス許可はすべてのプロキシエンドポイント間で一貫して機能します。エンドポイントごとに追加の IAM 設定は必要ありません。
8. 新しいエンドポイントが Available (利用可能) 状態に達するのを待ちます。
9. 完全なエンドポイント名を書き留めます。これは、データベースアプリケーションの接続文字列の一部として指定する、*Region_name*`.rds.amazonaws.com` で終わる値です。
10. エンドポイントと同じ VPC 内のリソースから新しいエンドポイントにアクセスします。このプロセスをテストする簡単な方法は、この VPC 内に新しい EC2 インスタンスを作成することです。次に、EC2 インスタンスにログインし、`mysql` または `psql` コマンドを実行して、接続文字列のエンドポイント値を使用して接続します。

プロキシエンドポイントの作成

プロキシエンドポイントを作成するには、次の手順に従います。

コンソール

プロキシエンドポイントを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. 新しいエンドポイントを作成するプロキシの名前をクリックします。

   そのプロキシの詳細ページが表示されます。
4. [プロキシエンドポイント] セクションで、[プロキシエンドポイントの作成] を選択します。

   [プロキシエンドポイントの作成] ウィンドウが表示されます。
5. [プロキシエンドポイント名] に、選択したわかりやすい名前を入力します。
6. [ターゲットロール] で、エンドポイントを読み取り/書き込みにするか、読み取り専用にするかを選択します。

   読み取り/書き込みエンドポイントを使用する接続では、データ定義言語 (DDL) ステートメント、データ操作言語 (DML) ステートメント、クエリなど、あらゆる種類の操作を実行できます。これらのエンドポイントは、常に RDS DB クラスターのプライマリインスタンスに接続

します。アプリケーションでエンドポイントを 1 つだけ使用する場合は、読み取り/書き込みエンドポイントを一般的なデータベース操作に使用できます。読み取り/書き込みエンドポイントは、管理操作、オンライントランザクション処理 (OLTP) アプリケーション、および抽出変換ロード (ETL) ジョブにも使用できます。

読み取り専用エンドポイントを使用する接続では、クエリのみを実行できます。RDS Proxy はエンドポイントへの接続ごとにリーダーインスタンスの 1 つを使用できます。そうすれば、クエリを多用するアプリケーションは、マルチ AZ DB クラスターのクラスタリング機能を利用できます。読み取り専用接続では、クラスターのプライマリインスタンスでオーバーヘッドが発生することはありません。このようにすると、レポートおよび分析クエリによって、OLTP アプリケーションの書き込みオペレーションの速度が低下することはありません。

7. 仮想プライベートクラウド (VPC) では、デフォルトで、プロキシまたはそれに関連付けられたデータベースへの通常のアクセスに使用するのと同じ EC2 インスタンスまたは他のリソースからエンドポイントにアクセスするように選択します。このプロキシに対してクロス VPC アクセスを設定するには、デフォルト以外の VPC を選択します。クロス VPC アクセスの詳細については、「VPC 間の RDS データベースへのアクセス」を参照してください。
8. [エンドポイントネットワークタイプ] で、プロキシエンドポイントの IP バージョンを選択します。次のオプションを使用できます。

   - IPv4 – プロキシエンドポイントは IPv4 アドレスのみを使用します (デフォルト)。
   - IPv6 – プロキシエンドポイントは IPv6 アドレスのみを使用します。
   - デュアルスタック – プロキシエンドポイントは IPv4 アドレスと IPv6 アドレスの両方をサポートします。

   IPv6 またはデュアルスタックを使用するには、選択したネットワークタイプをサポートするように VPC とサブネットを設定する必要があります。
9. Subnets では、RDS Proxy がデフォルトで関連するプロキシと同じサブネットを入力します。VPC のアドレス範囲の一部のみがエンドポイントに接続できるように、エンドポイントへのアクセスを制限するには、1 つ以上のサブネットを削除します。
10. [VPC セキュリティグループ] で、既存のセキュリティグループを選択することも、新しいセキュリティグループを作成することもできます。RDS Proxy は、デフォルトで、関連付けられたプロキシと同じセキュリティグループを入力します。プロキシのインバウンドルールとアウトバウンドルールがこのエンドポイントに適切な場合は、デフォルトの選択肢のままにしておきます。

新しいセキュリティグループを作成することにした場合は、このページでセキュリティグループの名前を指定します。その後、EC2 コンソールからセキュリティグループ設定を編集します。

11. [プロキシエンドポイントの作成] を選択します。

AWS CLI

プロキシエンドポイントを作成するには、AWS CLI create-db-proxy-endpoint コマンドを使用します。

以下の必須パラメータを含めます。

- `--db-proxy-name` *value*
- `--db-proxy-endpoint-name` *value*
- `--vpc-subnet-ids` *list_of_ids*. サブネット ID はスペースで区切ります。VPC 自体の ID は指定しません。

また、次のオプションパラメータを含めることができます。

- `--target-role { READ_WRITE | READ_ONLY }`このパラメータのデフォルトは です。READ_WRITEプロキシがライター DB インスタンスのみを含む マルチ AZ DB クラスターに関連付けられている場合、READ_ONLY を指定することはできません。マルチ AZ DB クラスターでの読み取り専用エンドポイントの使用目的の詳細については、「マルチ AZ DB クラスターのリーダーエンドポイント」を参照してください。
- `--vpc-security-group-ids` *value*。セキュリティグループ ID はスペースで区切ります。このパラメータを省略すると、RDS Proxy は VPC にデフォルトのセキュリティグループを使用します。RDS Proxy は、`--vpc-subnet-ids` パラメータとして指定したサブネット ID に基づいて VPC を決定します。
- `--endpoint-network-type { IPV4 | IPV6 | DUAL }`。このパラメータは、プロキシエンドポイントの IP バージョンを指定します。デフォルトは IPV4 です。IPV6 または DUAL を使用するには、選択したネットワークタイプをサポートするように VPC とサブネットを設定する必要があります。

Example

次の例では、my-endpoint という名前のプロキシエンドポイントを作成します。

Linux、macOS、Unix の場合:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id \
  --endpoint-network-type DUAL
```

Windows の場合:

```
aws rds create-db-proxy-endpoint ^
  --db-proxy-name my-proxy ^
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id ^
  --endpoint-network-type DUAL
```

RDS API

プロキシエンドポイントを作成するには、RDS API CreateDBProxyEndpoint アクションを使用します。

プロキシエンドポイントの表示

既存のプロキシエンドポイントを表示するには、次の手順に従います。

コンソール

プロキシエンドポイントの詳細を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. リストから、エンドポイントを表示するプロキシを選択します。プロキシ名をクリックして、詳細ページを表示します。
4. で、プロキシエンドポイントセクションで、表示するエンドポイントを選択します。名前をクリックすると、詳細ページが表示されます。

5. 関心のある値を持つパラメータを調べます。次のようなプロパティを確認できます。

   - エンドポイントが読み取り/書き込み可能か読み取り専用か。
   - データベース接続文字列で使用するエンドポイントアドレス。
   - エンドポイントに関連付けられた VPC、サブネットおよびセキュリティグループ。

AWS CLI

1 つ以上の DB プロキシエンドポイントを表示するには、AWS CLI describe-db-proxy-endpoints コマンドを使用します。

以下のオプションのパラメータを含めることができます。

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

次の例では、`my-endpoint` プロキシエンドポイントについて説明します。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Windows の場合:

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

1 つ以上のプロキシエンドポイントを記述するには、RDS API DescribeDBProxyEndpoints オペレーションを使用します。

プロキシエンドポイントの変更

プロキシエンドポイントを変更するには、次の手順に従います。

コンソール

1 つまたは複数のプロキシエンドポイントを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[プロキシ] を選択します。
3. リストから、エンドポイントを変更するプロキシを選択します。プロキシ名をクリックして、詳細ページを表示します。
4. [プロキシエンドポイント] セクションで、変更するエンドポイントを選択します。リストから選択するか、名前をクリックして詳細ページを表示できます。
5. プロキシの詳細ページの [プロキシエンドポイント] セクションで、[編集] を選択します。または、プロキシエンドポイントの詳細ページの [アクション] で、[編集] を選択します。
6. 変更するパラメータの値を変更します。
7. [Save changes] (変更の保存) をクリックします。

AWS CLI

プロキシエンドポイントを変更するには、次の必須パラメータで AWS CLI modify-db-proxy-endpoint コマンドを使用します。

- `--db-proxy-endpoint-name`

次のパラメータの 1 つまたは複数を使用して、エンドポイントプロパティの変更を指定します。

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`。セキュリティグループ ID はスペースで区切ります。

次の例では、`my-endpoint` プロキシエンドポイントの名前を `new-endpoint-name` に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
```

```
  --new-db-proxy-endpoint-name new-endpoint-name
```

Windows の場合:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

RDS API

プロキシエンドポイントを変更するには、RDS API の ModifyDBProxyEndpoint 操作を使用します。

プロキシエンドポイントの削除

プロキシのエンドポイントを削除するには、次の手順に従います。

> **Note**
>
> RDS Proxy がプロキシごとに自動的に作成するデフォルトのプロキシエンドポイントを削除することはできません。
> プロキシを削除すると、RDS Proxy は、関連するすべてのエンドポイントを自動的に削除します。

コンソール

プロキシエンドポイントを削除するにはAWS マネジメントコンソール

1. ナビゲーションペインで、[プロキシ] を選択します。
2. リストから、エンドポイントを設定するプロキシを選択します。プロキシ名をクリックして、詳細ページを表示します。
3. [プロキシエンドポイント] セクションで、削除するエンドポイントを選択します。リストから 1 つ以上のエンドポイントを選択するか、1 つのエンドポイントの名前をクリックして詳細ページを表示できます。
4. プロキシの詳細ページの [プロキシエンドポイント] セクションで、[削除] を選択します。または、プロキシエンドポイントの詳細ページの [アクション] で、[削除] を選択します。

AWS CLI

プロキシエンドポイントを削除するには、次の必須パラメータを指定して delete-db-proxy-endpoint コマンドを実行します。

- `--db-proxy-endpoint-name`

次のコマンドは、`my-endpoint` という名前のプロキシエンドポイントを削除します。

Linux、macOS、Unix の場合:

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Windows の場合:

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

RDS API でプロキシエンドポイントを削除するには、DeleteDBProxyEndpoint オペレーションを実行します。`DBProxyEndpointName` パラメータのプロキシエンドポイントの名前を指定します。

Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング

Amazon CloudWatch を使用して RDS Proxy のモニタリングができます。CloudWatch は、プロキシから raw データを収集し、リアルタイムに近い読み取り可能なメトリクスに加工します。これらのメトリクスを CloudWatch コンソールで確認するには、[Metrics (メトリクス)]、[RDS]、[Per-Proxy Metrics (プロキシごとのメトリクス)] の順に選択します。詳細については、Amazon CloudWatch ユーザーガイド の「Amazon CloudWatch メトリクスの使用」を参照してください。

RDS Proxy コンソールで利用可能なモニタリングダッシュボードを使用することもできます。ダッシュボードには、ウィジェットに配置された事前設定されたメトリクスのセットが用意されており、一般的なオブザーバビリティタスクとトラブルシューティングタスクの実行に役立つように設計されています。詳細については、「RDS Proxy モニタリングダッシュボード」を参照してください。

> **Note**
>
> RDS は、これらのメトリクスを、プロキシに関連付けられている基になる Amazon EC2 インスタンスごとに発行します。1 つのプロキシは、複数の EC2 インスタンスによって処理される場合があります。CloudWatch の統計情報を使用して、すべての関連付けられたインスタンスにわたってプロキシの値を集計します。
> これらのメトリクスの一部は、プロキシによる初期の接続が成功するまで表示されないことがあります。

RDS Proxy ログでは、各エントリの前に、関連付けられたプロキシエンドポイントの名前が付けられます。この名前は、ユーザー定義のエンドポイントに指定した名前、または読み取り/書き込みリクエストを実行するプロキシのデフォルトエンドポイントの特別な名前 `default` にすることができます。

すべての RDS Proxy メトリクスはグループ `proxy` にあります。

各プロキシエンドポイントには独自の CloudWatch メトリクスがあります。各プロキシエンドポイントの使用状況を個別にモニタリングできます。プロキシエンドポイントの詳細については、「Amazon RDS Proxy エンドポイントの操作」を参照してください。

次のいずれかのディメンションセットを使用して、各メトリクスの値を集計できます。例えば、`ProxyName` ディメンションセットを使用すると、特定のプロキシのすべてのトラフィックを分析できます。他のディメンションセットを使用すると、さまざまな方法でメトリクスを分割できます。メトリクスは、各プロキシの異なるエンドポイントまたはターゲットデータベース、または各データベースへの読み取り/書き込みおよび読み取り専用のトラフィックに基づいて分割できます。

- ディメンションセット 1: `ProxyName`
- ディメンションセット 2:`ProxyName,EndpointName`
- ディメンションセット 3:`ProxyName` 、`TargetGroup`、`Target`
- ディメンションセット 4:`ProxyName` 、`TargetGroup`、`TargetRole`

メトリクス	説明	有効期間	CloudWatch ディメンションセット
`AvailabilityPercentage`	ディメンションで示されたロールでター	1 分	Dimension set 4

メトリクス	説明	有効期間	CloudWatch ディメンションセット
	ゲットグループが利用できた時間の割合。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Average です。		
ClientConnections	セットアップ中の接続を含む、現在のクライアント接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 2
ClientConnectionsClosed	閉じられたクライアント接続の数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 2
ClientConnectionsInSetup	開いているがセットアップが完了していないクライアント接続の現在の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 2

メトリクス	説明	有効期間	CloudWatch ディメンションセット
ClientConnectionsNoTLS	Transport Layer Security (TLS) を使用せずに確立されている現在のクライアント接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 2
ClientConnectionsReceived	受信したクライアント接続リクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 2
ClientConnectionsSetupFailedAuth	認証または TLS の設定ミスのために失敗したクライアント接続の試行回数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 2
ClientConnectionsSetupSucceeded	TLS の有無にかかわらず、認証機構を使用して正常に確立されたクライアント接続の数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 2

メトリクス	説明	有効期間	CloudWatch ディメンションセット
ClientConnectionsTLS	TLS で確立されている現在のクライアント接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 2
DatabaseConnectionRequests	データベース接続の作成リクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionRequestsWithTLS	TLS を使用してデータベース接続を作成するリクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnections	セットアップ中の接続を含む、現在のデータベース接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4

メトリクス	説明	有効期間	CloudWatch ディメンションセット
DatabaseConnectionsBorrowLatency	モニタリングされているプロキシがデータベース接続を取得するのにかかる時間 (マイクロ秒)。このメトリクスの最も有用な統計は Average です。	1 分以上	Dimension set 1, Dimension set 2
DatabaseConnectionsBorrowRequests	プールからデータベース接続を借用するリクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 4
DatabaseConnectionsBorrowRequestsAborted	データベース接続を取得する前にクライアント接続が閉じられたために中止された借用リクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 4
DatabaseConnectionsCurrentlyBorrowed	借用状態のデータベース接続の現在の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4

メトリクス	説明	有効期間	CloudWatch ディメンションセット
DatabaseConnectionsCurrentlyInTransaction	トランザクションでの現在のデータベース接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsCurrentlySessionPinned	セッション状態を変更するクライアントリクエストのオペレーションのために現在固定されているデータベース接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsSetupFailed	失敗したデータベース接続リクエストの数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsSetupSucceeded	TLS の有無にかかわらず、正常に確立されたデータベース接続の数。このメトリクスの最も有用な統計は Sum です。	1 分以上	Dimension set 1, Dimension set 3, Dimension set 4

メトリクス	説明	有効期間	CloudWatch ディメンションセット
DatabaseConnectionsWithTLS	TLS を使用する確立された現在のデータベース接続の数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4
MaxDatabaseConnectionsAllowed	許可されるデータベース接続の最大数。このメトリクスは 1 分ごとにレポートされます。このメトリクスの最も有用な統計は Sum です。	1 分	Dimension set 1, Dimension set 3, Dimension set 4
QueryDatabaseResponseLatency	データベースがクエリに応答するのにかかった時間 (マイクロ秒)。このメトリクスの最も有用な統計は Average です。 注: 拡張プロトコルを使用した PostgreSQL ワークロードのデータは含まれません。	1 分以上	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4

メトリクス	説明	有効期間	CloudWatch ディメンションセット
QueryRequests	受信したクエリの数。複数のステートメントを含むクエリは、1 つのクエリとしてカウントされます。このメトリクスの最も有用な統計は Sum です。 注: 拡張プロトコルを使用した PostgreSQL ワークロードのデータは含まれません。	1 分以上	Dimension set 1, Dimension set 2
QueryRequestsNoTLS	非 TLS 接続から受信したクエリの数。複数のステートメントを含むクエリは、1 つのクエリとしてカウントされます。このメトリクスの最も有用な統計は Sum です。 注: 拡張プロトコルを使用した PostgreSQL ワークロードのデータは含まれません。	1 分以上	Dimension set 1, Dimension set 2

メトリクス	説明	有効期間	CloudWatch ディメンションセット
QueryRequestsTLS	TLS 接続から受信したクエリの数。複数のステートメントを含むクエリは、1 つのクエリとしてカウントされます。このメトリクスの最も有用な統計は Sum です。 注: 拡張プロトコルを使用した PostgreSQL ワークロードのデータは含まれません。	1 分以上	Dimension set 1, Dimension set 2
QueryResponseLatency	クエリリクエストを取得してから、プロキシが応答するまでの時間 (マイクロ秒)。このメトリクスの最も有用な統計は Average です。 注: 拡張プロトコルを使用した PostgreSQL ワークロードのデータは含まれません。	1 分以上	Dimension set 1, Dimension set 2

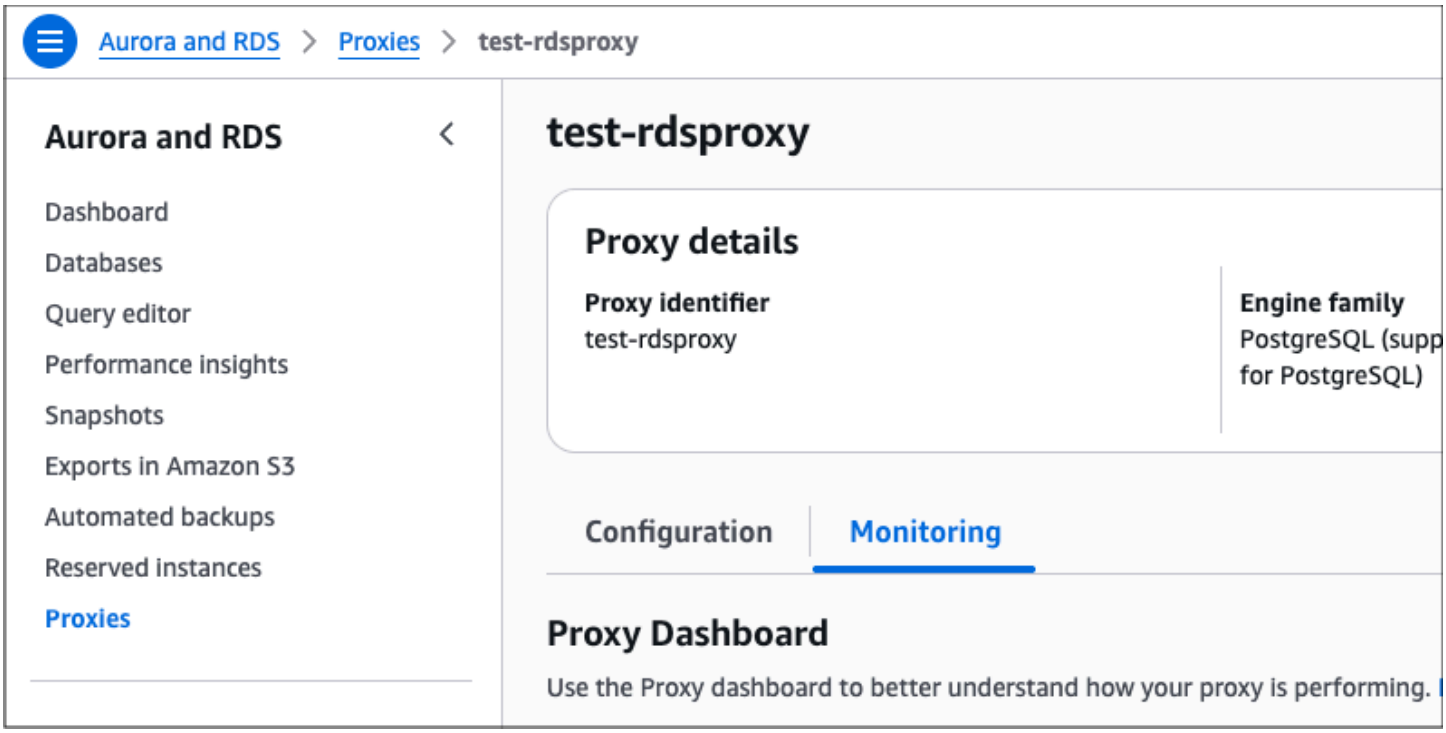
RDS Proxy アクティビティのログは、AWS マネジメントコンソール の CloudWatch にあります。各プロキシには、[ロググループ] ページにエントリがあります。

> **Important**
>
> これらのログは、トラブルシューティングを目的としたもので、プログラムによるアクセス用ではありません。ログの形式と内容は変更される可能性があります。

特に、古いログには、各リクエストのエンドポイントを示すプレフィックスは含まれていません。新しいログでは、各エントリの先頭に、関連付けられたプロキシエンドポイントの名前が付けられます。この名前は、ユーザー定義のエンドポイントに指定した名前、またはプロキシのデフォルトのエンドポイントを使用するリクエストの特別な名前 `default` にすることができます。

RDS Proxy モニタリングダッシュボード

RDS Proxy のモニタリングダッシュボードには、プロキシの [モニタリング] タブからアクセスできます。



ダッシュボードには、ウィジェットに配置された事前設定されたメトリクスのセットが用意されており、一般的なオブザーバビリティタスクとトラブルシューティングタスクの実行に役立つように設計されています。このダッシュボードにメトリクスが表示されない場合は、ダッシュボードのリンクを使用して CloudWatch のすべてのメトリクスを表示できます。

デフォルトのダッシュボードにメトリクスを追加または削除することはできません。プロキシのカスタムモニタリングビューを作成する場合は、CloudWatch で直接作成できます。詳細については、「CloudWatch ダッシュボードの使用」を参照してください。

モニタリングダッシュボードは複数のセクションに分かれており、メトリクスは事前定義された集計を使用してウィジェットごとにグループ化されています。基盤となる RDS Proxy メトリクスの詳細については、「CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。

トピック

- 概要
- レイテンシー
- 接続プール
- データベース接続の使用状況

概要

可用性 % (ロール別)

`AvailabilityPercentage` メトリクスの集計に基づいて、`READWRITE` および `READ_ONLY` ロールでターゲットが使用可能であった時間の割合。

可用性メトリクスは、プロキシによって実行されるデータベースのヘルスチェックの成功率を表すものであり、ターゲットデータベースインスタンスのヘルスステータスを反映します。このウィジェットは各ロールの `Average` 統計を使用し、計算は特定のロールにおけるデータベースインスタンスの数 (`READWRITE` ロールの場合は 1 つ、`READ_ONLY` ロールの場合は複数の場合あり) や、プロキシ内のノードの数 (RDS Proxy インフラストラクチャによって決定) の影響を受けます。

> **Note**
>
> RDS Proxy は、登録されている DB インスタンスの設定に基づいて自動的に容量を調整します。プロキシ内のノード数を直接制御することはできませんが、可用性メトリクスの `sample count` 統計を使用して現在のノード数を確認することは可能です。RDS Proxy が容量を調整する方法の詳細については、「IP アドレス容量の計画」を参照してください。

`READWRITE` ターゲットロールには単一のデータベースインスタンスのみを含めることができ、可用性の割合はすべてのプロキシノードにおけるヘルスチェック成功率の平均として計算されます。例えば、プロキシに 4 つのノードがあり、そのうちの 1 つが特定の期間内に正常なヘルスチェックレスポンスを受信しなかった場合、その期間の読み取り/書き込みの可用性は 75% と報告されます。同様

に、プロキシに 4 つのノードがあり、そのいずれもが 1 分間のうち 30 秒間ヘルスチェックを実行できなかった場合、プロキシはその 1 分間における可用性を 50% と報告します。

READ_ONLY ターゲットロールには、Aurora クラスター内の複数のレプリカなど、複数のデータベースインスタンスを含めることができます。各 RDS Proxy ノードの観点から、ターゲットインスタンスのいずれかがヘルスチェックに合格すると、読み取り専用ロールは使用可能であると見なされます。その後、すべてのプロキシノードにわたって可用性が平均化され、最終的な数がウィジェットに表示されます。次の 4 つのノードを含むプロキシの例を考えてみましょう。

- 各プロキシノードは、複数の読み取り専用ターゲット (データベースインスタンス) に対してヘルスチェックを実行します。一部のチェックは成功し、一部のチェックは失敗します。プロキシノードは、少なくとも 1 つのターゲットインスタンスから正常なヘルスチェックレスポンスを受け取る限り、読み取り専用ロールが使用可能であることを報告します。
- 4 つのプロキシノードは、複数の読み取り専用ターゲットに対してヘルスチェックを実行します。3 つのプロキシノードは少なくとも 1 つのターゲットに接続できますが、1 つのプロキシノードはどのターゲットにも接続できません。読み取り専用ロールの可用性は、プロキシノードから報告されたロールの可用性の `Average` として計算されます。`(100% + 100% + 100% + 0% )/4 = 75%`

> **Note**
>
> RDS Proxy `READ_ONLY` ターゲットロールは、Aurora クラスター、または 2 つの読み取り可能なスタンバイを持つ Amazon RDS マルチ AZ DB クラスターのデプロイでサポートされていますが、リードレプリカを使用する Amazon RDS DB インスタンスのデプロイではサポートされていません。プロキシに読み取り専用ターゲットが含まれていない場合、`READ_ONLY` ロールの可用性は「ゼロ」と表示されることがあります。

ロール可用性ウィジェットは、以下のシナリオで最も役立ちます。

- 接続エラーの診断。プロキシノードまたはターゲットデータベースインスタンスの正常性によって、可用性が影響を受けたかどうかをすばやく評価したい場合。
- データベースで明確な可用性の低下は発生していないものの、プロキシのヘルスチェックメトリクスがデータベースの過負荷という仮説を裏付ける場合の、高トラフィックイベントの診断。RDS Proxy は、モニタリングとヘルスチェックの役割のために予約された複数の永続的な接続を使用します。そのため、データベースメトリクスがダウンタイムを示していない一方で、プロキシのヘル

スチェックが断続的に失敗している場合は、データベースに過度の負荷がかかっていることを示す兆候である可能性があります。

- プロキシがターゲットインスタンスに接続できない原因となる設定の問題を特定します。例えば、プロキシの可用性メトリクスが低下しているにもかかわらず、他のシグナル上ではターゲットインスタンスが正常であるように見える場合は、プロキシのサブネットに関連付けられたネットワーク ACL の問題や、ターゲットデータベースインスタンスにアタッチされたセキュリティグループの設定に問題がある可能性があります。

RDS Proxy の可用性メトリクスに関連する問題を診断する場合は、データベース自体のヘルスと使用率の指標も確認する必要があります。プロキシの可用性の計算は、データベースがヘルスチェックに応答できるかどうかに左右されます。そのため、結論を出す前に両側から情報を収集する必要があります。

データベース接続の使用状況 % (ロール別)

各ロールにおけるプロキシの接続プールの使用率。DatabaseConnections を MaxDatabaseConnectionsAllowed で割って計算されます。

許可されるデータベース接続の最大数は、MaxConnectionsPercent 設定によって制御されるプロキシの最大プールサイズを指します。これは、max_connections パラメータなどのデータベース接続制限によって異なります。この設定の詳細については、「RDS Proxy 接続に関する考慮事項」を参照してください。

> **Note**
>
> RDS Proxy は、MaxConnectionsPercent クォータの一部を内部モニタリング接続用に確保します。予約済み接続の総数は一定ではなく、内部プロキシノードの数とターゲットデータベースインスタンスの数によって異なります。

許可される接続数とピーク負荷時に予想されるデータベース接続の最大数との間に、少なくとも 30% のヘッドルームを持たせることをお勧めします。このヘッドルームにより、予期しないワークロードの急増時にもクライアントエクスペリエンスが向上し、RDS Proxy が内部インフラストラクチャ全体に接続を再分散して、熱管理やその他の目的に役立てることができます。この推奨事項に従うために、接続使用率ウィジェットには、最大数の 70% の位置に固定のガイドラインが表示されています。

例えば、1 つのライターと 2 つのリーダーで構成され、各インスタンスの max_connections 設定が 500 である Aurora PostgreSQL クラスターがあり、MaxConnectionsPercent が 50 に設定されたプロキシを使用しているとします。

1. RDS Proxy は、モニタリング用に多数の接続を確保するため、インスタンスあたりの MaxDatabaseConnectionsAllowed は 250 未満となります。(max_connections=500 * MaxConnectionsPercent=50%) - reserved connections
2. 1 つのライターを含む READWRITE ロールの場合、プロキシ接続プールの最大サイズは 250 をわずかに下回る値 (50% * 500 - reserved_connections) となり、70% のガイドラインは 175 接続未満 (70% * 250 - reserved connections) に相当します。
3. 2 つのリーダーを含む READ_ONLY ロールの場合、プロキシ接続プールの最大サイズは 500 をわずかに下回り (2 * 50% * 500 - reserved_connections)、70% のガイドラインは 350 接続未満 (70% * 500 - reserved connections) に相当します。

このウィジェットは、接続使用率の一般的な指標として機能し、特定のトラブルシューティングシナリオの出発点として役立ちます。例えば、次のようになります。

- 接続使用率が頻繁かつ急激に変動する場合、MaxIdleConnectionsPercent 設定が低すぎる可能性があります。アイドル状態の接続をより長期間維持するようにすれば、接続のチャーンを減らすことができます。
- ワークロードパターンとクライアントアクティビティに基づいて、特定の時点で使用されている接続の数について、大まかな予想を立てられるようになります。ウィジェットは、これらの想定を検証し、クライアントの潜在的な問題を検出するのに役立ちます。MaxIdleConnectionsPercent 設定は非常に低いものの、接続使用率が一貫して非常に高い状況を考えてみましょう。これは、クライアントがアイドル状態になることはほとんどないことを示している可能性がありますが (必ずしも問題というわけではありません)、接続がピン留めされているため、アイドル状態チェックの対象外になっていることを意味している可能性もあります。
- ピン留めまたはプロキシ設定に問題がない場合、データベース接続使用率が高いことは、データベースの max_connections 設定を引き上げる必要があるか、読み取り専用ロールの場合には、クラスターに追加のリードレプリカが必要であることを示している可能性があります。

レイテンシーの内訳

接続の借用とクエリレスポンスのレイテンシー。

ラベル	説明	ソース
DatabaseConnectionsBorrowLatency	プール内の既存の接続か、データベースに対して新たに開く必要がある接続かにかかわらず、プロキシがデータベース接続を取得するのにかかる時間。	RDS Proxy、DatabaseConnectionsBorrowLatency メトリクス
QueryResponseLatency	クエリリクエストを取得してから、プロキシが応答するまでの時間。	RDS Proxy、QueryResponseLatency メトリクス
QueryDatabaseResponseLatency	プロキシとデータベース間のネットワークレイテンシーなど、データベースがクエリに応答するのにかかった時間。	RDS Proxy、QueryDatabaseResponseLatency メトリクス

このウィジェットは、クエリの平均応答時間や、プロキシレイヤーが合計レイテンシーにどのように寄与しているかを理解するのに役立ちます。借用レイテンシーメトリクスは、プールの輻輳を検出するのに特に役立ちます。これは、プールですぐに利用できる接続が十分にないため、クライアントリクエストが待機している状態です。

これらのメトリクスのデータポイントは、値が 0 より大きい場合にのみ発行されます。つまり、クエリトラフィックがない期間には、ウィジェットにギャップが含まれる可能性があります。

クライアント接続

接続状態別に分類された、クライアントとプロキシ間の接続の数。

ラベル	説明	ソース
ClientConnectionsReceived	受信したクライアント接続リクエストの数。	RDS Proxy、ClientConnectionsReceived メトリクス

ラベル	説明	ソース
ClientConnectionsSetupSucceeded	正常に確立されたクライアント接続の数。	RDS Proxy、ClientConnectionsSetupSucceeded メトリクス
ClientConnectionsSetupFailedAuth	認証または TLS の設定ミスのために失敗したクライアント接続の試行回数。	RDS Proxy、ClientConnectionsSetupFailedAuth メトリクス
ClientConnectionsClosed	閉じられたクライアント接続の数。	RDS Proxy、ClientConnectionsClosed メトリクス

このウィジェットは、複数のクライアント接続メトリクスを 1 つのグラフに表示することで、クライアント接続のスループットやステータスを評価するのに役立ちます。これは、クライアント接続の全体的な成功率をモニタリングし、接続の開閉率を観察するのに役立ちます。これは、接続ストームやクライアントアクティビティの経時的な変化を検出するのに役立ちます。

Proxy ワークロード

QueryRequests プロキシメトリクスに基づく、プロキシエンドポイントあたりのクエリリクエストの数。

データ系列のラベルはエンドポイントによって異なります。

- デフォルトの読み取り/書き込みプロキシエンドポイントは、default ラベルを使用します。
- AWS マネジメントコンソールを使用してプロキシを作成するときにオプションの読み取り専用エンドポイントを有効にした場合、このエンドポイントはプロキシ名のプレフィックスが付いた read-only ラベルを使用します。my-proxy という名前のプロキシの場合、ラベルは my-proxy-read-only になります。
- 追加のプロキシエンドポイントは、追加のプレフィックスなしでエンドポイント名をラベルとして使用します。

このウィジェットは、各プロキシエンドポイントで処理されるワークロードのボリュームをモニタリングするのに役立ちます。

> **Important**
>
> 拡張クエリプロトコルを使用する PostgreSQL 接続でのクエリアクティビティについては、現在、`QueryRequests` メトリクスは出力されません。

レイテンシー

データベース応答レイテンシー (ターゲット別)

`QueryDatabaseResponseLatency` プロキシメトリクスに基づくターゲット (データベースインスタンス) ごとのデータベースレスポンスレイテンシー。

このウィジェットは、プロキシ配下の各データベースインスタンスにおけるデータベースレスポンスレイテンシーをモニタリングするのに役立ちます。特に、ワークロードごとに異なるデータベースインスタンスを使用する場合や、インスタンスサイズが均一でない場合に有効です。次の例を考えます。

- データベースクラスターには複数のリードレプリカが含まれており、そのすべてがプロキシを介してトラフィックを受信します。さらに、一部のレプリカはプロキシをバイパスするクライアントから直接トラフィックを受信します。このウィジェットは、追加のトラフィックがプロキシを介して処理されるワークロードのクエリレイテンシーに影響を与えているかどうかを判断するのに役立ちます。
- データベースクラスターにはさまざまなサイズのレプリカが含まれており、ワークロード全体がプロキシを介して処理されます。また、データベースインスタンスはデフォルトの接続設定を使用しており、接続制限はインスタンスサイズに比例します。プロキシは接続制限に従い、小規模なインスタンスには比例してより少ない接続を割り当てますが、それらのインスタンスはキャッシュやバッファも小さいため、レイテンシーが高くなる可能性があります。このウィジェットは、それが当てはまるかどうか、およびクエリレイテンシーの点で、小さなインスタンスがより大きなインスタンスに遅れを取らずに処理できているかどうかを判断するのに役立ちます。

接続プール

ピン留めされたデータベース接続 % (ロール別)

各ロールにピン留めされたデータベース接続の割合。`DatabaseConnectionsCurrentlySessionPinned` を `DatabaseConnections` で割って計算されます。

このウィジェットは、ワークロードにおける接続ピン留めをモニタリングするのに役立ちます。他のウィジェットと同様に、ピン留めされた接続の絶対数よりも適切なヘルスインジケータとして、相対値が使用されています。理想的な値はゼロですが、すべてのワークロードで接続ピン留めを完全に回避できるとは限りません。接続が他のワークロードに識別可能な影響を与えない限り、接続のごく一部でピン留めが発生することを許容できるかもしれません。ただし、ワークロード全体の大部分でピン留めが発生している場合、RDS Proxy による接続の多重化 (トランザクション間での接続の再利用) を妨げるようなクライアントの動作に問題がある可能性があります。

接続の固定の詳細については、「RDS Proxy の固定の回避」を参照してください。

データベース接続メトリクスをセットアップ (ロール別)

プロキシとデータベース間の接続セットアップの数。セットアップ結果とターゲットロール別に分類されます。ウィジェットには、ロールごとに複数のデータ系列が表示されます。

ラベル	説明	ソース
DatabaseConnectionRequests	データベース接続の作成リクエストの数。	RDS Proxy、DatabaseConnectionRequests メトリクス
DatabaseConnectionsSetupSucceeded	正常に確立されたデータベース接続の数。	RDS Proxy、DatabaseConnectionsSetupSucceeded メトリクス
DatabaseConnectionsSetupFailed	失敗したデータベース接続リクエストの数。	RDS Proxy、DatabaseConnectionsSetupFailed メトリクス

これらのメトリクスのデータポイントは、値が 0 より大きい場合にのみ発行されます。つまり、関連するイベントがない期間には、ウィジェットにギャップが含まれる可能性があります。例えば、クライアントが特定の期間中にプロキシを介してクエリを実行し、プロキシが既存のバックエンドデータベース接続を (新しい接続を開かずに) 再利用することでトラフィック全体を処理できる場合、その期間のデータポイントは表示されません。

このウィジェットは、プロキシとデータベース間のアクティビティと接続チャーンを観察するのに役立ちます。[クライアント接続] ウィジェットに表示される情報と組み合わせることで、プロキシ内の接続プーリングや多重化の効率を評価できます。

RDS Proxy は、接続プール全体を最大容量で事前に割り当てるわけではなく、また、一定期間非アクティブなアイドル状態のデータベース接続を切断することに留意してください。負荷がかかっている際に、DatabaseConnectionRequests が 0 を上回り、変動することは正常ですが、注意が必要なのは DatabaseConnectionRequests と ClientConnectionsReceived との比率です。例えば、次のようになります。

- クライアントがプロキシに対して一定数の接続を開き、プロキシが各期間にほぼ同数のデータベースへの接続を開く場合、固定などの理由でプロキシが接続を再利用できていない可能性があります。この状況では、プロキシが接続を再利用できないようにしているクライアントの動作に焦点を当てて、さらに調査が必要になる場合があります。
- クライアントがプロキシへの一定数の接続を開いているにもかかわらず、新しいデータベース接続がその数のごく一部にすぎない場合、プロキシが既存の接続を再利用することでトラフィックの大部分を処理していることを示しています。これは、接続プールの通常の望ましい状態です。

データベース接続メトリクスをセットアップ (ターゲット別)

データベース接続メトリクスをセットアップ (ロール別) に相当するもので、ターゲット (データベースインスタンス)ごとにデータを内訳表示したものです。

データベース接続の使用状況

データベース接続の使用状況 % (ターゲット別)

各ターゲット (データベースインスタンス) におけるプロキシの接続プールの使用率。DatabaseConnections を MaxDatabaseConnectionsAllowed で割って計算されます。

このウィジェットは、データベース接続の使用状況 % (ロール別) と似ていますが、各データベースインスタンスの接続使用率を詳細に表示します。これは、複数のインスタンスを持つデータベースクラスターで、READ_ONLY ロールへの接続が公平に分散されているかを検証するのに役立ちます。

例えば、2 つのリーダーを持ち、リーダーあたりの MaxDatabaseConnectionsAllowed が 500 に設定されている Aurora クラスターがあるとします。プロキシ接続の数が最初のリーダーで 100、2 番目のリーダーで 125 の場合、ウィジェットには最初のリーダーで 20%、2 番目のリーダーで 25% の接続使用率が表示されます。

> **Note**
>
> RDS Proxy は、データベースの接続制限に基づいて重み付けを行い、各レプリカのクライアント接続の数を均等化しようとします。CPU 使用率などのデータベースメトリクスに基づい

てトラフィックを負荷分散することはなく、各クライアントセッションが開始された後、それがどのくらい続くかを把握することもありません。その結果、インスタンス間の接続使用率のわずかな不均衡は、通常の現象です。

データベース接続の使用状況 (READWRITE/READ_ONLY ロール別)

プロキシとデータベースが使用および許可するデータベース接続の数。このタイプのウィジェットは 2 つあります。1 つは `READWRITE` ロール用、もう 1 つは `READ_ONLY` ロール用です。

このウィジェットは、プロキシのデータベースターゲットが `READ_ONLY` ロールをサポートしている場合にのみ表示されます。これには Aurora DB クラスターと Amazon RDS マルチ AZ DB クラスターのデプロイが含まれます。Amazon RDS DB インスタンスのデプロイでは、ウィジェットは表示されません。

各ウィジェットには、複数のデータ系列が含まれています。

ラベル	説明	ソース
`Database Connections`	プロキシまたは他のクライアントによって開かれたかどうかにかかわらず、データベースに存在する接続の数。	データベース、`DatabaseConnections` メトリクス
`Database Connections: proxy`	RDS Proxy がデータベースのモニタリングに使用する内部接続を除く、プロキシとデータベースの間に存在するデータベース接続の数。	RDS Proxy、`DatabaseConnections` メトリクス
`MaxDatabaseConnectionsAllowed: Proxy`	`MaxConnectionsPercent` 設定によって制御される、プロキシによって許可されるデータベース接続の最大数。	RDS Proxy、`MaxDatabaseConnectionsAllowed` メトリクス

表示される数値は、プロキシと基盤となるデータベースクラスターの合計値です。例えば、データベースクラスターにそれぞれ 1,000 の接続を許可する 2 つのインスタンスが含まれてお

り、プロキシの MaxConnectionsPercent が 20 に設定されているとします。このシナリオでは、MaxDatabaseConnectionsAllowed: Proxy は 2 * 1000 * 20% = 400 から、モニタリング目的で RDS Proxy によって内部的に予約された接続の数を減算したものになります。実際の値は 400 未満となり、プロキシ内のノード数によって異なります。

このウィジェットは、以下の場合に最も役立ちます。

- プロキシとデータベースの絶対接続使用率の数値をモニタリングします。
- プロキシ経由でデータベースに到達する接続数と、プロキシをバイパスする接続数を特定します。
- 現在の接続使用率と最大許容使用率との差を計算します。

> **Note**
>
> DatabaseConnections の値は、すべてのクライアント接続がプロキシを介して処理される場合でも、Database Connections: Proxy よりも大きくなります。RDS プラットフォームと RDS Proxy インフラストラクチャによって維持される内部接続は常に多数あります。これらの接続はデータベースによってカウントされますが、プロキシメトリクスには反映されません。

RDS Proxy イベントの使用

イベントとは、AWS 環境やサービスまたは Software as a Service (SaaS) パートナーからのアプリケーションなどの環境での変化を示します。あるいは、お客様独自のカスタムアプリケーションやサービスのいずれかの場合があります。例えば、RDS Proxy を作成または変更すると、Amazon RDS がイベントを生成します。Amazon RDS は、Amazon EventBridge に対してほぼリアルタイムでイベントを配信します。以下に、サブスクライブできる RDS Proxy イベントのリストと、RDS Proxy イベントの例を示します。

イベントの操作に関する詳細は、以下を参照してください。

- を使用してイベントを表示する方法については、を参照してください。AWS マネジメントコンソール、AWS CLI、または RDS API については、Amazon RDS イベントの表示を参照してください。
- 構成方法については、こちらをご覧ください。Amazon RDSEventBridge にイベントを送信するには、「Amazon RDS イベントでトリガーするルールの作成」を参照してください。

RDS Proxy イベント

ソースタイプが RDS Proxy である場合の、イベントのカテゴリとその一覧を次の表に示します。

カテゴリ	RDS イベント ID	メッセージ	メモ
設定変更	RDS-EVENT-0204	RDS が DB プロキシ *name* を変更しました。	なし
設定変更	RDS-EVENT-0207	RDS において、DB プロキシ *name* のエンドポイントが修正されました。	なし
設定変更	RDS-EVENT-0213	RDS が DB インスタンスの追加を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループに自動的に追加しました。	なし
設定変更	RDS-EVENT-0214	RDS が DB インスタンス *name* の削除を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループ *name* から自動的に削除しました。	なし
設定変更	RDS-EVENT-0215	RDS が DB クラスター *name* の削除を検出し、そのインスタンスを DB プロキシ *name* のターゲットグループ *name* から自動的に削除しました。	なし
作成	RDS-EVENT-0203	RDS は DB プロキシ *name* を作成しました。	なし

カテゴリ	RDS イベント ID	メッセージ	メモ
作成	RDS-EVENT-0206	RDS は DB プロキシ *name* のエンドポイント*name*を作成しました。	なし
削除	RDS-EVENT-0205	RDS は DB プロキシ *name* を削除しました。	なし
削除	RDS-EVENT-0208	RDS は DB プロキシ*name*のエンドポイント *name* を削除しました。	なし
失敗	RDS-EVENT-0243	サブネット *name* に十分な IP アドレスがないため、RDS はプロキシの容量をプロビジョニングできませんでした: *name*。この問題を解決するには、RDS プロキシのドキュメントで推奨されているように、サブネットの未使用の IP アドレスが最小限であることを確認してください。	インスタンスクラスの推奨数を決定するには、「IP アドレス容量の計画」を参照してください。
失敗	RDS-EVENT-0275	RDS は DB プロキシ#への一部の接続をスロットリングしました。クライアントからプロキシへの同時接続リクエストの数が制限を超えました。	なし

以下は、JSON 形式の RDS Proxy イベントの例です。このイベントは、`my-rds-proxy` という名前の RDS Proxy の `my-endpoint` という名前のエンドポイントを、RDS が変更したことを示します。イベント ID は RDS-EVENT-0207 です。

```
{
```

```
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```


RDS Proxy のトラブルシューティング

以下に、いくつかの一般的な RDS Proxy 問題のトラブルシューティングのヒントと、RDS Proxy の CloudWatch ログに関する情報を示します。

RDS Proxy ログでは、各エントリの前に、関連付けられたプロキシエンドポイントの名前が付けられます。この名前には、ユーザー定義のエンドポイントに指定した名前を使えます。または、読み取り/書き込みリクエストを実行するプロキシのデフォルトエンドポイントの特別な名前 `default` にすることができます。プロキシエンドポイントの詳細については、「Amazon RDS Proxy エンドポイントの操作」を参照してください。

トピック

- プロキシでの接続の検証
- 一般的な問題と解決策
- RDS for MySQL での RDS Proxy の問題のトラブルシューティング
- RDS for PostgreSQL での RDS Proxy の問題のトラブルシューティング

プロキシでの接続の検証

以下のコマンドを使用して、接続内のプロキシ、データベース、コンピューティングインスタンスなどのすべてのコンポーネントが相互に通信できることを確認できます。

describe-db-proxies コマンドを使用して、プロキシ自体を調べます。また、describe-db-proxy-target-groups コマンドを使用して、関連するターゲットグループを確認します。ターゲットの詳細が、プロキシに関連付ける RDS DB インスタンス と一致していることを確認します。以下のようなコマンドを使用します。

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

プロキシが基になるデータベースに接続できることを確認するには、describe-db-proxy-targets コマンドを使用して、ターゲットグループで指定されたターゲットを調べます。以下のようなコマンドを使用します。

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

describe-db-proxy-targets コマンドの出力には、TargetHealth フィールドが含まれます。State 内のフィールド Reason、Description、および TargetHealth を調べて、プロキシが基になる DB インスタンスと通信できるかどうかを確認できます。

- State の値 AVAILABLE は、プロキシが DB インスタンスに接続できることを示します。
- State の値 UNAVAILABLE は、一時的または永続的な接続の問題を示します。この場合は、Reason および Description フィールドを調べます。例えば、Reason の値が PENDING_PROXY_CAPACITY の場合は、プロキシがスケーリングオペレーションを完了した後で、接続を再試行します。Reason の値が UNREACHABLE、CONNECTION_FAILED、または AUTH_FAILURE の場合は、Description フィールドの説明が問題の診断に役立ちます。
- State フィールドでは、REGISTERING または AVAILABLE に変わるまでの短い間、値が UNAVAILABLE になる場合があります。

次の Netcat コマンド (nc) が成功した場合は、ログインしている EC2 インスタンスや他のシステムからプロキシエンドポイントにアクセスできます。このコマンドは、プロキシおよび関連付けられたデータベースと同じ VPC 内に存在していない場合、失敗を報告します。同じ VPC に存在していなくても、データベースに直接ログインできる場合があります。ただし、同じ VPC 内に存在していない限り、プロキシにはログインできません。

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

次のコマンドを使用して、EC2 インスタンスに必要なプロパティがあることを確認できます。特に、EC2 インスタンスの VPC は、プロキシが接続する先の の VPC と同じである必要があります。

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

プロキシで使用されている Secrets Manager シークレットを確認します。

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

`get-secret-value` によって表示される `SecretString` フィールドが JSON 文字列としてエンコードされ、`username` フィールドと `password` フィールドが含まれていることを確認します。次の例は、`SecretString` フィールドの形式を示しています。

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

IAM 認証の問題をトラブルシューティングする場合は、以下を確認してください。

- データベースで IAM 認証が有効になっています。
- プロキシは正しい認証スキームで設定されています。
- プロキシに提供される IAM ロールの IAM ポリシーは、適切なデータベースとそのユーザー名に必要な `rds-db:connect` アクセス許可を付与します。
- エンドツーエンド IAM 認証の場合、IAM ユーザーまたはロール名に一致するデータベースユーザーが存在します。
- SSL/TLS は接続に対して有効になっています。

一般的な問題と解決策

このセクションでは、RDS Proxy を使用する際の一般的な問題と考えられる解決策について説明します。

`aws rds describe-db-proxy-targets` CLI コマンドの実行後、`TargetHealth` の説明に `Proxy does not have any registered credentials` と記載されている場合は、以下を確認してください。

- ユーザーがプロキシにアクセスするための認証情報が登録されています。
- プロキシが使用する Secrets Manager シークレットにアクセスする IAM ロールが有効であること。

DB プロキシの作成時や接続時に、次の RDS イベントが発生することがあります。

カテゴリ	RDS イベント ID	説明
失敗	RDS-EVENT-0243	サブネットに十分な IP アドレスがないため、RDS はプロキシの容量をプロビジョニングできませんでした。この問題を解決するには、サブネットの未使用の IP アドレスが最小限であることを確認してください。インスタンスクラスの推奨数を決定するには、「IP アドレス容量の計画」を参照してください。
失敗	RDS-EVENT-0275	RDS は DB プロキシ#への一部の接続をスロットリングしました。クライアントからプロキシへの同時接続リクエストの数が制限を超えました。

新しいプロキシの作成時やプロキシへの接続時に、次の問題が発生することがあります。

エラー	原因または回避策
`403: The security token included in the request is invalid`	新しい IAM ロールを作成せずに、既存の IAM ロールを選択します。

RDS for MySQL での RDS Proxy の問題のトラブルシューティング

MySQL プロキシへの接続時に次の問題が発生することがあります。

エラー	原因または回避策
`ERROR 1040 (HY000): Connections rate limit exceeded (`*`limit_value`*` )`	クライアントからプロキシへの接続リクエストのレートが制限を超えました。
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	クライアントからプロキシへの IAM 認証による同時リクエストの数が制限を超えました。
`ERROR 1040 (HY000): Number simultaneous connections exceeded (`*`limit_value`*` )`	クライアントからプロキシへの同時接続リクエストの数が制限を超えました。
`ERROR 1045 (28000): Access`	プロキシで使用される Secrets Manager シークレットが既存のデータベースユーザーのユーザー名およびパスワードと一致しません。Secrets Manager シークレットの認証情報を更新します。または、データベース

エラー	原因または回避策
`denied for user 'DB_USER'@'%' (usi password: YES)`	ユーザーが存在し、そのパスワードがシークレットのものと同じであることを確認します。
`ERROR 1105 (HY000): Unknown error`	不明なエラーが発生しました。
`ERROR 1231 (42000): Variable ''charact er_set_cl ient'' can't be set to the value of value`	`character_set_client` パラメータに設定した値が無効です。例えば、値 `ucs2` は、MySQL サーバーをクラッシュさせる可能性があるため、有効ではありません。
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	プロキシで [Transport Layer Security が必要] 設定を有効にしましたが、MySQL クライアントで接続にパラメータ `ssl-mode=DISABLED` が含まれていました。次のいずれかを行います。 • プロキシの [Transport Layer Security が必要] 設定を無効にします。 • MySQL クライアントで `ssl-mode=REQUIRED` の最小設定を使用してデータベースに接続します。
`ERROR 2026 (HY000): SSL connection error: Internal Server Error`	プロキシへの TLS ハンドシェイクが失敗しました。次のような原因が考えられます。 • SSL は必須ですが、サーバーではサポートされていません。 • 内部サーバーエラーが発生しました。 • 不正なハンドシェイクが発生しました。

エラー	原因または回避策
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	プロキシは、データベース接続の取得を待機中にタイムアウトしました。次のような原因が考えられます。 • 最大接続数に達したため、プロキシはデータベース接続を確立できません • データベースが使用不能であるため、プロキシはデータベース接続を確立できません。

RDS for PostgreSQL での RDS Proxy の問題のトラブルシューティング

PostgreSQL プロキシへの接続中に次の問題が発生することがあります。

エラー	原因	ソリューション
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	ユーザーが、PostgreSQL クライアントで `sslmode=disable` を設定して IAM 認証を使用してデータベースに接続しようとしました。	ユーザーは、PostgreSQL クライアントで `sslmode=require` の最小設定を使用して、データベースに接続する必要があります。詳細については、PostgreSQL の SSL サポートに関するドキュメントを参照してください。
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	このロールには Secrets Manager シークレットはありません。	このロールの Secrets Manager シークレットを追加します。詳細については、「RDS Proxy の IAM 認証の設定」を参照してください。
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	プロキシへの接続に使用されているデータベースクライアントが、プロキシで現在サ	IAM 認証を使用していない場合は、MD5 または SCRAM パスワード認証を使用してください。

エラー	原因	ソリューション
	ポートされていない認証メカニズムを使用しています。	
`ERROR 28000: A user name is missing from the connection startup packet. Provide a user name for this connection.`	プロキシへの接続に使用されているデータベースクライアントが、接続の確立を試みるときにユーザー名を送信していません。	選択した PostgreSQL クライアントを使用してプロキシへの接続を設定するときは、必ずユーザー名を定義してください。
`ERROR 28000: IAM is allowed only with SSL connections.`	クライアントが IAM 認証を使用して接続しようとしましたが、SSL が有効になっていませんでした。	PostgreSQL クライアントで SSL を有効にします。
`ERROR 28000: This RDS Proxy requires TLS connections.`	ユーザーは [Transport Layer Security が必要] オプションを有効にしましたが、PostgreSQL クライアントで `sslmode=disable` を使用して接続しようとしました。	このエラーを修正するには、以下のいずれかを行います。 • プロキシの [Transport Layer Security が必要] オプションを無効にします。 • PostgreSQL クライアントで `sslmode=allow` の最小設定を使用してデータベースに接続します。

エラー	原因	ソリューション
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	このエラーの原因としては、以下が考えられます。 • クライアントが不正な IAM ユーザー名を指定した。 • クライアントがユーザーの不正な IAM 認可トークンを指定した。 • クライアントが使用している IAM ポリシーに必要なアクセス許可がない。 • クライアントがユーザーの期限切れの IAM 認可トークンを指定した。	このエラーを修正する方法は次のとおりです。 1. 指定した IAM ユーザーが存在することを確認します。 2. IAM 認可トークンが指定した IAM ユーザーに属することを確認します。 3. IAM ポリシーに RDS への適切なアクセス許可があることを確認します。 4. 使用した IAM 認可トークンが有効であることを確認します。
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	このロールのパスワードが Secrets Manager シークレットと一致しません。	Secrets Manager でこのロールのシークレットをチェックして、パスワードが PostgreSQL クライアントで使用されているものと同じかどうかを確認します。
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	IAM 認証に使用される IAM トークンに問題があります。	新しい認証トークンを生成し、新しい接続で使用します。

エラー	原因	ソリューション
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	プロキシへの接続に使用される PostgreSQL クライアントは、3.0 より古いプロトコルを使用します。	3.0 メッセージングプロトコルをサポートする、より新しい PostgreSQL クライアントを使用します。PostgreSQL `psql` CLI を使用している場合は、バージョン 7.4 以降を使用します。
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	プロキシへの接続に使用されている PostgreSQL クライアントが、ストリーミングレプリケーションモードを使用しようとしています。このモードは、現在 RDS Proxy でサポートされていません。	接続に使用されている PostgreSQL クライアントでストリーミングレプリケーションモードをオフにします。
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	プロキシへの接続に使用されている PostgreSQL クライアントが、起動メッセージを通じて、RDS Proxy で現在サポートされていないオプションをリクエストしています。	接続に使用されている PostgreSQL クライアントで、上記のメッセージから、サポートされていないと表示されているオプションをオフにします。
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	クライアントからプロキシへの IAM 認証による同時リクエストの数が制限を超えました。	PostgreSQL クライアントからの IAM 認証を使用した接続の確立速度を下げます。
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	クライアントからプロキシへの同時接続リクエストの数が制限を超えました。	PostgreSQL クライアントからこの RDS Proxy へのアクティブな接続の数を減らします。

エラー	原因	ソリューション
`ERROR 53300: Rate of connection to proxy exceeded` *`number_value`* `.`	クライアントからプロキシへの接続リクエストのレートが制限を超えました。	PostgreSQL クライアントからの接続の確立速度を下げます。
`ERROR XX000: Unknown error.`	不明なエラーが発生しました。	AWS サポートに連絡して、問題の調査を依頼してください。
`ERROR 08000: Timed-out waiting to acquire database connection.`	プロキシは、`ConnectionBorrowTimeout` 設定で指定された期間内に、データベース接続の取得を待機中にタイムアウトしました。次のような原因が考えられます。 • 最大接続数に達したため、プロキシはデータベース接続を確立できません。 • データベースが使用できないか、設定された `ConnectionBorrowTimeout` よりもデータベースへの接続確立に時間がかかるため、プロキシはデータベース接続を確立できません。	以下の解決策が対象となります。 • プロキシ接続を固定しないでください。「RDS Proxy の固定の回避」を参照してください。 • `ConnectionBorrowTimeout` と `MaxConnectionsPercent` の設定を確認します。「RDS Proxy 接続に関する考慮事項」を参照してください。 • ターゲットの可用性を確認します。`AvailabilityPercentage` の「Amazon CloudWatch を使用した RDS Proxy メトリクスのモニタリング」を参照してください。

エラー	原因	ソリューション
`ERROR XX000: Request returned an error: database_error .`	プロキシから確立されたデータベース接続がエラーを返しました。	解決策は、具体的なデータベースエラーによって異なります。1 つの例は、`Request returned an error: database "your-database-name" does not exist` です。これは、指定されたデータベース名がデータベースサーバーに存在しないことになります。または、データベース名として使用されているユーザー名 (データベース名が指定されていない場合) がサーバーに存在しないことになります。
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	クライアントからプロキシへの IAM 認証による同時リクエストの数が制限を超えました。	PostgreSQL クライアントからの IAM 認証を使用した接続の確立速度を下げます。
`ERROR 28000: Enable IAM authentication for the client connection to the proxy and try again.`	プロキシへのクライアント接続で IAM 認証が有効になっていないため、RDS Proxy はデータベースに接続できません。これは、プロキシの `DefaultAuthScheme` パラメータが登録済みユーザーで `IAM_AUTH` に設定されているが、クライアントが IAM 認証の代わりにパスワード認証を使用している場合に発生します。	プロキシへのクライアント接続の IAM 認証を有効にして、もう一度試してください。

エラー	原因	ソリューション
`ERROR 28000: Configure IAM authentication as the DefaultAuthScheme in your proxy and try again.`	`DefaultAuthScheme` が `IAM_AUTH` に設定されていないため、RDS Proxy はデータベースに接続できません。プロキシの `DefaultAuthScheme` パラメータは `NONE` に設定されていますが、クライアントは IAM 認証を使用しようとしています。	プロキシの `DefaultAuthScheme` を `IAM_AUTH` に設定して、もう一度試してください。

削除された **postgres** データベースのトラブルシューティング

postgres データベースを誤ってインスタンスから削除した場合は、データベースを復元してインスタンスへの接続を回復する必要があります。以下のコマンドを DB インスタンス内で実行します。

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

RDS Proxy の AWS CloudFormation での使用

RDS Proxy は、AWS CloudFormation で使用できます。そうすることで、関連するリソースのグループを作成しやすくなります。このようなグループには、新しく作成された Amazon RDS DB インスタンスに接続できるプロキシを含めることができます。RDS Proxy の CloudFormation でのサポートには、`DBProxy` および `DBProxyTargetGroup` の 2 つの新しいレジストリタイプが含まれます。

以下のリストは、RDS Proxy のサンプル CloudFormation テンプレートを示しています。

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
   Properties:
     DBProxyName: CanaryProxy
     EngineFamily: MYSQL
     RoleArn:
```

```
      Fn::ImportValue: SecretReaderRoleArn
    Auth:
      - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
    VpcSubnetIds:
      Fn::Split: [",", "Fn::ImportValue": SubnetIds]

ProxyTargetGroup:
  Type: AWS::RDS::DBProxyTargetGroup
  Properties:
    DBProxyName: CanaryProxy
    TargetGroupName: default
    DBInstanceIdentifiers:
      - Fn::ImportValue: DBInstanceName
  DependsOn: DBProxy
```

このサンプルのリソースの詳細については、「DBProxy」と「DBProxyTargetGroup」を参照してください。

CloudFormation を使用して作成できるリソースの詳細については、「RDS リソースタイプのリファレンス」を参照してください。

ブルー/グリーンデプロイでの RDS Proxy の使用

Amazon RDS Proxy は、ブルー/グリーンデプロイとともに使用することで、スイッチオーバーのダウンタイムを短縮できます。RDS Proxy は、スイッチオーバープロセスを把握し、グリーン環境がアクティブな本番環境になるとグリーン環境への接続を自動的にリダイレクトすることで、データベース移行中に通常発生する DNS 伝播の遅延を排除します。RDS Proxy は、RDS for PostgreSQL、RDS for MySQL、RDS for MariaDB のブルー/グリーンデプロイをサポートしています。アプリケーションコードの変更やカスタムクライアントドライバーは必要ありません。

RDS Proxy を使用する場合のスイッチオーバー

ブルーインスタンスが RDS Proxy にアタッチされると、スイッチオーバー中に次のプロセスが発生します。

- スイッチオーバーガードレール
  - Amazon RDS は追加のガードレールチェックを実行して、プロキシがブルー環境とグリーン環境の両方に正常に到達でき、スイッチオーバーの準備が整っていることを検証します。
- スイッチオーバー中のアプリケーショントラフィックルーティング

スイッチオーバーアクションのリストについては、「スイッチオーバーアクション」を参照してください。RDS Proxy を介してスイッチオーバーを行う場合は、次の考慮事項に注意してください。

- スイッチオーバー中、ブルーデータベースは、グリーン環境が昇格する前に読み取り専用モードになります。RDS Proxy は、この移行期間中もブルーデータベースへの接続のルーティングを続行します。
- この期間中の RDS for MySQL の書き込みオペレーションは、読み取り専用エラーを返す可能性があります。例: `1290 (HY000): The MySQL server is running with the –read-only option` このステートメントを実行することはできません。RDS for PostgreSQL の読み取りクエリと書き込みクエリでは、管理者コマンドにより接続が終了しているため、AdminShutdown が返されます。
- スイッチオーバーが検出されると、プロキシは新しく昇格されたグリーン環境にトラフィックを自動的にルーティングします。
- グリーン環境が新しいライターとして昇格されると、プロキシへの既存の接続は削除されます。アプリケーションは、昇格の完了後に接続を再確立する必要があります。
- RDS Proxy の CloudWatch ログを確認して、この移行動作がいつ発生したかを確認できます。

- RDS Proxy API の動作
  - `describe-db-proxy-targets` などのプロキシ API は、トラフィックルーティングが以前に発生した場合でも、スイッチオーバーが完全に完了した後にのみ、更新されたターゲットを反映します。

制限事項

- ブルー/グリーンデプロイを作成する前に、ブルーインスタンスがプロキシのターゲットになっている必要があります。そのインスタンスのブルー/グリーンデプロイが作成された後、RDS Proxy にブルーインスタンスを追加することはできません。

RDS Proxy のベストプラクティス

このセクションでは、Amazon RDS Proxy のベストプラクティスと使用シナリオに関するガイドラインを提供します。ここで提供される情報は、さまざまなユースケースで RDS Proxy を使用する利点を理解し、最良の結果を得るためにアプリケーションとデータベースの設定を最適化するのに役立ちます。

トピック

- 一般的な使用シナリオ
- アプリケーションとワークロードに関する考慮事項
- 設定のガイドライン

一般的な使用シナリオ

トピック

- アプリケーションのスケーラビリティ
- 高可用性
- 1 つのターゲットに対して複数のプロキシを使用する

アプリケーションのスケーラビリティ

サーバーレスおよびイベント駆動アプリケーションでの接続処理

AWS Lambda を基盤とする API やウェブサービスなどのサーバーレスおよびイベント駆動アプリケーションは、多くの場合、多数の短期間のクライアントリクエストをサポートする必要があります。この使用パターンにより、アプリケーション側で接続プーリングを実装できないため、データベース側で接続チャーンが発生する可能性があります。同時接続の数だけが原因でデータベースのパフォーマンスが低下する可能性があり、データベースが接続制限を超えるとクライアント側エラーが発生する可能性があります。

これらのシナリオでは、RDS Proxy には次の利点があります。

1. データベースからプロキシへの接続を確立するコストをオフロードし、接続プーリングと多重化によって、多数のクライアント接続をはるかに少数のバックエンドデータベース接続に変換します。これにより、特に接続の確立と維持のコストが比較的高い PostgreSQL などのデータベースエンジンでは、接続のオーバーヘッドとデータベースの競合を軽減するのに役立ちます。
2. 接続の急増をより適切に処理できます。例えば、データベースが接続制限を超えると、即座にクライアントにエラーが返されます。RDS Proxy がプールから接続を借用する必要があるが、プールが既に満杯の場合、プロキシは接続が利用可能になるまで待機できます。この機能により、重大なエラーをクエリレイテンシーのわずかな増加に変えることで、クライアントエクスペリエンスを向上させることができます。
3. 接続プールのサイズを設定できるため、RDS Proxy をスロットリングまたは負荷分散メカニズムとして使用することもできます。接続数が指定した制限を超えると、RDS Proxy は設定可能な

タイムアウト内に接続が利用可能になるまで待機します。これは、データベースが複数のワークロードを処理し、特定のワークロードがデータベースに与える負荷の量を制限する場合に便利です。

コンテナベースの分散アプリケーションでの接続処理

コンテナベースの分散アプリケーションアーキテクチャには、数百または数千のコンテナがあり、それぞれでアプリケーションコードのコピーが実行される場合があります。個々のコンテナが接続プーリングに対応していたとしても、これらのプールはコンテナ固有であるため、非常に小さくなります。コンテナの数に各コンテナミニプールのサイズを掛けると、Amazon RDS または Aurora データベースの接続制限を超える可能性があります。

この状況では、RDS Proxy が接続プーリング (接続の再利用) と多重化 (1 つのバックエンド接続を使用して複数のクライアントに提供する) を実行する能力が最も価値を発揮します。各コンテナ内の接続プールを使用して、アプリケーションスレッドと RDS Proxy 間のチャーンを減らすことはできますが、プロキシはバックエンドデータベースへの接続の数を管理可能なレベルに減らすのに役立ちます。

リードレプリカの使用率を向上させる

読み取り量の多いデータベースでは、読み取り専用トラフィックをサポートするために複数のリードレプリカが必要になる場合があります。アプリケーションは独自のロジックを使用して、どのレプリカに接続するかを選択できます。より一般的には、Aurora クラスターリーダーエンドポイントなどの DNS ベースのラウンドロビンメカニズムを使用します。ただし、DNS ベースのアプローチでは、DNS キャッシュの結果としてレプリカ使用率が不均等になる可能性があります。例えば、クライアントが特定のレプリカに「ピン」接続したり、クラスターに追加された新しいレプリカを認識できなかったり、存在しなくなったレプリカに接続しようとしたりする可能性があります。

RDS Proxy 読み取り専用エンドポイントを使用する場合、プロキシは「未処理の接続が最も少ない」というロジックを使用して、使用可能なすべてのレプリカ間でクライアント接続をルーティングします。RDS Proxy は、CPU 使用率などのデータベースメトリクスに基づいてトラフィックの負荷分散を行うのではなく、データベースの接続制限に基づいて重み付けして、各レプリカのクライアント接続の数を均等化しようとします。例えば、`max_connections` の設定がそれぞれ 500、500、1000 の 3 つの Aurora レプリカが実行されている場合、プロキシは他の 2 つのレプリカと比較して、3 番目のレプリカに対して約 2 倍の接続数を送信しようとします。

RDS Proxy リーダーエンドポイントは、Aurora クラスターまたは 2 つの読み取り可能なスタンバイを持つ Amazon RDS マルチ AZ DB クラスターデプロイで使用できます。プロキシリーダーエンド

ポイントは、リードレプリカを使用する Amazon RDS DB インスタンスのデプロイではサポートされていません。

接続効率の向上

クライアントアプリケーションとデータベース間にプロキシを導入する場合、通常、接続処理の効率を高めることを目的とすると同時に、プロキシを介した追加のネットワークホップによるレイテンシーコストも考慮する必要があります。接続効率を向上させるために中間レイヤーを追加することは、一見すると直感に反しているように思えるかもしれません。なぜなら、データベースに対して直接開かれていた接続がプロキシに対して開かれるようになるためです。プロトコルハンドシェイクのステップはどちらの場合も同じであるため、接続ハンドシェイクにリソースを消費し続けている状況では、効率改善がどこから来ているのか分かりにくいかもしれません。

プロキシを使用しても、必ずしも接続確立コストが安くなるわけではありません。代わりに、ハンドシェイク処理の負担のほとんどをデータベースレイヤーからプロキシレイヤーに移動します。データベースリソースの料金を支払うのであれば、そのリソースを補助的なオーバーヘッドではなく、データベース作業に費やす必要があります。これは、暗号化された接続を使用する場合に特に重要になります。既存の接続を介して暗号化されたデータを送信するオーバーヘッドはそれほど大きくありませんが、暗号化された接続のセットアップのための初期オーバーヘッドは相当なものです。1 秒あたり数百または数千の接続を処理するような環境では、そうした余分な労力がすぐに蓄積される可能性があります。CPU 時間を (比較的高価な) データベースリソースに消費するのではなく、(比較的安価な) プロキシレイヤーにオフロードするほうが良いかもしれません。

追加のネットワークホップによって生じるレイテンシーについては、アプリケーションがどの程度「おしゃべり」であるか、およびデータベースとの「会話」ごとに実行されるステートメントの数によって異なります。RDS Proxy では、通常、1 桁ミリ秒程度のレイテンシーが増加しますが、必ずしもアプリケーションに目に見える影響を与えるとは限りません。例えば、次のようになります。

- クエリ実行時間が数十または数百ミリ秒 (またはそれ以上) のワークロードでは、プロキシのオーバーヘッドは、合計クエリ時間のごく一部であるため、それに気付く可能性はほとんどありません。
- 1 桁ミリ秒または 1 ミリ秒未満のクエリを実行するアプリケーションでは、クエリのオーバーヘッド (クエリごとに 1 つの追加のネットワークホップ) がクエリ実行時間と比較してかなり大きいため、違いが顕著に現れる可能性があります。クライアントセッションで少数のクエリのみを実行する場合であれば、累積オーバーヘッドが小さいため、これは問題ではない可能性があります。

お客様の状況において、追加されるレイテンシーが顕著で望ましくない場合は、プロキシを使用することによるその他の利点 (プーリング、多重化、フェイルオーバー処理) と比較検討する必要があります。

高可用性

Amazon RDS および Aurora (Aurora DSQL を除く) で実行されているマルチ AZ データベースは、フェイルオーバーメカニズムを使用して、プライマリデータベースインスタンスに問題がある場合に可用性を復元します。フェイルオーバーは、プライマリインスタンスのコンピューティングのスケーリングなど、運用ワークフローの一部としても使用されます。フェイルオーバーでは、DNS の変更によってプライマリ (読み取り/書き込み可能) データベースエンドポイントを、以前のプライマリインスタンスから新たに昇格したインスタンスに移動します。この DNS の変更は、クライアントが遅延なくプライマリをフォローできるように、クライアントアプリケーションによって監視および認識される必要があります。

一部のアプリケーションは、オペレーティングシステムまたはアプリケーションレベルでの DNS キャッシュが原因で、DNS の変更をタイムリーに認識できない場合があります。アプリケーションレベルで DNS キャッシュの問題に対処するのがベストプラクティスですが、アプリケーションの複雑さや変更の導入コストが原因で、必ずしも実行可能とは限りません。

このシナリオでは、RDS Proxy は DNS 関連のフェイルオーバーの遅延を回避する効果的な方法です。RDS Proxy が提供する読み取り/書き込みエンドポイントとオプションの読み取り専用エンドポイントは、データベースインスタンスがロールを交換するときにエンドポイントの背後にある IP アドレスが変更されないという点で「安定」しています。RDS Proxy は、バックエンドデータベースのトポロジを継続的に追跡し、インスタンスロールの変更をクライアントから抽象化します。

DNS キャッシュの問題を処理する別の方法として、DNS に依存することなくデータベーストポロジを直接検出できる高度なドライバーを使用する方法などがあります。ただし、アプリケーションレベルでコード変更や新しいドライバーを導入する代わりに、データベースの前に単一のプロキシをデプロイする方が簡単な場合があります。

読み取り可用性

データベースフェイルオーバー中のクライアントエクスペリエンスの向上に加えて、RDS Proxy はリードレプリカの問題が発生した場合のアプリケーションの可用性にも役立ちます。これは、次の 2 つの方法で行われます。

1. リードレプリカが使用できなくなっても、クラスターに他のレプリカが使用可能な場合、プロキシは新しい接続を使用可能なレプリカにルーティングできます。クライアントは、DNS 伝播の遅延や DNS キャッシュについて心配する必要はありません。

2. 多重化された既存のクライアント接続の場合、プロキシはその接続から利用可能な別のレプリカに後続のクエリを送信できます。このような状況では、クライアントはバックエンドのレプリカで問題が発生したことにさえ気付かない場合があります。この手法を使用する場合、RDS Proxy はレプリケーションの遅延に関して新しいレプリカが以前のレプリカより遅れないようにします。これにより、クライアントによって送信された後続のクエリは、古くなったと考えられるデータを参照することがなくなります。

1 つのターゲットに対して複数のプロキシを使用する

ベストプラクティスとして、Amazon RDS インスタンスや Aurora クラスターなどのデータベースターゲットごとに 1 つの RDS Proxy を使用します。これはほとんどのシナリオでうまく機能し、設定をシンプルに保ち、クライアントエクスペリエンスをより予測可能なものにします。これに対して、複数のプロキシを使用する場合は、予期しない動作を避けるために、すべてのプロキシ間で設定を慎重に調整する必要があります。例えば、すべてのプロキシプールの合計サイズが単一のデータベースターゲットの接続制限を超えないようにする必要があります。

特定の状況では、複数のプロキシを使用するというアイデアを試すこともできます。以下のセクションでは、最も一般的なシナリオについて説明し、単一プロキシとマルチプロキシアプローチの長所と短所について概説します。

プロキシの可用性

RDS Proxy のインフラストラクチャは可用性が高く、複数のアベイラビリティーゾーン (AZ) にデプロイできます。プロキシ容量は、継続的なヘルスモニタリングを備えた多数のノードに分散され、プロビジョニングされたインスタンスサイズまたはデータベースの Serverless v2 最大 ACU 設定に基づいて自動的に調整されます。プロキシの分散マルチ AZ 設計により、高可用性を実現するために Amazon RDS または Aurora クラスターの前に複数のプロキシを実行する必要はありません。

実際に、1 つのデータベースターゲットの前に複数のプロキシを使用すると、アプリケーションスタックはプロキシに組み込まれている堅牢なメカニズムに依存するのではなく、問題を検出して対応する責任を担うようになります。したがって、可用性を高めるために複数のプロキシを使用することは、解決できるよりも多くの問題を引き起こす可能性があるため、強く推奨されません。

複数のクライアントプールを処理する

各プロキシは、1 つの読み取り/書き込みエンドポイントと (オプションで) 追加の読み取り/書き込みエンドポイントまたは読み取り専用エンドポイントを提供します。単一のプロキシを使用して複数のクライアントグループまたは複数のワークロードを処理すると、それらのワークロードが干渉する可能性があります。例えば、あるワークロードが接続プールを独占してしまい、他のワークロードを処

理するのに十分な空き接続が残らなくなる可能性があります。ワークロードを分離するために個別のプロキシを使用することは有力なソリューションとなり得ますが、その前に他のオプションを検討することができます。

- データベース自体が、複数のプロキシを実行するよりも簡単な代替手段を提供する場合があります。例えば、特定のユーザーがプールを独占したことが原因で問題が発生した場合、データベースのアクセス許可システムを使用して、ユーザーが開くことができる同時接続の数を制限できます。
- 同様に、データベースレベルのクエリタイムアウトとアイドルタイムアウトは、孤立した接続やランナウェイクエリの問題に対処するのに役立ちます。
- 同時接続数ではなく、クエリまたはトランザクションの実行時間、またはクエリごとのリソース消費が原因で問題が発生した場合、プロキシはクエリまたはトランザクションの重みに制限を課さないため、追加のプロキシの使用は役に立ちません。ソースで問題に対処できない場合は、問題のあるクライアントを別のプロキシに移動する代わりに、データベースのイベントスケジューラを使用して、任意の条件に基づいてクライアントのアクティビティを検出して終了するコードを実行できます。

同じターゲットの前に複数のプロキシを使用する場合は、すべての接続プールの合計サイズがターゲットデータベースの容量を超えないようにしてください。例えば、すべてのプロキシの MaxConnectionsPercent の合計は 100 未満である必要があります。そうしないと、プロキシはデータベースが処理するように設定されているよりも多くの接続を開こうとする可能性があります。

各プロキシは個別に課金され、請求レートはワークロードアクティビティではなく、基盤となるデータベースリソースのサイズによって決まります。複数のプロキシを実行するコストと、代替ソリューションがある場合は、それを導入するコストを考慮してください。

読み取りプールと書き込みプールを個別に制御する

各プロキシは、1 つの読み取り/書き込みエンドポイントと (オプションで) 追加の読み取り/書き込みエンドポイントまたは読み取り専用エンドポイントを提供しますが、制限とタイムアウトはプロキシエンドポイントごとに個別に設定されるのではなく、プロキシターゲット全体に設定されます。

例えば、複数のリードレプリカとプロキシのリーダーエンドポイントを使用して大量の読み取り専用クエリを処理する Aurora クラスターがある場合でも、読み取り/書き込み接続の数を制限して単一ライターのリソース負荷を軽減したい場合があります。MaxConnectionsPercent 設定はプロキシターゲット (クラスター) 全体に対して定義されているため、読み取り専用エンドポイントの制限に影響を与えることなく、読み取り/書き込みエンドポイントに対する接続数を制限することはできません。

この課題にアプローチする方法は複数あります。

クラスターで追加のリードレプリカを起動し、プロキシの MaxConnectionsPercent 設定を比例的に減らすことができます。これにより、読み取り専用接続プールの合計サイズは維持されつつ、書き込みで許可される最大接続数が削減されます。ただし、これはクラスターのコストを増加させるだけでなく、レプリカの数が増えるほど効果が徐々に低下します。

インスタンスレベルのパラメータグループを使用して、ライターとは別に、リードレプリカのデータベース接続制限 (Aurora MySQL や PostgreSQL の max_connections など) を設定できます。ただし、非対称なパラメータ設定の使用は避ける必要があります。フェイルオーバー中にレプリカがライターロールに昇格される可能性があるため、初期のパラメータの差異が永続的なものにならないからです。フェイルオーバー中にどのレプリカが昇格対象として選択されるかに影響を与えるために、インスタンスレベルのフェイルオーバー優先度設定を変更し、それを緩やかなフェイルオーバー除外メカニズムとして使用することで、非対称な設定をより予測可能にすることができます。ただし、フェイルオーバーの優先順位はヒントとしてのみ機能し、フェイルオーバー動作の確実な保証にはなりません。したがって、非対称設定のマルチインスタンス設定は、継続的なモニタリングが必要であり、フェイルオーバーが意図しないインスタンスで発生した場合に予期しない動作が発生する可能性があるため、推奨されません。

このシナリオでは、複数のプロキシを使用することが、読み取りプールと書き込みプールを個別に管理する最もクリーンな方法でとなる可能性があります。単一のデータベースターゲットに 2 つのプロキシを作成し、アプリケーションが最初のプロキシのライターエンドポイントと 2 番目のプロキシの読み取り専用エンドポイントを使用するように設定できます。一方のプロキシは書き込みワークロードのみを処理し、もう一方のプロキシは読み取りワークロードのみを処理します。また、MaxConnectionsPercent (および他の設定) は、プロキシごとに個別に設定できます。このソリューションでは、2 つ目のプロキシを実行するために追加コストがかかりますが、前述の代替方法よりもシンプルで予測可能です。

アプリケーションとワークロードに関する考慮事項

トピック

- マルチテナントおよびマルチユーザー環境
- 複数のアプリケーションまたはソフトウェアスタックに対応するデータベース
- RDS Proxy でのアプリケーションレベルのプーリングと高度なドライバーの使用

マルチテナントおよびマルチユーザー環境

スケーラビリティと接続管理の改善に関して、RDS Proxy を使用する利点は接続プーリングを実行できること、そしてそれ以上に接続多重化を実行できることに大きく依存しています。接続プーリングは、接続の開閉に関連するオーバーヘッドを削減します。接続多重化により、プロキシはトランザクション後にバックエンドデータベース接続を再利用できます。詳細については、「RDS Proxy の概念と用語」を参照してください。

接続を多重化できない場合、プロキシは接続の固定と呼ばれる動作に戻ります。固定は、クライアントがセッション全体で同じ基盤となるプロキシ接続を使用することを強制される状況です。プロキシ接続はその 1 つのクライアント用に予約されているため、他のクライアントで再利用することはできません。つまり、固定により、クライアントとプロキシの接続とプロキシとデータベースの接続の間に排他的な 1 対1 の関連付けが作成されます。固定を回避することは、主にスケーラビリティと効率の理由から RDS Proxy が使用されるすべてのシナリオで重要です。固定の最新の動作については、「RDS Proxy の固定の回避」を参照してください。

一般的なルールとして、接続の状態が同じであれば、接続を多重化できます。セッション固有のカスタム状態情報が含まれている場合、接続を多重化することはできません。セッション状態を定義する要素の 1 つは、接続に関連付けられたデータベースユーザー名です。プロキシに「user_A」として接続する場合、プロキシはバックエンドデータベース接続も「user_A」として開く必要があります。プロキシは、「user_A」としてログインする他のクライアントに対して、このバックエンド接続をプールして再利用できる可能性がありますが、別のユーザー名を使用するクライアントには再利用できません。

この動作により、一意のデータベースアカウントが多数あるマルチユーザー環境において、プーリングと多重化の効率が大幅に低下する可能性があります。これは、データベースレベルまたはスキーマレベルのマルチテナンシーを使用するアーキテクチャで特に当てはまります。データベースに 1,000 個のスキーマ (テナントごとに 1 つ) が含まれており、各テナントが異なるユーザー名でデータベースに接続すると、接続プールはユーザー固有のマイクロプールに断片化され、全体的な効率が低下します。

さらに、データベースエンジンに固有の側面は、プーリング効率とプロキシの接続を多重化する機能にさらに影響を与える可能性があります。

1. Amazon RDS と Aurora PostgreSQL では、テナントごとに 1 つのデータベースを使用してマルチテナンシーを実装することがよくあります。ただし、PostgreSQL では、接続はデータベース固有です。あるデータベースに対して開かれた接続は、他のデータベースのデータにアクセスできません。したがって、データベースレベルのマルチテナンシーにより、プロキシレベルでのプーリングと多重化の効率が低下します。この考慮事項は、ワークロードがスキーマレベルのマ

ルチテナンシーを使用し、クライアントセッションがカスタムの search_path を使用する場合にも適用されます。ただし、すべてのセッションがデフォルトの検索パスを使用し、完全修飾名 (schema_name.table_name) を使用してテーブルを参照する場合は、それらのセッションを多重化できます。

2. Amazon RDS と Aurora MySQL では、「データベース」と「スキーマ」という用語は同義語です。マルチテナンシーは、多くの場合、テナントごとに 1 つのスキーマを使用して実装されます。MySQL では、テナントごとに 1 つのデータベースを使用することと同じです。接続は MySQL サーバー全体に対して開かれ、スキーマには関連付けられません。アプリケーションがスキーマレベルのマルチテナンシーを使用する場合、同じデータベースユーザー名を使用するクライアントは、それらの接続が異なるスキーマのデータにアクセスする必要がある場合でも、接続の多重化が可能です。多重化は、テナントごとに異なるデータベースアカウントを使用する代わりに、テナントの分離がアプリケーションレベルで行われる場合に最も効果的です。

マルチスキーマ環境では、多重化の効率はテーブル名をどのように参照するかによって異なります。

- セッション変数 (PostgreSQL では SET search_path ...、MySQL では USE schema;) を使用して現在のスキーマを選択し、クエリで非修飾テーブル名 (SELECT ... FROM table_name など) を使用するクライアントの場合、接続多重化は、同じスキーマまたは同じ検索パスを使用するクライアント間でのみ機能します。
- セッション状態を変更して現在のスキーマを定義するのではなく、SQL ステートメントで完全修飾テーブル名 (SELECT ... FROM schema_name.table_name など) を使用するクライアントの場合、多重化は同様に制約されません。

複数のアプリケーションまたはソフトウェアスタックに対応するデータベース

前のセクションで説明したように、特定の接続状態の特性によって固定が発生することはありませんが、異なるクライアント間での接続の再利用におけるプロキシの機能は低下します。MySQL ターゲットで使用すると、RDS Proxy は、文字セット、タイムゾーン、照合設定など、セッション状態を設定する多数のステートメントとセッション変数を追跡します。クライアントが追跡されたステートメントまたは変数を使用してセッション設定を構成する場合、プロキシ接続は、それらの設定に同じ値を持つ他のクライアントに対してのみ再利用できます。

その結果、特定のアプリケーションやドライバーの動作により、プロキシ内の接続を再利用できなくなる可能性があります。例えば、プロキシがそれらのアプリケーション間の接続を再利用および多重化できると仮定すれば、異なるアプリケーションが同じユーザー名を使用してデータベースに接続することを許可できるかもしれません。ただし、あるアプリケーションがタイムゾーン A (SET

time_zone = ?) との接続をブートストラップし、別のアプリケーションがタイムゾーン B を使用する場合、接続はアプリケーション内で再利用できますが、アプリケーション間では再利用できません。これにより、接続プールが断片化され、プールと多重化の有効性に悪影響を及ぼします。

詳細については、「RDS Proxy が RDS for MariaDB および RDS for MySQL データベースに対して追跡する内容」を参照してください。セッション状態の追跡は現在、MySQL 以外のデータベースターゲットではサポートされていません。

接続の固定を回避するためのセッション状態管理に関する設定ガイドラインとベストプラクティスについては、「設定のガイドライン」を参照してください。

RDS Proxy でのアプリケーションレベルのプーリングと高度なドライバーの使用

RDS Proxy は、アプリケーション自体が接続プーリングを使用していない状況において、スケーラビリティと接続効率の向上に役立ちます。同時に、多くのドライバーとフレームワークにはプーリング機能が含まれています。また、ドライバーレベルでプロキシの一部の機能を実装する高度なラッパーまたはドライバーを使用している場合もあります。

アプリケーションレベルのプーリングやその他の接続処理の改善策を使用しても、RDS Proxy の使用と本質的に競合することはなく、その利点が損なわれることもありません。例えば、アプリケーションコンテナで接続プーリングを使用している場合でも、コンテナの数が非常に多いため、プロキシを使用しないとデータベース接続の制限に達してしまう可能性があります。アプリケーションレベルのプールやその他の接続関連の機能と RDS Proxy を併用する場合は、高度な接続処理機能がアプリケーションレベルに存在する理由を確認して理解してください。それらの機能のうち、保持する価値がある機能 (または無害な機能) と、プロキシの動作と重複または干渉する可能性がある機能を判断します。例えば、次のようになります。

1. ドライバーやフレームワークに組み込まれたプーリング機能は、RDS Proxy 機能と重複しているように見える場合でも、有用な場合があります。アプリケーションレベルのプールがプロキシが提供する利点に加えてローカル接続の効率を向上させる場合は、それを維持できます。
2. フェイルオーバー処理に関連する機能は、RDS Proxy ロジックに干渉したり、利点をもたらさずにスタック全体の複雑さを増大させたりする可能性があります。例えば、アプリケーションが DNS 関連のフェイルオーバーの遅延を回避するために Aurora クラスターのトポロジを積極的に追跡している場合、RDS Proxy で追跡する必要はありません。このトポロジ追跡ロジックを保持すると、アプリケーションスレッドがプロキシをバイパスし、個々のデータベースインスタンスに直接接続するなど、望ましくない動作が発生する可能性があります。このシナリオでは、アプリケーションレベルの追跡ロジックを無効にし、RDS Proxy にクラスタートポロジを抽象化させることができます。

3. 接続プーリングライブラリは、理論的には有益と思われる状態管理機能を使用することがありますが、それらがプロキシの動作を妨げる場合があります。その一例として、PostgreSQL ライブラリが、借用間の接続状態をリセットするために `DISCARD ALL` クエリを呼び出すことが挙げられます。接続のリセットはプーリングと多重化に役立ちますが、Amazon RDS Proxy の内部セッション状態管理を妨げる可能性があります。`DISCARD` を使用する場合、プロキシはクライアント接続をリリース時に固定するため、多重化の効率が低下します。

保持するアプリケーションレベルの接続処理コンポーネントについては、それらの設定が Amazon RDS Proxy で使用される接続処理ロジックに干渉しないことを確認してください。例えば、次のようになります。

- スタックのすべてのレイヤーでプールのサイズ設定を合わせます。アプリケーションレベルのプールのサイズが大きすぎる (またはプロキシプールのサイズが小さすぎる) 場合、アプリケーションはプロキシが処理するように設定されていない接続を開こうとする可能性があります。これらの接続では、最良の場合でも遅延が発生し、最悪の場合は拒否エラーが発生する可能性があります。
- タイムアウト設定を調整してチャーンを減らし、接続動作に関する混乱を回避します。アプリケーションプールが接続を 300 秒間維持しているにも関わらず、プロキシが 60 秒後に接続を閉じるように設定されている場合、アプリケーションは予想される動作ではなく、早期の接続切断を経験することになります。

これらのアーキテクチャ上の決定や設定の選択には、テストと実験が必要になる場合があります。複数のレイヤーからなるプールと接続管理が存在する環境では、アプリケーションの動作を正確に予測できるとは限りません。

一般的な設定ガイドラインについては、「設定のガイドライン」を参照してください。

設定のガイドライン

このセクションでは、RDS Proxy で使用できる設定について説明し、アプリケーションスタック全体で設定を調整するためのベストプラクティスを提供します。これらは、Amazon RDS と Amazon Aurora で RDS Proxy を使用するためのガイドラインをまとめたものです。RDS 固有および Aurora 固有の注意事項は、該当する場合に明記されています。

特に明記されていない限り、「データベース」または「ターゲット」という用語は、Aurora クラスター、Amazon RDS マルチ AZ DB クラスター、または Amazon RDS インスタンスを指します。

トピック

- RDS Proxy の設定

- アプリケーション、プロキシ、データベース設定の調整
- データベース設定
- アプリケーションの構成

RDS Proxy の設定

MaxConnectionsPercent

最小値	最大値	デフォルトの値
(1、MaxIdleConnectionsPercent) のうち小さい方	100	100

詳細については、「MaxConnectionsPercent」を参照してください。

この設定では、RDS Proxy がターゲットデータベースと確立できる接続の数を、データベースで許可される最大接続数に対する割合で制限します。プロキシは必要に応じてバックエンド接続を開くため、任意の時点での実際の接続数が設定された最大数を下回る可能性があります。

`MaxConnectionsPercent` 設定がデータベース接続制限の割合である場合、プロキシの接続プールのサイズは自動的にデータベース設定に従います。つまり、データベースインスタンスのサイズを変更したり、設定を変更したりするたびに、プロキシを再設定する必要はありません。また、暗黙的か明示的かにかかわらず、データベース設定が変更される可能性があるシナリオに注意する必要があります。

- データベースのパラメータグループで接続制限 (`max_connections` パラメータなど) を明示的に設定しなかった場合、Amazon RDS と Aurora はデータベースインスタンスクラスに基づいて自動的に計算します。その結果、データベースインスタンスをスケールするときに暗黙的な接続制限が変更され、プロキシの最大プールサイズに影響する可能性があります。詳細については、「Amazon RDS ユーザーガイド」の「データベース接続の最大数」、「Aurora MySQL DB インスタンスへの最大接続数」、および「Amazon Aurora ユーザーガイド」の「Aurora PostgreSQL DB インスタンスへの最大接続数」を参照してください。
- Aurora Serverless v2 データベースインスタンスは、データベースの最大 ACU 設定に基づいて接続制限を決定します。詳細については、「Amazon Aurora ユーザーガイド」の「Aurora Serverless v2 の最大接続数」を参照してください。

ベストプラクティス:

- デフォルト設定の 100% は、プロキシを介してすべてのトラフィックを受信し、管理アクセスやその他のクライアントにヘッドルームを必要としないデータベースに適しています。
- データベースがアプリケーションから直接 (プロキシをバイパスして) トラフィックを受信し、プロキシがすべての接続を消費しないようにする場合、またはデータベース管理者による直接アクセスなど、他の目的で特定の数の接続を確保したい場合は、この設定を減らします。
- Aurora Global Database クラスターで RDS Proxy を使用し、書き込み転送が有効になっている場合は、書き込み転送に割り当てられたクォータ分だけプロキシの `MaxConnectionsPercent` 値を減らします。詳細については、「Amazon Aurora ユーザーガイド」の Aurora MySQL および Aurora PostgreSQL での書き込み転送の設定パラメータを参照してください。この推奨事項は、プロキシがグローバルデータベースのプライマリリージョンまたはセカンダリリージョンのいずれのクラスターを提供している場合でも適用されます。セカンダリクラスターはプライマリの役割に昇格できるため、グローバルトポロジ全体でプロキシ設定の一貫性を保つことをお勧めします。プライマリリージョンとセカンダリリージョンに対応するプロキシには非対称設定を使用できますが、グローバルフェイルオーバーまたはスイッチオーバーのたびにこれらの設定を調整する必要があります。
- ターゲットが複数のプロキシを提供する場合、データベースがオーバーサブスクライブされないように、これらのプロキシ全体の `MaxConnectionsPercent` の合計値は 100 を超えることはできません。設定と管理を簡素化するために、ターゲットごとに 1 つのプロキシを使用することをお勧めします。特に、冗長性のためにデータベースごとに複数のプロキシを使用する必要はありません。詳細については、「1 つのターゲットに対して複数のプロキシを使用する」を参照してください。

デフォルトの `MaxConnectionsPercent` 設定とカスタム値のどちらを使用している場合でも、ピーク時に許可される接続数と予想されるクライアント接続の最大数の間に少なくとも 30% の余裕を確保してください。例えば、次のようになります。

- プロキシがデータベース用に設定された接続制限の最大 50% を必要とすると思われる場合は、`MaxConnectionsPercent` の設定を少なくとも `1.3 * 50% = 65%` に設定します。
- デフォルトの `MaxConnectionsPercent` 設定の 100 を使用する場合は、データベースの制限自体に十分なヘッドルームがあることを確認してください。

この追加ヘッドルームにより、予期しないワークロードの急増時にもクライアントエクスペリエンスが向上し、RDS Proxy が内部インフラストラクチャ全体に接続を再分散して、熱管理やその他の目的に役立てることができます。

MaxConnectionsPercent を 1 という低い値に設定できますが、インスタンスタイプに基づいて以下の最小値を使用することをお勧めします。

- db.t3.small: 100
- db.t3.medium: 55
- db.t3.large: 35
- db.r3.large またはそれ以上: 20

MaxIdleConnectionsPercent

最小値	最大値	デフォルト値 (SQL Server)	デフォルト値 (他のエンジン)
(ゼロ)	MaxConnectionsPercent	MaxConnectionsPercent の 5%	MaxConnectionsPercent の 50%

詳細については、「MaxIdleConnectionsPercent」を参照してください。

この設定は、RDS Proxy が接続プールに保持するアイドル状態のデータベース接続数を、データベースで許可される接続の最大数に対するパーセンテージで制限します。データベース (バックエンド) 接続は、接続に対するアクティビティが 5 分間なかった場合、アイドル状態とみなされます。この設定は、プロキシとバックエンドデータベース間の接続に適用されます。

次の点に注意してください。

- この設定は、プール内のアイドル接続の数を制限しますが、プロキシが一定数のアイドル接続を保持するように強制することはありません。クライアントアクティビティが非常に少ない場合、バックエンドデータベース接続の実際の数は MaxIdleConnectionsPercent を下回る可能性があります。
- プロキシの接続プールで再利用できる場合、接続はアイドル状態としてカウントされます。固定接続は他のクライアントで再利用できないため、MaxIdleConnectionsPercent の適用においてはアイドル状態としてカウントされません。固定の詳細については、「RDS Proxy の固定の回避」を参照してください。
- データベースのメタデータによって報告されるアイドル接続の数は、通常、RDS Proxy のメトリクスによって記録されたアイドル接続の数よりも多くなります。これは、プロキシをバイパスする

直接クライアント接続と、Amazon RDS および Aurora オートメーションで使用される内部接続が原因である可能性があります。

> **Note**
>
> プロキシレイヤーで観察および適用される接続アイドル時間は、MySQL のプロセスリストや PostgreSQL のアクティビティ統計テーブルなどのデータベースツールによって報告されるアイドル時間とは異なる場合があります。RDS Proxy はバックエンド接続に ping を送信することがあります。これにより、クライアントアクティビティに関して接続がアイドル状態のままであっても、データベースアイドルタイマーがリセットされます。

ベストプラクティス:

デフォルト設定の 50 は、ほとんどのワークロードに適しており、推奨されます。設定を変更すると、次の効果があります。

- `MaxIdleConnectionsPercent` を上げると、データベースはより多くのアイドル接続を検出し、ピーク時間外にデータベースリソースの消費量が増加する可能性があります。一方、プロキシを介して処理される接続の急増は、プールですぐに利用できる接続が多いため、借用レイテンシーが低くなります。
- `MaxIdleConnectionsPercent` を下げると、プロキシはアイドル状態の接続をより積極的に閉じ、それらの接続による競合とリソースの消費を減らす可能性があります。ただし、プロキシを通過する接続の急増により、借用時間が長くなる可能性があります。プールで使用できる接続が少ないため、プロキシは急増中に新しいバックエンド接続を開くために余分の時間を費やす必要があります。

プロビジョニングされたインスタンスタイプを使用するデータベースでは、アイドル状態の接続によるリソース消費が重要な懸念事項ではない場合がありますが、Serverless v2 インスタンスタイプを使用する Aurora データベースでは、アイドル状態のリソース消費を最適化してコストを削減することが望ましい場合があります。

IdleClientTimeout

最小値	最大値	デフォルトの値
1 分	8 時間	30 分

詳細については、「IdleClientTimeout」を参照してください。

この設定は、プロキシがクライアント接続を閉じる前にアイドル状態を維持できる時間を決定します。RDS Proxy は、接続がアイドル状態であるかどうかにかかわらず、最大接続有効期間を 24 時間とすることに注意してください。例えば、IdleClientTimeout を 30 分 (デフォルト) に設定し、1 分ごとにデータベースに ping を実行する場合、接続がアイドルタイムアウトを超えることはありませんが、無期限に開いたままになることはありません。RDS Proxy は、アイドル状態とみなされない場合でも、24 時間後に接続を閉じます。

IdleClientTimeout 設定は、クライアント (アプリケーションまたはインタラクティブユーザー) と RDS Proxy 間の接続に適用されます。RDS Proxy とデータベース間のバックエンド接続のアイドル状態の動作については、「MaxIdleConnectionsPercent」を参照してください。

ベストプラクティス:

- アイドルタイムアウトは、クライアント接続またはトランザクション内の SQL アクティビティの通常の一時停止に対応するのに十分な長さである必要があります。これは、クライアントが実質的には不要になったリソースを保持し続け、そのリソースを必要とする他のクライアントが利用できなくなるほど長く設定するべきではありません。これは、あるクライアントによって固定された接続を他のクライアントで再利用できないため、接続の固定が見られる場合に特に重要です。
- RDS Proxy がタイムアウトを正しく適用するには、クライアント接続がクエリ (SELECT 1; などの単純なヘルスチェックを含む) やプロトコル ping (MySQL の COM_PING など) を送信せずに、本当にアイドル状態である必要があります。タイムアウトを超過しても接続が閉じられない場合は、アプリケーションドライバーの接続ロジックを確認してください。アプリケーションレベルの接続プールは、独自のライブネスチェックを実行する可能性が高く、IdleClientTimeout と干渉する可能性があります。

ConnectionBorrowTimeout

最小値	最大値	デフォルトの値
(ゼロ)	5 分	2 分

詳細については、「ConnectionBorrowTimeout」を参照してください。

クライアントが RDS Proxy に接続する場合、プロキシはプールから既存の利用可能な接続を借用するか、新しいデータベース接続を開く必要があります。この設定では、RDS Proxy が接続を借用または開くのを待つ時間を定義します。この時間を過ぎるとエラーを返します。

次の点に注意してください。

- ConnectionBorrowTimeout に 0 を設定すると、接続プールに使用可能な接続がまだ含まれていない場合、タイムアウトエラーが発生します。これは、プールが最大容量を下回っていて、新しいバックエンド接続を開く可能性がある場合でも当てはまります。
- MaxIdleConnectionsPercent が MaxConnectionsPercent と等しい場合でも、プール内の実際の接続数は、設定された最大値を下回る可能性があります。つまり、MaxIdleConnectionsPercent はアイドル状態の接続の量を制限しますが、接続を強制的に開いたままにすることはありません。

接続プールが最大容量を下回って稼働するのは正常です。この状況では、ConnectionBorrowTimeout の設定を 0 にすることで、プールが新しい接続が開かれるのを待つことができないため、プールが増大するのを防ぐことができます。そのため、前述の動作が優先されない限り、すべてのワークロードにおいて、ConnectionBorrowTimeout の値をゼロ以外に設定する必要があります。

> **Note**
>
> この設定は、オフラインメンテナンスオペレーションやフェイルオーバー中など、データベースが接続を受け入れる準備ができていない場合にも適用されます。ConnectionBorrowTimeout を適用するロジックは、データベースが稼働中か停止中かにかかわらず同じです。

初期化クエリと固定フィルター

RDS Proxy は、接続の固定を減らし、結果として多重化の効率を向上させることができる追加機能をサポートしています。

初期化クエリは、プロキシが新しいバックエンドデータベース接続を設定するたびに実行される 1 つ以上のステートメントです。クライアントが同じステートメントを使用してセッションパラメータを設定する場合は、それらのステートメントをプロキシの初期化クエリに移動できます。これにより、固定の可能性が減り、効率も向上します。特定のバックエンドデータベース接続は、セットアップ中に初期化クエリを 1 回実行しますが、その後、多くのクライアントによって再利用される可能性があります。SQL ステートメントを初期化クエリに含めても、クライアントトラフィックから除外されないことに注意してください。アプリケーションコードからこれらのステートメントを削除して、多重化を妨げないようにする必要があります。

セッション固定フィルターは、プロキシが特定のセッション状態を固定できないようにする設定プロパティです。現在利用可能な唯一のフィルターオプションである EXCLUDE_VARIABLE_SETS は、セッションを固定するかどうかを決定するときに、すべての SET ステートメントを無視するようにプロキシに指示します。SET ステートメントは引き続きデータベースに渡され、セッション状態に影響する可能性があります。つまり、このオプションは、次の状況でのみ安全です。

1. SET ステートメントが何もしない場合、例えばシステム変数をサーバーのデフォルトと同じ値に設定している場合です。
2. SET ステートメントと後続のクエリは同じトランザクションの一部であり、すべてのトランザクションは、他のトランザクションによって設定された変数の影響を受けないように、完全に独立して独自の状態を設定します。

> **Note**
>
> EXCLUDE_VARIABLE_SETS 固定フィルターはすべてか無かの設定であり、無視する SET ステートメントを個別に選択することはできません。ユースケースが前のリストで説明したカテゴリのいずれかに該当しない限り、このフィルターを固定の包括的な解決策として使用しないでください。

最良の結果を得るには、可能な限りアプリケーションコードから不要なステートメントを削除し、アプリケーションの変更が不可能な場合にのみフィルターを使用します。これにより、そもそも必要のないステートメントに特別な処理を適用するのではなく、ノイズが少なく予測可能なクライアントサーバー環境が促進されます。

> **Important**
>
> 初期化クエリも固定フィルターも、RDS Proxy がクライアントとサーバーのクエリトラフィックを変更する原因にはなりません。クライアントから到着したステートメントは、初期化クエリや固定フィルターの設定に関係なく、データベースに渡されます。

詳細については、以下を参照してください。

- Amazon RDS の「プロキシの作成」および「プロキシの変更」。
- Amazon Aurora の「プロキシの作成」と「プロキシの変更」。

PostgreSQL 接続の場合、クライアントドライバーとプロキシの間で交換される起動メッセージに、サポートされている接続パラメータを含めることもできます。これにより、個別の SET コマンドが送信されなくなり、ラウンドトリップが減少し、明示的な SET ステートメントによる接続の固定が回避されます。詳細については、「PostgreSQL への接続に関する考慮事項」を参照してください。

アプリケーション、プロキシ、データベース設定の調整

前のセクションで説明したように、RDS Proxy はプロキシの動作をアプリケーションのニーズに合わせるのに役立つさまざまなパラメータをサポートしています。ただし、適切な設定値を選択することは、アプリケーション、プロキシ、データベース自体など、スタックのすべてのレイヤーにまたがるタスクです。これらのすべてのコンポーネントの設定は、以下の目標を念頭に置いて調整する必要があります。

1. 通常のオペレーション中に期待されるレベルのパフォーマンスとスケーラビリティを提供します。
2. ワークロードの問題発生時のトラブルシューティングの明確さと容易性を促進します。
3. アプリケーションへの影響を最小限に抑えながら、スタックが予期しないイベント (ワークロードの急増など) に対処できるようにします。

多層環境で設定を選択して調整する場合は、下位レイヤーの制限とタイムアウトが、上位レイヤーの制限とタイムアウトに対応する値以上になるように設定値を調整してください。つまり、1 つのレイヤーの設定を、スタックのさらに下にある次の設定エンベロープに収まるエンベロープとして扱います。

例えば、アプリケーションが最上位レイヤー、プロキシが中間レイヤー、データベースが下位レイヤーであるとします。プロキシレベルのタイムアウトと制限がアプリケーションレベルの制限より低い場合、プロキシ制限はアプリケーション制限よりも優先されます。アプリケーションは設定を実行できず、独自の設定では説明できない動作が発生します。

例として `IdleClientTimeout` プロキシ設定を考えてみましょう。アプリケーションドライバーまたはクライアントプールが独自のアイドルタイムアウトを適用する場合、プロキシのアイドルタイムアウトはアプリケーション設定上のセーフティネットとして機能します。つまり、混乱を防ぐために、`IdleClientTimeout` はアプリケーションレベルのアイドルタイムアウトと少なくとも等しくなければなりません。

- アプリケーションのアイドルタイムアウトがプロキシタイムアウトよりも短い場合、アプリケーションは設定どおりに接続を閉じることが予想されます。アプリケーションがアイドル状態の接続をタイムリーに閉じられない場合、プロキシはバックストップとして機能します。

- アプリケーションのアイドルタイムアウトがプロキシタイムアウトよりも長い場合、アプリケーションは途中で接続が閉じられる可能性があります。これにより、アプリケーション側で混乱が生じる可能性があります。

接続制限などの他の設定にも同じロジックが適用されます。各レイヤーの設定は、次のレイヤーの設定で定義されたエンベロープ内に収まる必要があります。

最良の結果を得るには、設定において、1 つのレイヤーと次のレイヤーの間にパディングを含める必要があります。例えば、クライアント/サーバーのクロックドリフトによる散発的なエラーを回避するために、またはクライアントが接続を適切に閉じるために追加の時間が必要な場合に備えて、プロキシのタイムアウトをアプリケーションのタイムアウトよりも数秒長くすることができます。

つまり、次のように設定を調整します。

```
client timeout < proxy timeout < database timeout
```

次の例は推奨されません:

```
client timeout = proxy timeout = database timeout
```

また、以下は避けてください。

```
client timeout > proxy timeout > database timeout
```

データベース設定

接続制限

RDS Proxy は MaxConnectionsPercent 設定を使用して接続プールの最大サイズを決定します。つまり、プロキシの接続プールのサイズはデータベースの接続制限に相対的です。データベースの接続制限を変更すると、プロキシのプールサイズも自動的に変更されます。データベースで非プロキシユーザーの接続制限の一部を予約する場合は、データベース制限を増やすのではなく、プロキシの MaxConnectionsPercent 設定を下げる必要があります。

RDS Proxy を使用しても、データベース接続制限を適切に設定する必要性がなくなるわけではありません。単一のプロキシ接続は、本質的に単一の直接クライアント接続よりも軽量ではないため、プロキシを使用しているという理由だけでデータベース制限を増やさないでください。プロキシは、データベースがクエリを処理するために実行する作業量を削減するわけではありませんが、データベースがより少ない接続で同じワークロードを処理するのに役立ちます。

アイドルタイムアウト

データベースは、MySQL の `wait_timeout` および `interactive_timeout` 設定、PostgreSQL の `transaction_timeout` および `idle_in_transaction_session_timeout` 設定などを使用して、独自のアイドルタイムアウトを適用できます。これらの設定のデフォルト値がプロキシ設定に干渉する可能性は低いですが、カスタムのデータベースレベルのタイムアウトを使用する場合は、対応するプロキシタイムアウトと同じ長さ以上であることを確認してください。そうでない場合、タイムアウトが早すぎるためにプロキシで接続エラーが発生します。

接続キラーを使用するデータベース環境にも同じロジックが適用されます。接続キラーは、セッション状態をモニタリングし、特定の基準に基づいて接続をアクティブに終了させるスクリプトまたはプロセスです。環境がこのような手法を使用している場合は、接続終了ロジックがプロキシ設定と一致していることを確認してください。

プロキシを介してすべてのワークロードを処理するデータベースは、通常、アイドルタイムアウトのプロキシ設定に依存し、データベースレベルの設定をデフォルト値のままにすることができます。

アプリケーションの構成

セッション状態の管理

多くのデータベースドライバー、アプリケーションフレームワーク、オブジェクトリレーショナルマッピング (ORM) ツールでは、セッション変数または `SET` ステートメントを使用して、クエリトラフィックを送信する前に接続を設定します。接続およびトランザクションの初期化ステートメントの使用は、アプリケーションコードのみを見ると明確ではない場合があり、データベース自体と SQL ステートメントおよびアプリケーションロジックの間に複数の抽象化レイヤーが存在する可能性があります。RDS Proxy の拡張ログ記録機能を使用して、接続固定の理由を記録できます。また、データベースクエリログは、データベース接続を介して送信されるすべてのステートメントに関する詳細情報を提供できます。

以下のベストプラクティスを考慮します。

1. 接続パラメータは、データベースのデフォルトと異なる場合、クライアントセッションがそれらのデフォルトから逸脱する必要がある場合にのみ設定します。不要な初期化ステートメントを削除することは、多重化に役立つだけでなく、新しいデータベース接続ごとにクライアントとサーバーのラウンドトリップの数を減らすことにもつながります。
2. すべての接続間で可変と構成設定を一貫して設定します。

3. クエリ実行時に適用できるセッション設定は避けてください。例えば、異なるクライアントが異なるタイムゾーンのデータを参照する必要がある場合は、セッションレベルでタイムゾーンを設定するのではなく、クエリレベルでタイムゾーン変換関数を使用することを検討してください。
4. 可能であれば、初期化クエリ機能を使用してセッション設定ステートメントをプロキシレイヤーに移動します。詳細については、以下を参照してください。初期化クエリと固定フィルター

ライブネスチェック

アプリケーションが接続プーリングまたは高度なドライバーを使用している場合は、プロトコル ping、ヘルスチェックステートメント、キープアライブクエリなどのライブネスチェックに関連する設定を探します。RDS Proxy はすべてのクライアントリクエストを均等に扱うため、アプリケーションの観点からは `SELECT 1;` クエリまたは `COM_PING` リクエストが何もしない場合も、プロキシがアイドル状態のクライアントタイムアウトを適用や、`MaxIdleConnectionsPercent` に基づく接続プールサイズの管理を妨げることになります。

> **Note**
>
> RDS Proxy は、接続のアクティビティに関係なく、最大接続期間に 24 時間を適用します。

ほとんどの場合、クライアント側のライブネスチェックを無効にしてプロトコルノイズを減らし、RDS Proxy がアイドル状態の接続を管理できるようにすることをお勧めします。以下に関係なく、これらのヘルスチェックを実行するエッジケースがあります。

- 意図的にプロキシレイヤーで特定の接続がタイムアウトしないようにしたい。
- アプリケーションドライバーまたはプールが、プロキシによって接続が切断されたことをプロアクティブに検出できるようにしたい。例えば、固定されたバックエンド接続がデータベースの再起動によって閉じられたり、クライアント接続が 24 時間の最大存続期間を超えたりする可能性があります。

アプリケーション側でライブネスチェックを無効にするか、タイムアウトを防ぎたい特定の接続に対してのみ実行することを検討してください。

セッション状態の追跡

MariaDB ドライバーなどの一部の MySQL データベースドライバーは、セッション状態の追跡を有効にするために `session_track_*` 変数を使用します。この機能を有効にすると、クライアントが

サーバーが追跡できるセッション状態を変更するたびに、サーバーはレスポンスパケットに状態変更情報を含めます。これにより、ドライバーはサーバーからセッション状態に関する通知を受け取ることができます。

このセッション状態の追跡方法は、クライアントがサーバーと直接やり取りし、マルチサーバー環境でのセッション移行などの独自のセッション管理機能を実装する場合に有効です。RDS Proxy は独自の状態追跡メカニズムを実装し、`session_track_*` 変数によって有効化された情報は使用しませんが、これらの変数を設定するとセッションが固定されます。

データベースドライバーがこれらの変数を設定する場合は、ドライバーの追跡機能を無効にしたり、別のドライバーに切り替えたり、安全であればセッション固定フィルターを使用してステートメントを無視したりする方法を探すことができます。詳細については、以下を参照してください。初期化クエリと固定フィルター

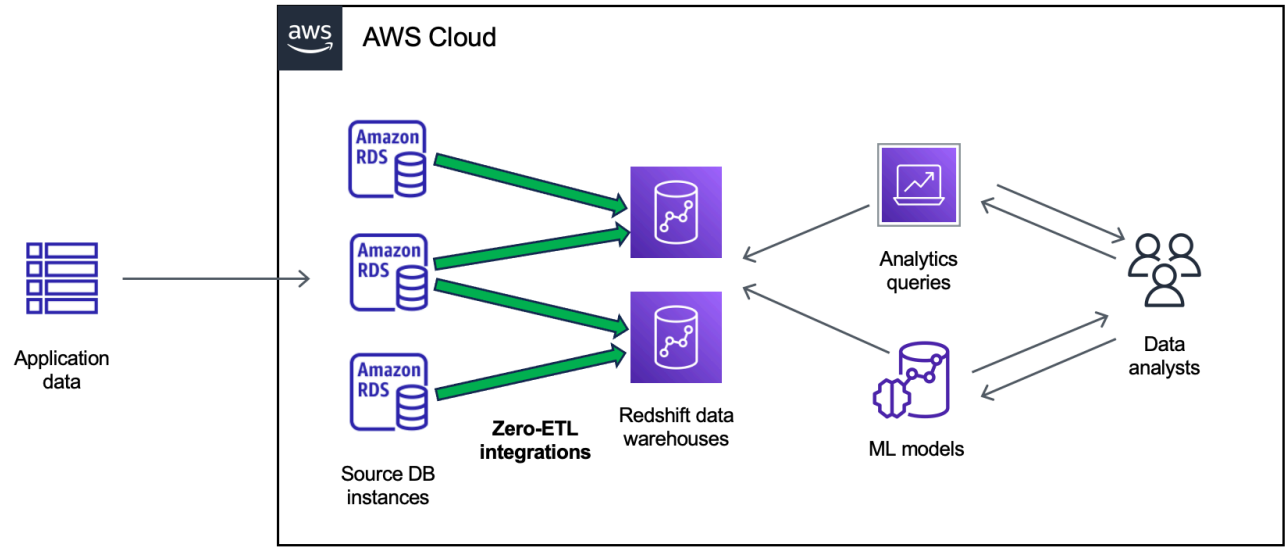
Amazon RDS ゼロ ETL 統合

Amazon Redshift および Amazon SageMaker AI との Amazon RDS ゼロ ETL 統合では、RDS からのデータを使用して、ほぼリアルタイムの分析と機械学習 (ML) が可能です。これは、トランザクションデータを RDS データベースに書き込んだ後に分析送信先で利用できるようにするためのフルマネージドソリューションです。抽出、変換、ロード (ETL) は、複数のソースからのデータを大規模な中央のデータウェアハウスにまとめるプロセスです。

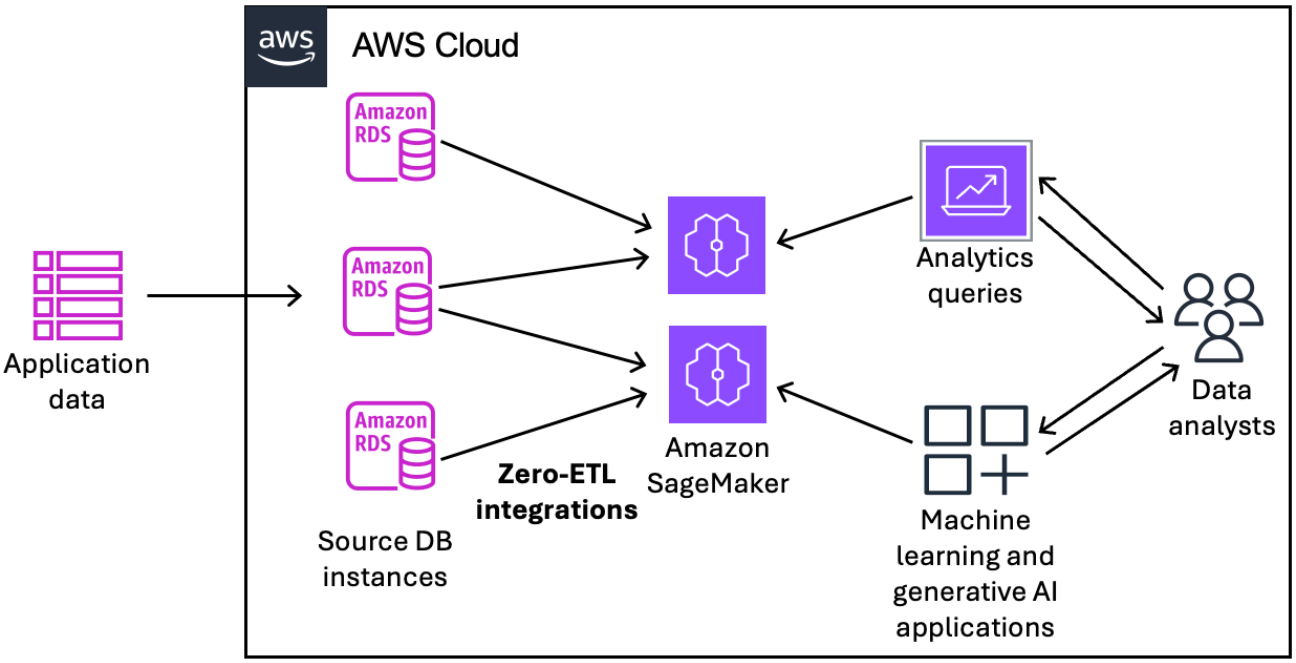
ゼロ ETL 統合では、RDS データベースのデータが Amazon Redshift または Amazon SageMaker AI Lakehouse でほぼリアルタイムで利用できます。データがターゲットデータウェアハウスまたはデータレイクに格納されると、機械学習、マテリアライズドビュー、データ共有、複数のデータストアやデータレイクへのフェデレーションアクセス、および Amazon SageMaker AI、Quick、その他の AWS のサービスとの統合といった組み込み機能を使用して、分析、ML、AI のワークロードを強化できます。

ゼロ ETL 統合を作成するには、RDS データベースをソースとして指定し、サポートされたデータウェアハウスやレイクハウスをターゲットとして指定します。統合では、ソースデータベースからターゲットデータウェアハウスやレイクハウスにデータがレプリケートされます。

次の図は、Amazon Redshift とのゼロ ETL 統合のこの機能を示しています。



次の図は、Amazon SageMaker AI Lakehouse とのゼロ ETL 統合のこの機能を示しています。



統合では、データパイプラインの状態をモニタリングし、可能な場合は問題から回復します。複数の RDS データベースから単一のターゲットデータウェアハウスまたはレイクハウスに統合を作成できるため、複数のアプリケーションにわたってインサイトを引き出すことができます。

トピック

- 利点
- 主要なコンセプト
- 制限事項
- クォータ
- サポート対象のリージョン
- Amazon RDS ゼロ ETL 統合の開始方法
- Amazon Redshift との Amazon RDS ゼロ ETL 統合の作成
- Amazon SageMaker Lakehouse との Amazon RDS ゼロ ETL 統合の作成
- Amazon RDS ゼロ ETL 統合でのデータフィルタリング
- ソース RDS データベースへのデータの追加とクエリ
- Amazon RDS ゼロ ETL 統合の表示と監視
- Amazon RDS ゼロ ETL 統合の変更
- AmazonRDS ゼロ ETL 統合の削除
- Amazon RDS ゼロ ETL 統合のトラブルシューティング

利点

RDS ゼロ ETL 統合には、主に次のような利点があります。

- 複数のデータソースから総合的なインサイトを引き出すのに役立ちます。
- 抽出、変換、ロード (ETL) 操作を実行する複雑なデータパイプラインを構築して管理する必要がなくなります。ゼロ ETL 統合は、パイプラインのプロビジョニングと管理を顧客に代わって行うことで、パイプラインの構築と管理に伴う課題を排除します。
- 運用上の負担とコストを削減し、アプリケーションの改善に集中できます。
- ターゲット送信先の分析機能と ML 機能を活用して、トランザクションデータやその他のデータからインサイトを引き出し、重要で時間的制約のあるイベントに効果的に対応できます。

主要なコンセプト

ゼロ ETL 統合を始める際には、以下の概念を検討してください。

統合

RDS データベースからデータウェアハウスまたはカタログにトランザクションデータとスキーマを自動的に複製する、フルマネージドデータパイプライン。

ソースデータベース

データがレプリケートされる RDS データベース。シングル AZ またはマルチ AZ DB インスタンス、あるいはマルチ AZ DB クラスター (RDS for MySQL のみ) を指定できます。

Target

データがレプリケートされるデータウェアハウスまたはレイクハウス。データウェアハウスには、プロビジョニングされたクラスターデータウェアハウスとサーバーレスデータウェアハウスの 2 種類があります。プロビジョニングされたクラスターデータウェアハウスは、ノードと呼ばれるコンピューティングリソースのコレクションであり、クラスターと呼ばれるグループに編成されています。サーバーレスデータウェアハウスは、コンピューティングリソースを格納するワークグループと、データベースオブジェクトとユーザーを収容する名前空間で構成されています。どちらのデータウェアハウスも分析エンジンを実行し、1 つ以上のデータベースを含んでいます。

ターゲットレイクハウスは、カタログ、データベース、テーブル、ビューで構成されます。レイクハウスアーキテクチャの詳細については、「Amazon SageMaker AI Unified Studio ユーザーガイド」の「SageMaker Lakehouse components」を参照してください。

複数のソースデータベースの を同じターゲットに書き込むことができます。

詳細については、「Amazon Redshift デベロッパーガイド」の「データウェアハウスのシステムアーキテクチャ」を参照してください。

制限事項

RDS ゼロ ETL 統合には、以下の制限が適用されます。

トピック

- 一般的な制限事項
- RDS for MySQL の制限事項
- RDS for PostgreSQL の制限事項
- RDS for Oracle の制限事項
- Amazon Redshift の制限事項
- Amazon SageMaker AI Lakehouse の制限事項

一般的な制限事項

- ソースデータベースは、ターゲットと同じリージョンにある必要があります。
- クラスターに既存の統合がある場合、データベースやそのインスタンスの名前を変更することはできません。
- 同じソースデータベースとターゲットデータベースの間に複数の統合を作成することはできません。
- 既存の統合があるデータベースは削除できません。まず、関連する統合をすべて削除する必要があります。
- ソースデータベースを停止すると、データベースを再開するまで、最後のいくつかのトランザクションがターゲットにレプリケートされない場合があります。
- ソースデータベースが停止している場合、統合を削除することはできません。
- データベースがブルー/グリーンデプロイのソースである場合、ブルー環境とグリーン環境の切り替え中に既存のゼロ ETL 統合を置くことはできません。最初に統合を削除してから切り替えて、再作成する必要があります。
- 統合を作成中のソースデータベースで別の統合を作成することはできません。

- 初めて統合を作成するとき、またはテーブルを再同期するとき、ソースデータベースのサイズによっては、ソースからターゲットへのデータシードに 20 ~ 25 分以上かかる場合があります。この遅延により、レプリカラグが長くなる可能性があります。
- 一部のデータ型はサポートされていません。詳細については、「the section called “データタイプの相違点”」を参照してください。
- システムテーブル、一時テーブル、ビューはターゲットウェアハウスにレプリケートされません。
- ソーステーブルで DDL コマンド (ALTER TABLE など) を実行すると、テーブルの再同期がトリガーされ、再同期中はテーブルをクエリできなくなる場合があります。詳細については、「the section called “1 つ以上の Amazon Redshift テーブルを再同期する必要がある”」を参照してください。

RDS for MySQL の制限事項

- ソースデータベースは、サポートされているバージョンの RDS for MySQL を実行している必要があります。サポートされているバージョンのリストについては、「」を参照してくださいthe section called “ゼロ ETL 統合 ”
- ゼロ ETL 統合は、同じ AWS リージョンのプライマリインスタンスとリードレプリカインスタンスの両方ではサポートされません。
- ゼロ ETL 統合では、MySQL バイナリロギング (binlog) を利用して継続的なデータ変更をキャプチャします。バイナリログベースのデータフィルタリングは使用しないでください。ソースとターゲットのデータベース間でデータの不整合が生じる可能性があります。
- ゼロ ETL 統合は、InnoDB ストレージエンジンを使用するように設定されたデータベースでのみサポートされています。
- 定義済みのテーブル更新を伴う外部キー参照はサポートされていません。具体的には、ON DELETE および ON UPDATE ルールは、CASCADE、SET NULL、および SET DEFAULT アクションではサポートされていません。別のテーブルへの参照を含むテーブルを作成または更新しようとすると、テーブルは失敗状態になります。
- マグネティックストレージを使用するソースデータベースの統合を作成することはできません。

RDS for PostgreSQL の制限事項

- ソースデータベースは、バージョン 15.7 以降、16.3 以降、または 17.1 以降を実行している RDS for PostgreSQL インスタンスである必要があります。以前のバージョンはサポートされていません。

- RDS for PostgreSQL ゼロ ETL 統合は、ソースデータベースとしてマルチ AZ DB クラスターをサポートしていません。
- RDS for PostgreSQL リードレプリカインスタンスからゼロ ETL 統合を作成することはできません。
- PostgreSQL のログに記録されていないテーブルとマテリアライズドビューは Amazon Redshift にレプリケートされません。
- Amazon Redshift の制限により、ジオメトリデータタイプや 64KB を超えるデータなど、特定の PostgreSQL データタイプのレプリケーションはサポートされていません。RDS for PostgreSQL と Amazon Redshift のデータタイプの違いの詳細については、「データタイプの相違点」セクションの「the section called “RDS for PostgreSQL”」を参照してください。
- アクティブなゼロ ETL 統合があるソース RDS for PostgreSQL インスタンスでは、メジャーバージョンアップグレードを実行できません。ソースインスタンスをアップグレードするには、まず既存のゼロ ETL 統合をすべて削除する必要があります。メジャーバージョンアップグレードが完了したら、ゼロ ETL 統合を再作成できます。
- ソース DB インスタンスで宣言的パーティショニングトランザクションを実行すると、影響を受けるすべてのテーブルが失敗状態になり、アクセスできなくなります。
- ソース RDS for PostgreSQL インスタンスで `max_slot_wal_keep_size` が有限値に設定されている場合、WAL の保持量がそのサイズを超えると、統合に使用されている論理レプリケーションスロットが無効になる可能性があります。無効化されたスロットはレプリケーションを停止させ、再作成しない限り統合を復旧させることはできません。`max_slot_wal_keep_size` は、PostgreSQL のデフォルト値である `-1` (無制限) のままにするか、統合ブートストラップや一時的な遅延中に生成される WAL に対応するのに十分な大きさに設定することをお勧めします。

RDS for Oracle の制限事項

- ソースデータベースは、バージョン 19c の Enterprise Edition または Standard Edition 2 (2019 年 7 月リリースアップデート以降) を実行している RDS for Oracle インスタンスである必要があります。以前のバージョンはサポートされていません。
- RDS for Oracle リードレプリカインスタンスからゼロ ETL 統合を作成することはできません。
- テナントデータベースにゼロ ETL 統合がある場合、このテナントデータベースの名前を変更することはできません。
- テナントデータベースは 1 つのゼロ ETL 統合のみを持つことができます。

- RDS for Oracle と Amazon Redshift のデータタイプ間には、いくつかの違いがあります。詳細については、「データタイプの相違点」セクションの「the section called “RDS for Oracle”」を参照してください。

Amazon Redshift の制限事項

ゼロ ETL 統合に関連する Amazon Redshift の制限の一覧については、「Amazon Redshift 管理ガイド」の「Amazon Redshift とのゼロ ETL 統合を使用する際の考慮事項」を参照してください。

Amazon SageMaker AI Lakehouse の制限事項

以下は、Amazon SageMaker AI Lakehouse のゼロ ETL 統合の制限です。

- カタログ名の長さは 19 文字に制限されています。

クォータ

お客様のアカウントには、RDS ゼロ ETL 統合に関連する以下のクォータが設定されています。特に指定がない限り、各クォータはリージョンあたりです。

名前	デフォルト	説明
統合	100	AWS アカウント 内の統合の総数。
ターゲットあたりの統合	50	1 つのターゲットデータウェアハウスまたはレイクハウスにデータを送信する統合の数。
ソースインスタンスごとの統合	5	単一のソース DB インスタンスからデータを送信する統合の数。

さらに、ターゲットウェアハウスは、各 DB インスタンスまたはクラスターノードで使用できるテーブルの数に一定の制限を設けています。Amazon Redshift のクォータと制限の詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

サポート対象のリージョン

RDS ゼロ ETL 統合は、AWS リージョンのサブセットで利用できます。サポートされているリージョンのリストについては「the section called “ゼロ ETL 統合 ”」を参照してください。

Amazon RDS ゼロ ETL 統合の開始方法

ゼロ ETL 統合を作成する前に、必要なパラメータとアクセス許可で RDS データベースとデータウェアハウスを設定します。セットアップ時には、以下の手順を完了します。

1. カスタム DBのパラメータグループを作成します。
2. ソースデータベースを作成します。
3. Amazon Redshift のターゲットのデータウェアハウスを作成するか、ターゲット Amazon SageMaker Lakehouse を作成します。

これらのタスクが完了したら、「the section called “Amazon Redshift とのゼロ ETL 統合の作成”」または「the section called “Amazon SageMaker Lakehouse とのゼロ ETL 統合の作成”」に進みます。

> **Tip**
>
> RDS では、統合の作成中にこれらのセットアップ手順を手動で行わずに、自動的に完了させることができます。統合の作成をすぐに開始するには、「the section called “Amazon Redshift とのゼロ ETL 統合の作成”」を参照してください。

ステップ 3 では、必要に応じて、ターゲットデータウェアハウス (ステップ 3a) またはターゲットレイクハウス (ステップ 3b) のいずれかを作成できます。

- SQL ベースの分析で従来のデータウェアハウス機能が必要な場合は、データウェアハウスを選択します。
- 機械学習機能が必要であるとともに、データサイエンスと ML ワークフローに Lakehouse 機能を使用する場合は、Amazon SageMaker Lakehouse を選択します。

ステップ 1: カスタム DB のパラメータグループを作成する

Amazon RDS ゼロ ETL 統合では、データのレプリケーションを制御する特定の値が DB パラメータに必要です。特定のパラメータは、ソース DB エンジンによって異なります。これらのパラメータを

設定するには、まずカスタム DB パラメータグループを作成して、これをソースデータベースに関連付ける必要があります。ソース DB エンジンに応じて、以下のパラメータ値を設定します。カスタムパラメータグループを作成するには、「the section called “DB パラメータグループ”」を参照してください。依存関係の問題を避けるため、すべてのパラメータ値を同じリクエスト内で設定することをお勧めします。

RDS for MySQL:

- `binlog_format=ROW`
- `binlog_row_image=full`

また、`binlog_row_value_options` パラメータが `PARTIAL_JSON` に設定されていないことも確認してください。ソースデータベースがマルチ AZ DB クラスターの場合は、`binlog_transaction_compression` パラメータが `ON` に設定されていないことを確認します。

これらのパラメータの一部 (`binlog_format` など) は動的です。つまり、再起動をトリガーせずにパラメータに変更を適用できます。これは、一部の既存のセッションがパラメータの古い値を使用し続ける可能性があることを意味します。これに伴ってゼロ ETL 統合の作成時に問題が起きないようにするには、パフォーマンススキーマを有効にします。パフォーマンススキーマにより、ゼロ ETL 事前チェックが実行されるため、プロセスの早い段階で欠落しているパラメータを検出できます。

RDS for PostgreSQL:

- `rds.logical_replication = 1`
- `rds.replica_identity_full = 1`
- `session_replication_role = origin`
- `wal_sender_timeout ≥ 20000 or = 0`
- `max_wal_senders ≥ 20`
- `max_replication_slots ≥ 20`
- `max_slot_wal_keep_size = -1` (推奨)、または統合がアクティブに消費していない間に発生する WAL 生成のピーク量をカバーするのに十分なサイズ。

複数の PostgreSQL 統合の場合、統合ごとに 1 つの論理レプリケーションスロットが使用されます。使用状況に基づいて `max_replication_slots` パラメータと `max_wal_senders` パラメータを確認します。

ゼロ ETL 統合で効率的なデータ同期を行うには、ソース DB インスタンスで `rds.replica_identity_full` を設定します。これにより、UPDATE および DELETE オペレーション中に、先行書き込みログ (WAL) にプライマリキー情報だけでなく、行全体のデータを記録するようデータベースに指示します。ゼロ ETL では、レプリケートされたすべてのテーブルに必要なプライマリキーがある場合でも、行全体のデータが必要です。クエリ時にどのデータを表示するかを決定するために、Amazon Redshift は特殊なアンチ結合戦略を使用して、データを内部の削除追跡テーブルと比較します。Amazon Redshift は、行全体のイメージをログに記録することで、これらのアンチ結合を効率的に実行できます。行全体のデータがない場合、Amazon Redshift は追加のルックアップを実行する必要があるため、Amazon Redshift が使用する列指向エンジンの高スループットオペレーション中にパフォーマンスが低下する可能性があります。

> **Important**
>
> 行全体をログに記録するようにレプリカ ID を設定すると、WAL ボリュームが増加します。これに伴って、特に幅の広いテーブルや頻繁な更新において、書き込み増幅と I/O 使用量が高くなる可能性があります。このような影響に備えるには、ストレージ容量と I/O 要件を計画し、WAL の増加をモニタリングして、書き込みが多いワークロードのレプリケーション遅延を追跡します。

> **Important**
>
> `max_slot_wal_keep_size` が有限値に設定され、統合が WAL をアクティブに消費していない間 (統合の作成中、変更中、一時的な遅延中など)、統合の論理レプリケーションスロットにそのサイズを超える WAL が蓄積された場合、PostgreSQL はそのスロットを無効にします。一度無効にすると、統合はソースデータベースからのレプリケーションを再開できません。PostgreSQL のデフォルト値 -1 (無制限) は、これを防ぎます。WAL 保持量に上限を設ける必要がある場合は、統合がアクティブに消費していない期間中のデータベースの書き込みスループットに対応するサイズを選択します。

RDS for Oracle:

RDS for Oracle ではパラメータの変更は不要です。

ステップ 2: ソースデータベースを選択または作成する

カスタム DB のパラメータグループを作成したら、RDS DB インスタンスを選択または作成します。このデータベースは、ターゲットデータウェアハウスへのデータレプリケーションのソースになります。シングル AZ DB インスタンスまたはマルチ AZ DB インスタンスの作成手順については、「the section called “DB インスタンスの作成”」を参照してください。マルチ AZ DB クラスター (RDS for MySQL のみ) を作成する手順については、「the section called “マルチ AZ DB クラスターの作成”」を参照してください。

データベースは、サポートされている DB エンジンバージョンを実行している必要があります。サポートされているバージョンのリストについては、「the section called “ゼロ ETL 統合 ”」を参照してください。

データベースを作成するには、[追加設定] で、デフォルトの DB パラメータグループを、前のステップで作成したカスタムパラメータグループに変更します。

> **Note**
>
> データベースの作成後にパラメータグループをデータベースに関連付ける場合は、ゼロ ETL 統合を作成する前にデータベースを再起動して変更を適用する必要があります。手順については、「the section called “DB インスタンスを再起動する”」または「the section called “マルチ AZ DB クラスターの再起動”」を参照してください。

さらに、データベースで自動バックアップが有効になっていることを確認してください。詳細については、「the section called “自動バックアップの有効化”」を参照してください。

ステップ 3a: ターゲットデータウェアハウスを作成する

ソースデータベースを作成した後、ターゲットのデータウェアハウスを作成して設定する必要があります。データウェアハウスは、以下の要件を満たしている必要があります。

- 少なくとも 2 つのノードがある RA3 ノードタイプまたは Redshift Serverless を使用している。
- 暗号化されている (プロビジョニングされたクラスターを使用している場合) 詳細については、「Amazon Redshift データベースの暗号化」を参照してください。

データウェアハウスを作成する手順については、プロビジョニングされたクラスター用の「クラスターの作成」またはRedshift Serverless 用の「名前空間を使用したワークグループの作成」を参照してください。

データウェアハウスで大文字と小文字の区別を有効にします。

統合を正常に行うには、データウェアハウスで大文字と小文字を区別するパラメータ (enable_case_sensitive_identifier) を有効にする必要があります。デフォルトでは、プロビジョニング済みクラスターと Redshift Serverless ワークグループの大文字と小文字の区別は無効になっています。

大文字と小文字の区別を有効にするには、データウェアハウスのタイプに応じて以下の手順を実行します。

- プロビジョニングされたクラスター — プロビジョニングされたクラスターで大文字と小文字の区別を有効にするには、enable_case_sensitive_identifier パラメータを有効にしたカスタムパラメータグループを作成します。次に、そのパラメータグループとクラスターを関連付けます。手順については、「コンソールを使用したパラメータグループの管理」または「AWS CLI を使用したパラメータ値の設定」を参照してください。

  > **Note**
  >
  > クラスターにパラメータグループを関連付けたら、クラスターを再起動することを忘れないでください。

- Serverless ワークグループ — Redshift Serverless ワークグループで大文字と小文字の区別を有効にするには、AWS CLI を使用する必要があります。Amazon Redshift コンソールは現在、Redshift Serverless パラメータ値の変更をサポートしていません。次の update-workgroup リクエストを送信します。

  ```
  aws redshift-serverless update-workgroup \
    --workgroup-name target-workgroup \
    --config-parameters
   parameterKey=enable_case_sensitive_identifier,parameterValue=true
  ```

  パラメータ値を変更した後、ワークグループを再起動する必要はありません。

データウェアハウスの認可を設定します。

データウェアハウスを作成したら、ソース RDS データベースを承認済みの統合ソースとして設定する必要があります。手順については、「Amazon Redshift データウェアハウスの認可を設定する」を参照してください。

AWS を使用して統合をセットアップする

各リソースを手動でセットアップするのではなく、次の Python スクリプトを実行して、必要なリソースを自動的にセットアップできます。このコード例では AWS SDK for Python (Boto3) を使用してソース RDS for MySQL DB インスタンスとターゲットデータウェアハウスを作成し、それぞれに必要なパラメータ値を指定します。次に、データベースが使用可能になるまで待ってから、データベース間にゼロ ETL 統合を作成します。設定する必要があるリソースに応じて、さまざまな関数をコメントアウトできます。

必要な従属関係をインストールには、次のコマンドを実行します。

```
pip install boto3
pip install time
```

スクリプト内で、オプションでソース、ターゲット、パラメータグループの名前を変更します。最後の関数は、リソースのセットアップ後に `my-integration` という名前の統合を作成します。

Python コード例

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_db_name = 'my-source-db' # A name for the source database
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_db(*args):
    """Creates a source RDS for MySQL DB instance"""

    response = rds.create_db_parameter_group(
        DBParameterGroupName=source_param_group_name,
```

```
        DBParameterGroupFamily='mysql8.0',
        Description='RDS for MySQL zero-ETL integrations'
    )
    print('Created source parameter group: ' + response['DBParameterGroup']
['DBParameterGroupName'])

    response = rds.modify_db_parameter_group(
        DBParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' + response['DBParameterGroupName'])

    response = rds.create_db_instance(
        DBInstanceIdentifier=source_db_name,
        DBParameterGroupName=source_param_group_name,
        Engine='mysql',
        EngineVersion='8.0.32',
        DBName='mydb',
        DBInstanceClass='db.m5.large',
        AllocatedStorage=15,
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source database: ' + response['DBInstance']
['DBInstanceIdentifier'])
    source_arn = (response['DBInstance']['DBInstanceArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
```

```
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='RDS for MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy granting access to source database and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
```

```
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_db_availability(*args):
    """Waits for both databases to be available"""

    print('Waiting for source and target to be available...')

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_db_name
    )
    source_status = response['DBInstances'][0]['DBInstanceStatus']
    source_arn = response['DBInstances'][0]['DBInstanceArn']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the databases are available
    if source_status != 'available' or target_status != 'available':
        time.sleep(60)
        response = wait_for_db_availability(
            source_db_name, target_cluster_name)
```

```
    else:
        print('Databases available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target databases"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_db(source_db_name, source_param_group_name)
    wait_for_db_availability(source_db_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

ステップ 3b: Amazon SageMaker Lakehouse ゼロ ETL 統合用の AWS Glue カタログを作成する

Amazon SageMaker Lakehouse とのゼロ ETL 統合を作成する場合は、AWS Lake Formation で AWS Glue マネージドカタログを作成する必要があります。ターゲットカタログは Amazon Redshift マネージドカタログである必要があります。Amazon Redshift マネージドカタログを作成するには、まず `AWSServiceRoleForRedshift` サービスにリンクされたロールを作成します。Lake Formation コンソールで、読み取り専用管理者として `AWSServiceRoleForRedshift` を追加します。

その前のタスクの詳細については、次のトピックを参照してください。

- Amazon Redshift マネージドカタログの作成の詳細については、「AWS Lake Formation デベロッパーガイド」の「Creating an Amazon Redshift managed catalog in the AWS Glue Data Catalog」を参照してください。
- Amazon Redshift でのサービスにリンクされたロールの詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift でのサービスにリンクされたロールの使用」を参照してください。

- Lake Formation の読み取り専用での管理者アクセス許可の詳細については、「AWS Lake Formation デベロッパーガイド」の「Lake Formation personas and IAM permissions reference」を参照してください。

ターゲット AWS Glue カタログのアクセス許可を設定する

ゼロ ETL 統合のターゲットカタログを作成する前に、Lake Formation ターゲット作成ロールと AWS Glue データ転送ロールを作成する必要があります。Lake Formation ターゲット作成ロールを使用して、ターゲットカタログを作成します。ターゲットカタログを作成するときは、[エンジンからのアクセス] セクションの [IAM ロール] フィールドに Glue データ転送ロールを入力します。

Lake Formation ターゲット作成ロール

ターゲット作成ロールは Lake Formation 管理者であり、次のアクセス許可が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lakeformation:RegisterResource",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutEncryptionConfiguration",
                "iam:PassRole",
                "glue:CreateCatalog",
                "glue:GetCatalog",
                "s3:PutBucketTagging",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "redshift-serverless:CreateNamespace",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
```

```
                "redshift-serverless:CreateWorkgroup"
            ],
            "Resource": [
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:catalog/*",
                "arn:aws:s3:::*",
                "arn:aws:redshift-serverless:*:111122223333:workgroup/*",
                "arn:aws:redshift-serverless:*:111122223333:namespace/*",
                "arn:aws:iam::111122223333:role/GlueDataCatalogDataTransferRole"
            ]
        }
    ]
}
```

ターゲット作成ロールには、以下の信頼関係が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:user/Username"
          },
          "Action": "sts:AssumeRole"
        }
    ]
}
```

Glue データ転送ロール

Glue データ転送ロールは MySQL カタログオペレーションに必要であり、次のアクセス許可が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferRolePolicy",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "glue:GetCatalog",
                "glue:GetDatabase"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Glue データ転送ロールには、以下の信頼関係が必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "glue.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

次のステップ

ソースの RDS データベースと、Amazon Redshift ターゲットデータウェアハウスまたは Amazon SageMaker Lakehouse により、ゼロ ETL 統合を作成してデータをレプリケートできます。手順については、「the section called “Amazon Redshift とのゼロ ETL 統合の作成”」を参照してください。

Amazon Redshift との Amazon RDS ゼロ ETL 統合の作成

Amazon RDS ゼロ ETL 統合を作成するには、ソース RDS データベースとターゲットの Amazon Redshift データウェアハウスを指定します。暗号化設定をカスタマイズし、タグを追加することもできます。Amazon RDS はソースデータベースとそのターゲットの間の統合を作成します。統合がアクティブになると、ソースデータベースに挿入したデータはすべて、設定された Amazon Redshift ターゲットにレプリケートされます。

前提条件

ゼロ ETL 統合を作成する前に、ソースのデータベースとターゲットの Amazon Redshift データウェアハウスを作成する必要があります。また、データベースを承認済みの統合ソースとして追加することによって、データウェアハウスへのレプリケーションを許可する必要があります。

これらの各手順の実行方法については、「the section called “ゼロ ETL 統合の開始方法”」を参照してください。

必要なアクセス許可

ゼロ ETL 統合を作成するには、特定の IAM アクセス権限が必要です。少なくとも、次のアクションを実行するためのアクセス権限が必要です。

- ソースの RDS データベースのゼロ ETL 統合を作成します。
- すべてのゼロ ETL 統合を表示および削除します。
- ターゲットデータウェアハウスへのインバウンド統合を作成します。

以下のサンプルポリシーは、統合の作成と管理に必要な最小特権を示しています。ユーザーまたはロールが AdministratorAccess マネージドポリシーなど、より広範なアクセス許可を持つ場合、これらの正確なアクセス許可を必要としない場合があります。

> **Note**
>
> Redshift Amazon リソースネーム (ARN) の形式は次のとおりです。サーバーレス名前空間 UUID の前にコロン (:) ではなくフォワードスラッシュ (/) を使用していることに注意してください。
>
> - プロビジョニング済みクラスタ — arn:aws:**redshift**:{region}:{account-id}:namespace:*namespace-uuid*
> - サーバーレス - arn:aws:**redshift-serverless**:{region}:{account-id}:namespace/*namespace-uuid*

Redshift ターゲットのサンプルポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateIntegration",
      "Effect": "Allow",
      "Action": [
        "rds:CreateIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:db:source-db",
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "DescribeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeIntegrations"
```

```
      ],
      "Resource": [
      "arn:aws:rds:us-east-1:123456789012:integration:*"
  ]
    },
    {
      "Sid": "ChangeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DeleteIntegration",
        "rds:ModifyIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "AllowRedShiftIntegration",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:123456789012:namespace:namespace-uuid"
      ]
    }
  ]
}
```

別のアカウントでターゲットデータウェアハウスを選択する

別の AWS アカウント にあるターゲット Amazon Redshift データウェアハウスを指定する場合は、現在のアカウントのユーザーがターゲットアカウントのリソースにアクセスするのを許可するロールを作成する必要があります。詳細については、「所有している別の AWS アカウント の IAM ユーザーにアクセス権を付与する」を参照してください。

ロールには以下のアクセス許可が必要です。これにより、ユーザーは使用可能な Amazon Redshift のプロビジョニング済みクラスターとターゲットアカウントの Redshift Serverless 名前空間を表示できます。

必要なアクセス許可ポリシー

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

ロールには、ターゲットアカウント ID を指定する次の信頼ポリシーが必要です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成」を参照してください。

ゼロ ETL 統合の作成

AWS マネジメントコンソール、AWS CLI、または RDS API を使用してゼロ ETL 統合を作成できます。

> **Important**
>
> ゼロ ETL 統合は、更新または再同期オペレーションをサポートしていません。作成後に統合で問題が発生した場合は、統合を削除して新しい統合を作成する必要があります。

デフォルトで、RDS for MySQL はバイナリログファイルをすぐに消去します。ゼロ ETL 統合では、ソースからターゲットへのデータレプリケーションをバイナリログに依存しているため、ソースデータベースインスタンスの保持期間は 1 時間以上である必要があります。統合を作成したら、直ちに Amazon RDS は選択したソースデータベースのバイナリログファイルの保持期間を確認します。現在の値が 0 時間の場合、Amazon RDS は自動的に 1 時間に変更します。それ以外の場合、値は変わりません。

RDS コンソール

ゼロ ETL 統合を作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 左側のナビゲーションペインから、[ゼロ ETL 統合] を選択します。
3. [ゼロ ETL 統合の作成] を選択します。
4. [統合 ID] に、統合の名前を入力します。名前には最大 63 文字の英数字を使用でき、ハイフンを含めることができます。

   > **Important**
   >
   > カタログ名の長さは 19 文字に制限されています。カタログ名として使用する場合は、統合識別子がこの要件を満たしていることを確認します。

5. [次へ] を選択します。
6. [ソース] で、データの送信元となる RDS データベースを選択します。

> **Note**
>
> DBパラメータが正しく設定されていないと、RDS から通知されます。このメッセージを受け取った場合は、[Fix it for me] を選択するか、手動で設定することができます。手動で修正する手順については、「the section called “ステップ 1: カスタム DB のパラメータグループを作成する”」を参照してください。
> DB パラメータを変更するには再起動が必要です。統合を作成する前に、再起動が完了し、新しいパラメータ値のデータベースのへの適用が正常に完了している必要があります。

7. ソースデータベースが正常に設定されたら、[次へ] を選択します。
8. [ターゲット] で、以下を実行します。

   1. (オプション) Amazon Redshift ターゲットとして別の AWS アカウント を使用するには、[別のアカウントを指定] を選択します。次に、データウェアハウスを表示するアクセス許可を持つ IAM ロールの ARN を入力します。IAM ロールの作成手順については、「the section called “別のアカウントでターゲットデータウェアハウスを選択する”」を参照してください。
   2. [Amazon Redshift データウェアハウス] で、ソースデータベースからのレプリケートデータのターゲットを選択します。ターゲットとして、プロビジョニングされた Amazon Redshift クラスターまたは Redshift Serverless 名前空間を選択できます。

> **Note**
>
> 指定したデータウェアハウスのリソースポリシーまたは大文字と小文字の区別の設定が正しく構成されていないと、RDS から通知されます。このメッセージを受け取った場合は、[Fix it for me] を選択するか、手動で設定することができます。手動で修正する手順については、「Amazon Redshift 管理ガイド」の「データウェアハウスの大文字と小文字の区別を有効にする」と「データウェアハウスの認可を設定する」を参照してください。
> プロビジョニングされた Redshift クラスターの大文字と小文字の区別を変更するには、再起動が必要です。インテグレーションを作成する前に、再起動を完了し、新しいパラメータ値をクラスターに正常に適用する必要があります。
> 選択したソースとターゲットの AWS アカウント が異なる場合、Amazon RDS はこれらの設定を自動的に修正できません。他のアカウントに移動し、Amazon Redshift で手動で修正する必要があります。

9. ターゲットデータウェアハウスを正しく設定したら、[次へ] を選択します。
10. (オプション) [タグ] で、1 つ以上のタグを統合に追加します。詳細については、「the section called “RDS リソースのタグ付け”」を参照してください。
11. [暗号化] として、統合の暗号化方法を指定します。デフォルトでは、RDS はすべての統合を AWS 所有のキー で暗号化します。代わりにカスタマーマネージドキーを選択するには、[暗号化設定のカスタマイズ] を有効にして、暗号化に使用する KMS キーを選択します。詳細については、「the section called “Amazon RDS リソースの暗号化”」を参照してください。

    オプションで、暗号化コンテキストを追加します。詳しくは、AWS Key Management Service デベロッパーガイドの Encryption context を参照してください。

    > **Note**
    >
    > Amazon RDS は、追加する以下に加えて、次の暗号化コンテキストペアを追加します。
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > これにより、追加できるペアの総数が 8 から 6 に減り、許可の制約の全体的な文字数制限に影響します。詳細については、「AWS Key Management Service デベロッパーガイド」の「Using grant constraints」を参照してください。

12. [次へ] を選択します。
13. 統合設定を確認し、[ゼロ ETL 統合を作成] を選択します。

    作成に失敗した場合は、トラブルシューティングの手順について「the section called “ゼロ ETL 統合を作成できない”」を参照してください。

統合のステータスは、作成中は `Creating` であり、ターゲットの Amazon Redshift データウェアハウスのステータスは `Modifying` です。この間、データウェアハウスをクエリしたり、設定を変更したりすることはできません。

統合が正常に作成されると、統合とターゲットの Amazon Redshift データウェアハウスの両方のステータスが `Active` に変わります。

AWS CLI

AWS CLI を使用してゼロ ETL 統合を作成するには、create-integration コマンドに以下のオプションを指定して使用します。

> **Note**
>
> カタログ名は 19 文字に制限されていることにご注意ください。カタログ名として使用する場合は、それに応じて統合名を選択します。

- `--integration-name` — 統合の名前を指定します。
- `--source-arn` — 統合のソースとなる RDS データベースの ARN を指定します。
- `--target-arn` — 統合のターゲットとなる Amazon Redshift データウェアハウスの ARN を指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Windows の場合:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

RDS API

Amazon RDS API を使用してゼロ ETL 統合を作成するには、以下のパラメータを指定して CreateIntegration オペレーションを使用します。

> **Note**
>
> カタログ名は 19 文字に制限されています。カタログ名として使用する場合は、IntegrationName パラメータがこの要件を満たしていることを確認します。

- `IntegrationName` — 統合の名前を指定します。
- `SourceArn` — 統合のソースとなる RDS データベースの ARN を指定します。
- `TargetArn` — 統合のターゲットとなる Amazon Redshift データウェアハウスの ARN を指定します。

カスタマーマネージドキーでの統合の暗号化

統合の作成時に AWS 所有のキー ではなくカスタム KMS キーを指定する場合、キーポリシーは Amazon Redshift サービスプリンシパルに `CreateGrant` アクションへのアクセスを提供する必要があります。さらに、現在のユーザーに `DescribeKey` アクションと `CreateGrant` アクションの実行を許可する必要があります。

次のポリシー例は、キーポリシーに必要なアクセス許可を指定する方法を示しています。アクセス許可の範囲をさらに狭めるためのコンテキストキーが含まれています。

キーポリシーの例

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
```

```
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
```

```
            "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「キーポリシーの作成」を参照してください。

次のステップ

ゼロ ETL 統合を正常に作成した後、ターゲット Amazon Redshift クラスターまたはワークグループ内にデスティネーションデータベースを作成する必要があります。これで、ソースの RDS データベースにデータを追加し、Amazon Redshift でクエリを実行できるようになります。手順については、「Amazon Redshift でのデスティネーションデータベースの作成」を参照してください。

Amazon SageMaker Lakehouse との Amazon RDS ゼロ ETL 統合の作成

Amazon SageMaker Lakehouse と Amazon RDS ゼロ ETL 統合を作成するときは、ソース RDS データベースとターゲット AWS Glue マネージドカタログを指定します。暗号化設定をカスタマイズし、タグを追加することもできます。Amazon RDS はソースデータベースとそのターゲットの間の統合を作成します。統合がアクティブになると、ソースデータベースに挿入したデータはすべて、設定されたターゲットにレプリケートされます。

前提条件

Amazon SageMaker Lakehouse とゼロ ETL 統合を作成する前に、ソースデータベースとターゲット AWS Glue マネージドカタログを作成する必要があります。また、データベースを承認済みの統合ソースとして追加することによって、カタログへのレプリケーションを許可する必要があります。

これらの各手順の実行方法については、「the section called “ゼロ ETL 統合の開始方法”」を参照してください。

必要なアクセス許可

Amazon SageMaker Lakehouse とのゼロ ETL 統合を作成するには、特定の IAM アクセス許可が必要です。少なくとも、次のアクションを実行するためのアクセス権限が必要です。

- ソースの RDS データベースのゼロ ETL 統合を作成します。
- すべてのゼロ ETL 統合を表示および削除します。
- ターゲット AWS Glue マネージドカタログへのインバウンド統合を作成します。
- AWS Glue マネージドカタログで使用される Amazon S3 バケットにアクセスします。
- カスタム暗号化が設定されている場合は、暗号化に AWS KMS キーを使用します。
- Lake Formation にリソースを登録します。
- AWS Glue マネージドカタログにリソースポリシーを配置して、インバウンド統合を承認します。

以下のサンプルポリシーは、Amazon SageMaker Lakehouse との統合の作成と管理に必要な最小特権を示しています。ユーザーまたはロールが AdministratorAccess マネージドポリシーなど、より広範なアクセス許可を持つ場合、これらの正確なアクセス許可を必要としない場合があります。

さらに、ターゲット AWS Glue マネージドカタログでリソースポリシーを設定して、インバウンド統合を承認する必要があります。次に、以下の AWS CLI コマンドを実行してリソースポリシーを適用します。

ターゲットカタログのインバウンド統合を承認するサンプル AWS CLI コマンド

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
```

```
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Glue カタログの Amazon リソースネーム (ARN) 形式は次のようになります。
>
> - Glue カタログ – `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`

別のアカウントでターゲット AWS Glue マネージドカタログを選択する

別の AWS アカウント にあるターゲット AWS Glue マネージドカタログを指定する場合は、現在のアカウントのユーザーがターゲットアカウントのリソースにアクセスするのを許可するロールを作成する必要があります。詳細については、「所有している別の AWS アカウント の IAM ユーザーにアクセス権を付与する」を参照してください。

ロールには次のアクセス許可が必要です。これにより、ユーザーはターゲットアカウントで使用可能な AWS Glue カタログを表示できます。

必要なアクセス許可ポリシー

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
```

```
            "Effect":"Allow",
            "Action":[
               "glue:GetCatalog"
            ],
            "Resource":[
               "*"
            ]
         }
      ]
}
```

ロールには、ターゲットアカウント ID を指定する次の信頼ポリシーが必要です。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::111122223333:root"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成」を参照してください。

Amazon SageMaker Lakehouse とのゼロ ETL 統合の作成

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して Amazon SageMaker Lakehouse とのゼロ ETL 統合を作成できます。

> **Important**
>
> Amazon SageMaker Lakehouse とのゼロ ETL 統合は、更新または再同期オペレーションをサポートしていません。作成後に統合で問題が発生した場合は、統合を削除して新しい統合を作成する必要があります。

デフォルトで、RDS for MySQL はバイナリログファイルをすぐに消去します。ゼロ ETL 統合では、ソースからターゲットへのデータレプリケーションをバイナリログに依存しているため、ソースデータベースインスタンスの保持期間は 1 時間以上である必要があります。統合を作成したら、直ちに Amazon RDS は選択したソースデータベースのバイナリログファイルの保持期間を確認します。現在の値が 0 時間の場合、Amazon RDS は自動的に 1 時間に変更します。それ以外の場合、値は変わりません。

RDS コンソール

Amazon SageMaker Lakehouse とゼロ ETL 統合を作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 左側のナビゲーションペインから、[ゼロ ETL 統合] を選択します。
3. [ゼロ ETL 統合の作成] を選択します。
4. [統合 ID] に、統合の名前を入力します。名前には最大 63 文字の英数字を使用でき、ハイフンを含めることができます。
5. [次へ] を選択します。
6. [ソース] で、データの送信元となる RDS データベースを選択します。

> **Note**
>
> DBパラメータが正しく設定されていないと、RDS から通知されます。このメッセージを受け取った場合は、[Fix it for me] を選択するか、手動で設定することができます。手動で修正する手順については、「the section called “ステップ 1: カスタム DB のパラメータグループを作成する”」を参照してください。
> DB パラメータを変更するには再起動が必要です。統合を作成する前に、再起動が完了し、新しいパラメータ値のデータベースのへの適用が正常に完了している必要があります。

7. ソースデータベースが正常に設定されたら、[次へ] を選択します。
8. [ターゲット] で、以下を実行します。

   1. (オプション) Amazon SageMaker Lakehouse ターゲットとして別の AWS アカウントを使用するには、[別のアカウントを指定] を選択します。次に、AWS Glue カタログを表示するアクセス許可を持つ IAM ロールの ARN を入力します。IAM ロールの作成手順については、「the section called “別のアカウントでターゲット AWS Glue マネージドカタログを選択する”」を参照してください。
   2. AWS Glue カタログで、ソースデータベースからのレプリケートデータのターゲットを選択します。既存の AWS Glue マネージドカタログをターゲットとして選択できます。
   3. ターゲット IAM ロールは、ターゲットカタログに対する記述アクセス許可を必要とし、次のアクセス許可が必要です。

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Sid": "VisualEditor0",
                  "Effect": "Allow",
                  "Action": "glue:GetCatalog",
                  "Resource": [
                      "arn:aws:glue:us-east-1:111122223333:catalog/*",
                      "arn:aws:glue:us-east-1:111122223333:catalog"
                  ]
              }
          ]
      }
      ```

      ターゲット IAM ロールには、次の信頼関係が必要です。

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "glue.amazonaws.com"
      ```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. ステップ 3b: Amazon SageMaker Lakehouse ゼロ ETL 統合用の AWS Glue カタログを作成する で作成された Lake Formation 管理者ロールを使用して、ターゲット AWS Glue マネージドカタログの記述アクセス許可をターゲット IAM ロールに付与する必要があります。

> **Note**
>
> 指定された AWS Glue マネージドカタログのリソースポリシーまたは設定が正しく設定されていない場合、RDS から通知されます。このメッセージを受け取った場合は、[修正をお願いする] を選択するか、手動で設定することができます。
> 選択したソースとターゲットの AWS アカウント が異なる場合、Amazon RDS はこれらの設定を自動的に修正できません。他のアカウントに移動し、SageMaker Unified Studio で手動で修正する必要があります。

9. ターゲット AWS Glue マネージドカタログが正しく設定されたら、[次へ] を選択します。
10. (オプション) [タグ] で、1 つ以上のタグを統合に追加します。詳細については、「the section called “RDS リソースのタグ付け”」を参照してください。
11. [暗号化] として、統合の暗号化方法を指定します。デフォルトでは、RDS はすべての統合を AWS 所有のキー で暗号化します。代わりにカスタマーマネージドキーを選択するには、[暗号化設定のカスタマイズ] を有効にして、暗号化に使用する KMS キーを選択します。詳細については、「the section called “Amazon RDS リソースの暗号化”」を参照してください。

    オプションで、暗号化コンテキストを追加します。詳しくは、AWS Key Management Service デベロッパーガイドの Encryption context を参照してください。

> **Note**
>
> Amazon RDS は、追加する以下に加えて、次の暗号化コンテキストペアを追加します。
>
> - `aws:glue:integration:arn` - `IntegrationArn`
> - `aws:servicename:id` - `glue`

これにより、追加できるペアの総数が 8 から 6 に減り、許可の制約の全体的な文字数制限に影響します。詳細については、「AWS Key Management Service デベロッパーガイド」の「Using grant constraints」を参照してください。

12. [次へ] を選択します。
13. 統合設定を確認し、[ゼロ ETL 統合を作成] を選択します。

    作成に失敗した場合は、トラブルシューティングの手順について「the section called “ゼロ ETL 統合のトラブルシューティング”」を参照してください。

統合のステータスは、作成中は `Creating` であり、ターゲット Amazon SageMaker Lakehouse のステータスは `Modifying` です。この間、カタログをクエリしたり、設定を変更したりすることはできません。

統合が正常に作成されると、統合とターゲット Amazon SageMaker Lakehouse の両方のステータスが `Active` に変わります。

AWS CLI

AWS CLI を使用してゼロ ETL 統合用のターゲット AWS Glue マネージドカタログを準備するには、まず create-integration-resource-property コマンドを次のオプションとともに使用する必要があります。

- `--resource-arn` – 統合のターゲットとなる AWS Glue マネージドカタログの ARN を指定します。
- `--target-processing-properties` – ターゲット AWS Glue マネージドカタログにアクセスするための IAM ロールの ARN を指定します。

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

AWS CLI を使用して Amazon SageMaker Lakehouse とのゼロ ETL 統合を作成するには、create-integration コマンドに以下のオプションを指定して使用します。

- `--integration-name` — 統合の名前を指定します。

- `--source-arn` — 統合のソースとなる RDS データベースの ARN を指定します。
- `--target-arn` – 統合のターゲットとなる AWS Glue マネージドカタログの ARN を指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Windows の場合:

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

RDS API

Amazon RDS API を使用して Amazon SageMaker とのゼロ ETL 統合を作成するには、以下のパラメータを指定して `CreateIntegration` オペレーションを使用します。

> **Note**
>
> カタログ名は 19 文字に制限されています。カタログ名として使用する場合は、IntegrationName パラメータがこの要件を満たしていることを確認します。

- `IntegrationName` — 統合の名前を指定します。
- `SourceArn` — 統合のソースとなる RDS データベースの ARN を指定します。
- `TargetArn` – 統合のターゲットとなる AWS Glue マネージドカタログの ARN を指定します。

カスタマーマネージドキーでの統合の暗号化

Amazon SageMaker との統合の作成時に AWS 所有のキー ではなくカスタム KMS キーを指定する場合、キーポリシーは SageMaker Unified Studio サービスプリンシパルに `CreateGrant` アクションへのアクセスを提供する必要があります。さらに、現在のユーザーに `DescribeKey` アクションと `CreateGrant` アクションの実行を許可する必要があります。

次のポリシー例は、キーポリシーに必要なアクセス許可を指定する方法を示しています。アクセス許可の範囲をさらに狭めるためのコンテキストキーが含まれています。

キーポリシーの例

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "EnablesIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "GlueServicePrincipalAddGrant",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
```

```
                    "GenerateDataKey",
                    "CreateGrant"
                ]
            }
        }
    },
    {
        "Sid": "AllowsCurrentUserRoleAddGrantKMSKey",
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/{role-name}"
        },
        "Action": "kms:CreateGrant",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:{context-key}":"{context-value}",
                "kms:ViaService": "rds.us-east-1.amazonaws.com"
            },
            "ForAllValues:StringEquals": {
                "kms:GrantOperations": [
                    "Decrypt",
                    "GenerateDataKey",
                    "CreateGrant"
                ]
            }
        }
    },
    {
        "Sid": "AllowsCurrentUserRoleRetrieveKMSKeyInformation",
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/{role-name}"
        },
        "Action": "kms:DescribeKey",
        "Resource": "*"
    }
]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「キーポリシーの作成」を参照してください。

次のステップ

Amazon SageMaker とのゼロ ETL 統合を正常に作成したら、ソース RDS データベースへのデータの追加と、Amazon SageMaker Lakehouse でのクエリを開始できます。データは自動的にレプリケートされ、分析や機械学習のワークロードで利用できるようになります。

Amazon RDS ゼロ ETL 統合でのデータフィルタリング

Amazon RDS ゼロ ETL 統合はデータフィルタリングをサポートしているため、ソース Amazon RDS データベースからターゲットデータウェアハウスにレプリケートされるデータを制御できます。データベース全体をレプリケートする代わりに、1 つ以上のフィルターを適用して、特定のテーブルを選択的に含めたり除外したりできます。これにより、関連するデータのみが転送されるようにすることで、ストレージとクエリのパフォーマンスを最適化できます。現在、フィルタリングはデータベースレベルとテーブルレベルに制限されています。列レベルと行レベルのフィルタリングはサポートされていません。

データフィルタリングは、次のような場合に便利です。

- 2 つ以上の異なるソースデータベースの特定のテーブルを結合し、いずれのデータベースのデータ全体は必要ない場合。
- データベース全体ではなく、テーブルのサブセットのみを使用して分析を行うことで、コストを節約する場合。
- 電話番号、住所、クレジットカード情報などの機密情報を特定のテーブルから除外する場合。

ゼロ ETL 統合には、AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または RDS API を使用して、データフィルターを追加できます。

統合でプロビジョンしたクラスターをターゲットとして使用している場合、データフィルタリングを使用するには、クラスターがパッチ 180 以降である必要があります。

トピック

- データフィルターの形式
- フィルター論理
- フィルターの優先順位
- RDS for MySQL の例
- RDS for PostgreSQL の例

- RDS for Oracle の例
- 統合へのデータフィルターの追加
- 統合からのデータフィルターの削除

データフィルターの形式

1 つの統合に対して複数のフィルターを定義できます。各フィルターは、フィルター式のパターンのいずれかに一致する既存および今後利用するデータベーステーブルを含めるまたは除外します。Amazon RDS ゼロ ETL 統合では、データフィルタリングに Maxwell フィルター構文を使用します。

各フィルターには以下の要素が含まれます。

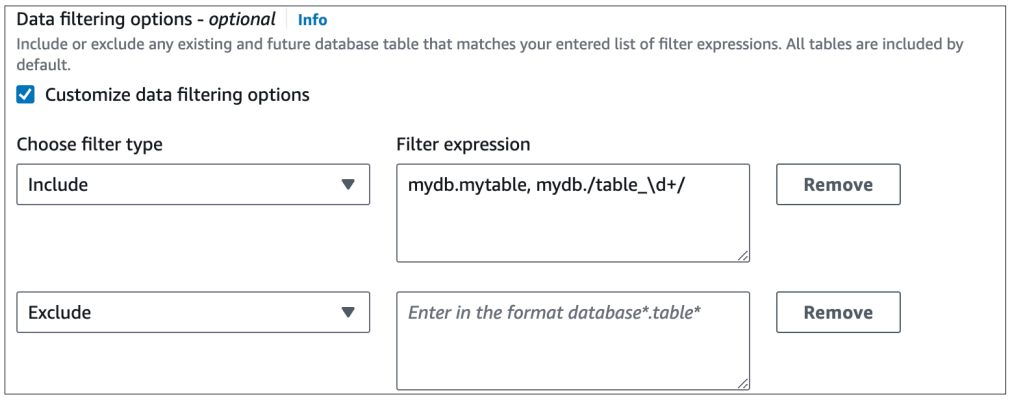
Element	説明
フィルタータイプ	`Include` フィルタータイプは、フィルター式のパターンのいずれかに一致するすべてのテーブルを含めます。`Exclude` フィルタータイプは、いずれかのパターンに一致するすべてのテーブルを除外します。
フィルター式	コンマ区切りのパターンのリスト。式では Maxwell フィルター構文を使用する必要があります。
パターン	フィルターパターンの形式は *`database`*`.`*`table`* (RDS for MySQL の場合) または *`database`*`.`*`schema`*`.`*`table`* (RDS for PostgreSQL の場合)。リテラル名を指定するか、正規表現を定義できます。 **Note** RDS for MySQL の場合、正規表現はデータベース名とテーブル名の両方でサポートされています。RDS for PostgreSQL の場合、正規表現はス

Element	説明
	キーマ名とテーブル名でのみサポートされ、データベース名ではサポートされていません。 列レベルのフィルターや拒否リストを含めることはできません。 1 つの統合に含めることができるパターンの合計数は、最大 99 個です。コンソールでは、パターンを 1 つのフィルター式に含めることも、複数の式に分散することもできます。1 つのパターンの長さは 256 文字を超えることはできません。

Important

RDS for PostgreSQL ソースデータベースを選択する場合は、少なくとも 1 つのデータフィルターパターンを指定する必要があります。このパターンには、ターゲットデータウェアハウスへのレプリケーション用のデータベース (*database-name*.*.*) が少なくとも 1 つ含まれている必要があります。

次の図は、コンソールでの RDS for MySQL データフィルターの構造を示しています。



> **Important**
>
> フィルターパターンには、個人を特定する情報、または機密情報を含めないでください。

AWS CLI のデータフィルター

AWS CLI を使用してデータフィルターを追加する場合、構文はコンソールと少し異なります。1 つのフィルタータイプで複数のパターンをグループ化できないように、各パターンにフィルタータイプ (`Include` または `Exclude`) を個別に割り当てる必要があります。

例えば、コンソールでは、以下のカンマで区切られたパターンを 1 つの `Include` ステートメントにまとめることができます。

RDS for MySQL

```
mydb.mytable, mydb./table_\d+/
```

RDS for PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

ただし AWS CLI を使用する際は、データフィルターを次のように記述する必要があります。

RDS for MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

RDS for PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

フィルター論理

統合でデータフィルターを指定しない場合、Amazon RDS は `include:*.*` をデフォルトのフィルターと見なし、すべてのテーブルをターゲットデータウェアハウスに複製します。ただし、少なくとも 1 つのフィルターを追加すると、デフォルトのロジックは `exclude:*.*` に切り替わり、デフォルトですべてのテーブルが除外されます。これにより、レプリケーションに含めるデータベースとテーブルを明示的に定義できます。

例えば、次のフィルターを定義する場合、

```
'include: db.table1, include: db.table2'
```

Amazon RDS は、フィルターを次のように解釈します。

```
'exclude:*.*, include: db.table1, include: db.table2'
```

結果として、Amazon RDS は db という名前のデータベースから table1 および table2 のみをターゲットデータウェアハウスにレプリケートします。

フィルターの優先順位

Amazon RDS は、指定された順番にデータフィルターを適用します。AWS マネジメントコンソール では、フィルタ式を左から右、上から下へと処理します。2 番目のフィルターまたは最初のフィルターに続く個々のパターンは、それを上書きできます。

例えば、最初のフィルターが Include books.stephenking の場合、books データベースの stephenking テーブルのみが含まれます。ただし、2 つ目のフィルター、Exclude books.* を追加すると、最初のフィルターが上書きされます。これにより、books インデックスのテーブルがターゲットデータウェアハウスにレプリケートされるのを防ぐことができます。

1 つまたは複数のフィルターを指定すると、ロジックはデフォルトで exclude:*.* を前提として開始されます。そのため、すべてのテーブルが自動的にレプリケーションから除外されます。ベストプラクティスとして、最も広範なフィルターから最も具体的なフィルターを定義します。1 つ以上の Include ステートメントから始めてレプリケートするデータを指定し、Exclude フィルターを追加して特定のテーブルを選択的に削除します。

AWS CLI を使用して定義するフィルターにも同じ原則が適用されます。Amazon RDS は、これらのフィルターパターンを指定された順番で適用するため、あるパターンによってその前に指定されたフィルターパターンが上書きされる場合があります。

RDS for MySQL の例

以下の例は、RDS for MySQL の例におけるゼロ ETL 統合でのデータフィルタリングの仕組みを示しています。

- すべてのデータベースとすべてのテーブルを含める。

```
'include: *.*'
```

- books データベース内のすべてのテーブルを含める。

```
'include: books.*'
```

- mystery という名前のすべてのテーブルを除外します。

```
'include: *.*, exclude: *.mystery'
```

- books データベース内の 2 つの特定のテーブルを含める。

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- サブストリング mystery を含んでいるものを除き、books データベース内のすべてのテーブルを含めます。

```
'include: books.*, exclude: books./.*mystery.*/'
```

- mystery で始まるものを除き、books データベース内のすべてのテーブルを含めます。

```
'include: books.*, exclude: books./mystery.*/'
```

- mystery で終わるものを除き、books データベース内のすべてのテーブルを含めます。

```
'include: books.*, exclude: books./.*mystery/'
```

- table_stephen_king という名前のテーブルを除き、table_ で始まる books データベース内の すべてのテーブルを含めます。例えば、table_movies や table_books はレプリケートされますが、table_stephen_king はレプリケートされません。

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

RDS for PostgreSQL の例

以下の例は、RDS for PostgreSQL ゼロ ETL 統合でのデータフィルタリングの仕組みを示しています。

- books データベース内のすべてのテーブルを含めます。

```
'include: books.*.*'
```

- books データベース内の mystery という名前のテーブルをすべて除外します。

```
'include: books.*.*, exclude: books.*.mystery'
```

- books データベース内の mystery スキーマの 1 つのテーブルを含め、employee データベース内の finance スキーマの 1 つのテーブルを含めます。

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- サブストリング king を含むテーブルを除き、books データベースおよび science_fiction スキーマ内のすべてのテーブルを含めます。

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- sci で始まるスキーマ名を持つテーブルを除き、books データベース内のすべてのテーブルを含めます。

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- mystery スキーマ内の king で終わるテーブルを除き、books データベース内のすべてのテーブルを含めます。

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- table_stephen_king という名前のテーブルを除き、table_ で始まる books データベース内のすべてのテーブルを含めます。例えば、fiction スキーマの table_movies と mystery スキーマの table_books はレプリケートされますが、どちらのスキーマでも table_stephen_king はレプリケートされません。

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

RDS for Oracle の例

以下の例は、RDS for Oracle ゼロ ETL 統合でのデータフィルタリングの仕組みを示しています。

- books データベース内のすべてのテーブルを含めます。

```
'include: books.*.*'
```

- books データベース内の mystery という名前のテーブルをすべて除外します。

```
'include: books.*.*, exclude: books.*.mystery'
```

- books データベース内の mystery スキーマのテーブルを 1 つ含め、employee データベース内の finance スキーマのテーブルを 1 つ含めます。

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- books データベース内の mystery スキーマのすべてのテーブルを含めます。

```
'include: books.mystery.*'
```

大文字と小文字の区別に関する考慮事項

Oracle Database と Amazon Redshift では、オブジェクト名の大文字と小文字の処理方法が異なり、これはデータフィルター設定とターゲットクエリの両方に影響します。次の点に注意してください。

- Oracle Database は、CREATE ステートメントで明示的に引用符で囲まれていない限り、データベース、スキーマ、およびオブジェクト名を大文字で保存します。例えば、mytable (引用符なし) を作成する場合、Oracle データディクショナリはテーブル名を MYTABLE として保存します。オブジェクト名を引用すると、データディクショナリは大文字と小文字を保持します。
- ゼロ ETL データフィルターでは大文字と小文字が区別され、Oracle データディクショナリに表示されるオブジェクト名の大文字と小文字が正確に一致する必要があります。
- Amazon Redshift クエリは、明示的に引用符で囲まれていない限り、デフォルトで小文字のオブジェクト名になります。例えば、MYTABLE (引用符なし) のクエリは mytable を検索します。

Amazon Redshift フィルターを作成してデータをクエリするときは、大文字と小文字の違いに注意してください。

大文字の統合の作成

二重引用符で名前を指定せずにテーブルを作成すると、Oracle データベースは名前をデータディクショナリに大文字で保存します。例えば、次のいずれかの SQL ステートメントを使用して MYTABLE を作成できます。

```
CREATE TABLE REINVENT.MYTABLE (id NUMBER PRIMARY KEY, description VARCHAR2(100));
CREATE TABLE reinvent.mytable (id NUMBER PRIMARY KEY, description VARCHAR2(100));
CREATE TABLE REinvent.MyTable (id NUMBER PRIMARY KEY, description VARCHAR2(100));
CREATE TABLE reINVENT.MYtabLE (id NUMBER PRIMARY KEY, description VARCHAR2(100));
```

上記のステートメントでテーブル名を引用しなかったため、Oracle データベースはオブジェクト名を大文字で MYTABLE として保存します。

このテーブルを Amazon Redshift にレプリケートするには、create-integration コマンドのデータフィルターで大文字の名前を指定する必要があります。ゼロ ETL フィルター名と Oracle データディクショナリ名は一致する必要があります。

```
aws rds create-integration \
  --integration-name upperIntegration \
  --data-filter "include: ORCL.REINVENT.MYTABLE" \
...
```

デフォルトでは、Amazon Redshift はデータを小文字で保存します。Amazon Redshift のレプリケートされたデータベースで MYTABLE をクエリするには、Oracle データディクショナリのケースと一致するように大文字名 MYTABLE を引用符で囲む必要があります。

```
SELECT * FROM targetdb1."REINVENT"."MYTABLE";
```

次のクエリでは、引用符メカニズムを使用しません。これらは、デフォルトの小文字名を使用する mytable という名前の Amazon Redshift テーブルを検索するため エラーを返しますが、Oracle データディクショナリではそのテーブルの名前は MYTABLE です。

```
SELECT * FROM targetdb1."REINVENT".MYTABLE;
SELECT * FROM targetdb1."REINVENT".MyTable;
SELECT * FROM targetdb1."REINVENT".mytable;
```

次のクエリでは、引用符メカニズムを使用して大文字と小文字が混在する名前を指定します。クエリはすべて、MYTABLE という名前ではない Amazon Redshift テーブルを検索するため、エラーを返します。

```
SELECT * FROM targetdb1."REINVENT"."MYtablE";
SELECT * FROM targetdb1."REINVENT"."MyTable";
SELECT * FROM targetdb1."REINVENT"."mytable";
```

小文字の統合の作成

次の代替例では、二重引用符を使用して、Oracle データディクショナリにテーブル名を小文字で保存します。次のように mytable を作成します。

```
CREATE TABLE REINVENT."mytable" (id NUMBER PRIMARY KEY, description VARCHAR2(100));
```

Oracle データベースは、テーブル名 mytable を小文字として保存します。このテーブルを Amazon Redshift にレプリケートするには、ゼロ ETL データフィルターで小文字の名前 mytable を指定する必要があります。

```
aws rds create-integration \
  --integration-name lowerIntegration \
  --data-filter "include: ORCL.REINVENT.mytable" \
...
```

Amazon Redshift のレプリケートされたデータベースでこのテーブルをクエリするときは、小文字の名前 mytable を指定できます。クエリは、Oracle データディクショナリのテーブル名である mytable という名前のテーブルを検索するため、成功します。

```
SELECT * FROM targetdb1."REINVENT".mytable;
```

Amazon Redshift のデフォルトは小文字のオブジェクト名であるため、次のクエリも mytable の検索に成功します。

```
SELECT * FROM targetdb1."REINVENT".MYtablE;
SELECT * FROM targetdb1."REINVENT".MYTABLE;
SELECT * FROM targetdb1."REINVENT".MyTable;
```

次のクエリでは、オブジェクト名の引用符メカニズムを使用します。名前が mytable と異なる Amazon Redshift テーブルを検索するため、すべてエラーが返されます。

```
SELECT * FROM targetdb1."REINVENT"."MYTABLE";
SELECT * FROM targetdb1."REINVENT"."MyTable";
SELECT * FROM targetdb1."REINVENT"."MYtablE";
```

大文字と小文字が混在する統合でテーブルを作成する

次の例では、二重引用符を使用して、Oracle データディクショナリにテーブル名を小文字で保存します。次のように MyTable を作成します。

```
CREATE TABLE REINVENT."MyTable" (id NUMBER PRIMARY KEY, description VARCHAR2(100));
```

Oracle データベースでは、このテーブル名が大文字と小文字が混在した MyTable として保存されます。このテーブルを Amazon Redshift にレプリケートするには、データフィルターで大文字と小文字が混在した名前を指定する必要があります。

```
aws rds create-integration \
  --integration-name mixedIntegration \
  --data-filter "include: ORCL.REINVENT.MyTable" \
...
```

Amazon Redshift のレプリケートされたデータベースでこのテーブルをクエリする場合は、オブジェクト名を引用符で囲んで大文字と小文字が混在した名前 MyTable を指定する必要があります。

```
SELECT * FROM targetdb1."REINVENT"."MyTable";
```

Amazon Redshift のデフォルトは小文字のオブジェクト名であるため、次のクエリは小文字の名前 mytable を検索しているため、オブジェクトを見つけられません。

```
SELECT * FROM targetdb1."REINVENT".MYtablE;
SELECT * FROM targetdb1."REINVENT".MYTABLE;
SELECT * FROM targetdb1."REINVENT".mytable;
```

> **Note**
>
> RDS for Oracle 統合では、データベース名、スキーマ、またはテーブル名のフィルター値に正規表現を使用することはできません。

統合へのデータフィルターの追加

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用してデータフィルタリングを設定できます。

> **Important**
>
> 統合の作成後にフィルターを追加すると、Amazon RDS はフィルターがもともと存在していたものであるかのようにフィルターを扱います。新しいフィルタリング条件に一致しない

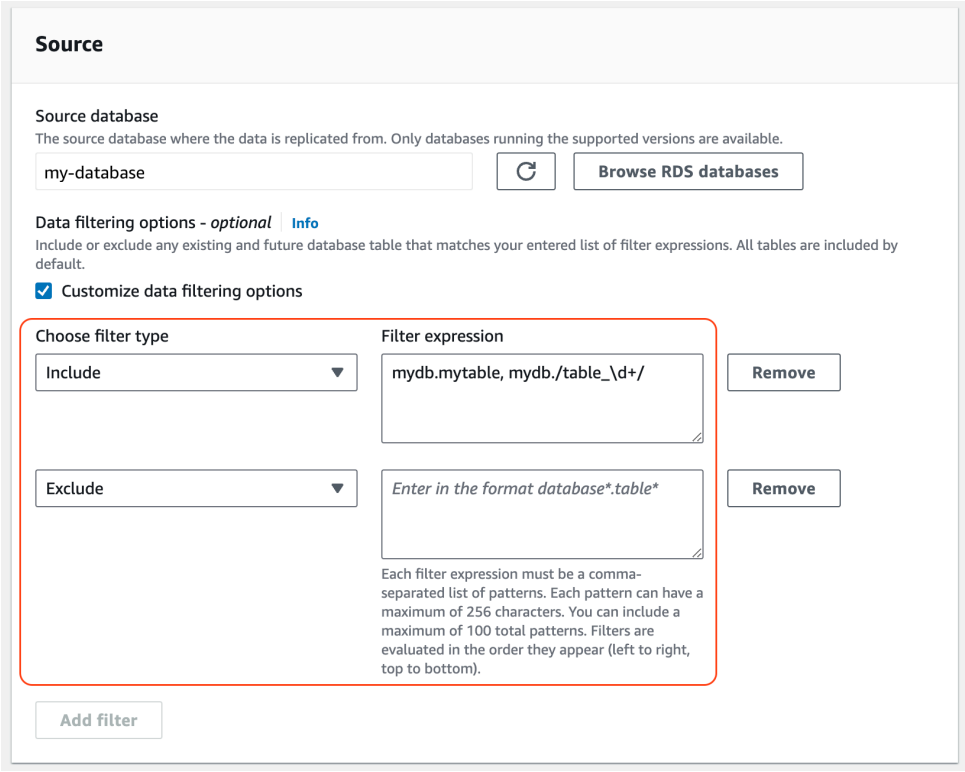
ターゲットデータウェアハウスのデータを削除し、影響を受けるすべてのテーブルを再同期します。

RDS コンソール

ゼロ ETL 統合にデータフィルターを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインから、[ゼロ ETL 統合] を選択します。データフィルターを追加する統合を選択して [変更] を選択します。
3. [ソース] で、1 つまたは複数の `Include` ステートメントと `Exclude` ステートメントを追加します。

   次の図は、MySQL 統合のデータフィルターの例を示しています。



4. 変更が適切であることを確認したら、[続行] および [変更を保存] を選択します。

AWS CLI

AWS CLI を使用してゼロ ETL 統合にデータフィルターを追加するには、modify-integration コマンドを呼び出します。統合 ID に加えて、Include および Exclude の Maxwell フィルターのカンマ区切りリストで --data-filter パラメーターを指定します。

Example

次の例は、my-integration にフィルターパターンを追加します。

Linux、macOS、Unix の場合:

```
aws rds modify-integration \
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Windows の場合:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

RDS API

RDS API を使用してゼロ ETL 統合を変更するには、ModifyIntegration オペレーションを呼び出します。統合 ID を指定し、フィルターパターンのカンマ区切りリストを指定します。

統合からのデータフィルターの削除

統合からデータフィルターを削除すると、Amazon RDS は削除したフィルターが存在しなかったかのようにそれ以外のフィルターを適用し始めます。次に、基準を満たした以前に除外されたデータをターゲットデータウェアハウスにレプリケートします。これにより、影響を受けるすべてのテーブルの再同期がトリガーされます。

ソース RDS データベースへのデータの追加とクエリ

Amazon RDS から Amazon Redshift にデータをレプリケートするゼロ ETL 統合の作成を終了するには、ターゲット送信先にデータベースを作成する必要があります。

Amazon Redshift と接続するため、Amazon Redshift クラスターまたはワークグループに接続し、統合識別子を参照してデータベースを作成します。これで、ソースの RDS データベースにデータを追加でき、Amazon Redshift または Amazon SageMaker で複製されることが確認できます。

トピック

- ターゲットデータベースの作成
- ソースデータベースへのデータの追加
- Amazon Redshift での Amazon RDS データのクエリ
- RDS データベースと Amazon Redshift データベース間のデータタイプの相違点
- RDS for PostgreSQL の DDL オペレーション

ターゲットデータベースの作成

Amazon Redshift へのデータの複製を開始する前に、統合を作成後、データベースをターゲットのデータウェアハウスに作成する必要があります。このデータベースには、統合識別子への参照が含まれている必要があります。Amazon Redshift コンソールまたはクエリエディタ v2 を使用して、データベースを作成することができます。

デスティネーションデータベースを作成する手順については、「Amazon Redshift にデスティネーションデータベースを作成する」を参照してください。

ソースデータベースへのデータの追加

統合を設定した後で、データウェアハウスにレプリケートするデータを、ソースの RDS データベースに入力できます。

> **Note**
>
> Amazon RDS のデータ型とターゲット分析ウェアハウスには違いがあります。データ型マッピングの表については、「the section called “データタイプの相違点”」を参照してください。

まず、任意の MySQL クライアントを使用して、ソースデータベースに接続します。手順については、「the section called “MySQL を実行している DB インスタンスへの接続”」を参照してください。

次に、テーブルを作成し、1 行のサンプルデータを挿入します。

> **Important**
>
> テーブルにプライマリキーがあることを確認してください。そうしないと、ターゲットのデータウェアハウスに複製できません。

RDS for MySQL

次の例では、MySQL Workbench ユーティリティを使用しています。

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

RDS for PostgreSQL

次の例では、psql PostgreSQL インタラクティブターミナルを使用しています。データベースに接続するときは、レプリケートするデータベース名を含めます。

```
psql -h mydatabase.123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username -
d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

named_db=> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

RDS for Oracle:

次の例では、SQL*Plus を使用して RDS for Oracle データベースに接続します。

```
sqlplus 'user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))'

SQL> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

SQL> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

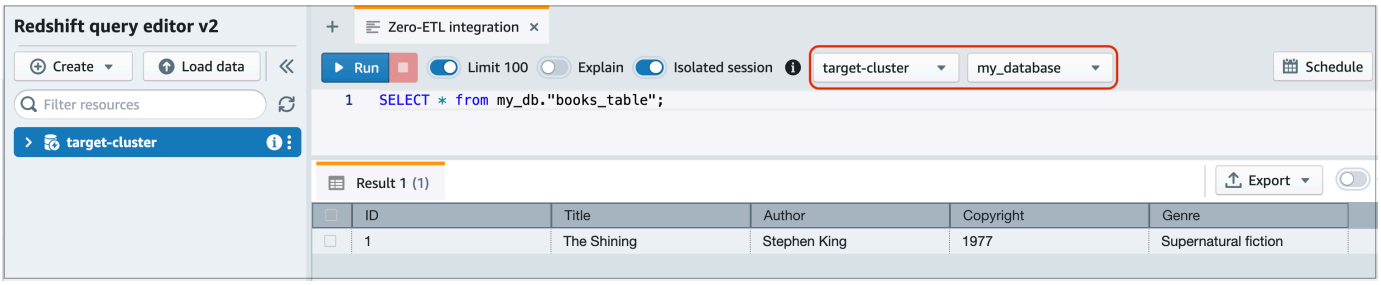
Amazon Redshift での Amazon RDS データのクエリ

RDS データベースにデータを追加すると、送信先データベースにレプリケートされ、クエリを実行できるようになります。

複製されたデータをクエリするには

1. Amazon Redshift コンソールに移動し、左側のナビゲーションペインから [クエリエディタ v2] を選択します。
2. クラスターまたはワークグループに接続し、ドロップダウンメニュー (この例では destination_database) から送信先データベース (統合から作成したもの) を選択します。デスティネーションデータベースを作成する手順については、「Amazon Redshift にデスティネーションデータベースを作成する」を参照してください。
3. SELECT ステートメントを使用してデータをクエリします。この例では、次のコマンドを実行して、ソースの RDS データベースで作成したテーブルからすべてのデータを選択します。

```
SELECT * from my_db."books_table";
```



- *my_db* は RDS データベーススキーマ名です。
- *books_table* は RDS テーブル名です。

コマンドラインクライアントを使用してデータをクエリすることもできます。例えば、次のようになります。

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                 Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+-----------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> 大文字と小文字を区別するには、スキーマ、テーブル、および列の名前を二重引用符 (" ") で囲みます。詳細については、「enable_case_sensitive_identifier」を参照してください。

RDS データベースと Amazon Redshift データベース間のデータタイプの相違点

以下の表は、RDS for MySQL、RDS for PostgreSQL、RDS for Oracle のデータタイプおよび対応する送信先のデータタイプのマッピングを示しています。Amazon RDS は現在、ゼロ ETL 統合ではこれらのデータ型のみをサポートしています。

ソースデータベースのテーブルにサポートされていないデータ型が含まれている場合、そのテーブルは同期されず、送信先ターゲットで使用できなくなります。ソースからターゲットへのストリーミングは継続されますが、サポートされていないデータ型のテーブルは使用できません。テーブルを修正してターゲット送信先で使用できるようにするには、変更内容を手動で元に戻し、`ALTER DATABASE...INTEGRATION REFRESH` を実行して統合を更新する必要があります。

> **Note**
>
> Amazon SageMaker Lakehouse とのゼロ ETL 統合を更新することはできません。代わりに、統合を削除して再度作成してみてください。

トピック

- RDS for MySQL
- RDS for PostgreSQL
- RDS for Oracle

RDS for MySQL

RDS for MySQL データ型	ターゲットのデータ型	説明	制限事項
INT	INTEGER	符号付き 4 バイト整数	なし
SMALLINT	SMALLINT	符号付き 2 バイト整数	なし
TINYINT	SMALLINT	符号付き 2 バイト整数	なし
MEDIUMINT	INTEGER	符号付き 4 バイト整数	なし
BIGINT	BIGINT	符号付き 8 バイト整数	なし
INT UNSIGNED	BIGINT	符号付き 8 バイト整数	なし
TINYINT UNSIGNED	SMALLINT	符号付き 2 バイト整数	なし
MEDIUMINT UNSIGNED	INTEGER	符号付き 4 バイト整数	なし
BIGINT UNSIGNED	DECIMAL(20,0)	精度の選択が可能な真数	なし
DECIMAL(p,s) = NUMERIC(p,s)	DECIMAL(p,s)	精度の選択が可能な真数	精度が 38 より大きく、スケールが 37 より大きい

RDS for MySQL データ型	ターゲットのデータ型	説明	制限事項
			場合、サポートされない
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL(p,s)	精度の選択が可能な真数	精度が 38 より大きく、スケールが 37 より大きい場合、サポートされない
FLOAT4/REAL	REAL	単精度浮動小数点数	なし
FLOAT4/REAL UNSIGNED	REAL	単精度浮動小数点数	なし
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	倍精度浮動小数点数	なし
DOUBLE/REAL/FLOAT8 UNSIGNED	DOUBLE PRECISION	倍精度浮動小数点数	なし
BIT(n)	VARBYTE(8)	可変長バイナリ値	なし
BINARY(n)	VARBYTE(n)	可変長バイナリ値	なし
VARBINARY (n)	VARBYTE(n)	可変長バイナリ値	なし
CHAR(n)	VARCHAR(n)	可変長文字列値	なし
VARCHAR(n)	VARCHAR(n)	可変長文字列値	なし
TEXT	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし

RDS for MySQL データ型	ターゲットのデータ型	説明	制限事項
TINYTEXT	VARCHAR(255)	最大 255 文字の可変長文字列値	なし
MEDIUMTEXT	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
LONGTEXT	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
ENUM	VARCHAR(1,020)	最大 1,020 文字の可変長文字列値	なし
SET	VARCHAR(1,020)	最大 1,020 文字の可変長文字列値	なし
DATE	DATE	カレンダー日付 (年、月、日)	なし
DATETIME	TIMESTAMP	日付と時刻 (タイムゾーンなし)	なし
TIMESTAMP(p)	TIMESTAMP	日付と時刻 (タイムゾーンなし)	なし
TIME	VARCHAR(18)	最大 18 文字の可変長文字列値	なし
YEAR	VARCHAR(4)	最大 4 文字の可変長文字列値	なし
JSON	SUPER	値としての半構造化データまたは文書	なし

RDS for PostgreSQL

RDS for PostgreSQL のゼロ ETL 統合は、カスタムデータタイプ、または拡張機能で作成したデータタイプをサポートしていません。

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
array	SUPER	値としての半構造化データまたは文書	なし
bigint	BIGINT	符号付き 8 バイト整数	なし
bigserial	BIGINT	符号付き 8 バイト整数	なし
bit varying(n)	VARBYTE(n)	最大 16,777,216 バイトの可変長バイナリ値	なし
bit(n)	VARBYTE(n)	最大 16,777,216 バイトの可変長バイナリ値	なし
BIT、BIT VARYING	VARBYTE(16777216)	最大 16,777,216 バイトの可変長バイナリ値	なし
boolean	BOOLEAN	論理ブール演算型 (true/false)	なし
bytea	VARBYTE(16777216)	最大 16,777,216 バイトの可変長バイナリ値	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
char(n)	CHAR(n)	最大 65,535 バイトの固定長文字列値	なし
char varying(n)	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
cid	BIGINT	符号付き 8 バイト整数	なし
cidr	VARCHAR(19)	最大 19 文字の可変長文字列値	なし
date	DATE	カレンダー日付 (年、月、日)	294,276 A.D. より大きい値はサポートされません
double precision	DOUBLE PRECISION	倍精度浮動小数点数	非正規値は完全にはサポートされません
gtsvector	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
inet	VARCHAR(19)	最大 19 文字の可変長文字列値	なし
integer	INTEGER	符号付き 4 バイト整数	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
int2vector	SUPER	値としての半構造化データまたはドキュメント。	なし
interval	INTERVAL	期間	year to month 修飾子または day to second 修飾子を指定する INTERVAL タイプのみがサポートされます。
json	SUPER	値としての半構造化データまたは文書	なし
JSONB	SUPER	値としての半構造化データまたは文書	なし
jsonpath	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
macaddr	VARCHAR(17)	最大 17 文字の可変長文字列値	なし
macaddr8	VARCHAR(23)	最大 23 文字の可変長文字列値	なし
money	DECIMAL(20,3)	通貨額	なし
名前	VARCHAR(64)	最大 64 文字の可変長文字列値	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
numeric(p,s)	DECIMAL(p,s)	ユーザー定義の固定精度値	• NaN 値はサポートされていません • 精度とスケールは明示的に定義し、38 (精度) と 37 (スケール) 以下にする必要があります。 • 負のスケールはサポートされていません
oid	BIGINT	符号付き 8 バイト整数	なし
oidvector	SUPER	値としての半構造化データまたはドキュメント。	なし
pg_brin_bloom_summary	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
pg_dependencies	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
pg_lsn	VARCHAR(17)	最大 17 文字の可変長文字列値	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
pg_mcv_list	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
pg_ndistinct	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
pg_node_tree	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
pg_snapshot	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
real	REAL	単精度浮動小数点数	非正規値は完全にはサポートされません
refcursor	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
smallint	SMALLINT	符号付き 2 バイト整数	なし
smallserial	SMALLINT	符号付き 2 バイト整数	なし
シリアル	INTEGER	符号付き 4 バイト整数	なし
テキスト	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
tid	VARCHAR(23)	最大 23 文字の可変長文字列値	なし
time [(p)] without time zone	VARCHAR(19)	最大 19 文字の可変長文字列値	`Infinity` および `-Infinity` 値はサポートされていません
time [(p)] with time zone	VARCHAR(22)	最大 22 文字の可変長文字列値	`Infinity` および `-Infinity` 値はサポートされていません
timestamp [(p)] without time zone	TIMESTAMP	日付と時刻 (タイムゾーンなし)	• `Infinity` および `-Infinity` 値はサポートされていません • サポートされていない `9999-12-31` より大きい値 • B.C. 値はサポートされていません

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
timestamp [(p)] with time zone	TIMESTAMPTZ	日付と時刻 (タイムゾーンあり)	• `Infinity` および `-Infinity` 値はサポートされていません • サポートされていない `9999-12-31` より大きい値 • B.C. 値はサポートされていません
tsquery	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
tsvector	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
txid_snapshot	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし
uuid	VARCHAR(36)	可変長 36 文字の文字列	なし
xid	BIGINT	符号付き 8 バイト整数	なし
xid8	DECIMAL(20, 0)	固定精度 10 進数	なし

RDS for PostgreSQL データタイプ	Amazon Redshift データタイプ	説明	制限事項
xml	VARCHAR(65,535)	最大 65,535 文字の可変長文字列値	なし

RDS for Oracle

サポートされていないデータタイプ

以下の RDS for Oracle データタイプは、Amazon Redshift ではサポートされていません。

- ANYDATA
- BFILE
- REF
- ROWID
- UROWID
- VARRAY
- SDO_GEOMETRY
- ユーザー定義のデータタイプ

データタイプの相違点

次の表は、RDS for Oracle がソースで Amazon Redshift がターゲットである場合に、ゼロ ETL 統合に影響するデータタイプの相違点を示しています。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
BINARY_FLOAT	FLOAT4
BINARY_DOUBLE	FLOAT8
BINARY	VARCHAR (長さ)
FLOAT (P)	精度が 24 以下の場合は FLOAT4。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
	精度が 24 を超える場合は FLOAT8。
NUMBER (P,S)	スケールが 0 以上、37 以下の場合は NUMERIC (p,s)。 スケールが 38 以上、127 以下の場合は VARCHAR (長さ)。 スケールが 0 の場合: • 精度が 2 以下の場合は INT2。 • 精度が 2 より大きく、4 以下の場合は INT2。 • 精度が 4 より大きく、9 以下の場合は INT4。 • 精度が 9 より大きい場合は NUMERIC (p,s)。 • 精度がスケール以上の場合は NUMERIC (p,s)。 スケールが 0 未満の場合は INT8。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
DATE	スケールが 0 以上、6 以下の場合は、Redshift ターゲット列のタイプに応じて、次のいずれかになる。 • TIMESTAMP (s) • TIMESTAMPTZ (s) — ソースタイムスタンプにゾーンオフセットが含まれている場合 (SQL Server や Oracle など)、insert または update 時に UTC に変換される。オフセットが含まれていない場合、時刻は既に UTC に変換済みとみなされる。 スケールが 7 以上、9 以下の場合は VARCHAR (37)。
INTERVAL_YEAR TO MONTH	長さが 1 ~ 65,535 の場合は VARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は VARCHAR (65535)。
INTERVAL_DAY TO SECOND	長さが 1 ~ 65,535 の場合は VARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は VARCHAR (65535)。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
TIMESTAMP	スケールが 0 以上、6 以下の場合は、Redshift ターゲット列のタイプに応じて、次のいずれかになる。 • TIMESTAMP (s) • TIMESTAMPTZ (s) — ソースタイムスタンプにゾーンオフセットが含まれている場合 (SQL Server や Oracle など)、insert または update 時に UTC に変換される。オフセットが含まれていない場合、時刻は既に UTC に変換済みとみなされる。 スケールが 7 以上、9 以下の場合は VARCHAR (37)。
TIMESTAMP WITH TIME ZONE	長さが 1 ~ 65,535 の場合は VARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は VARCHAR (65535)。
タイムスタンプ現地時間帯あり	長さが 1 ~ 65,535 の場合は VARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は VARCHAR (65535)。
CHAR	長さが 1 ~ 65,535 の場合は VARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は VARCHAR (65535)。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
VARCHAR2	長さが 4,000 バイトを超える場合は VARCHAR (最大 LOB サイズ)。最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。 長さが 4,000 バイト以下の場合は VARCHAR (バイト単位の長さ)。
NCHAR	長さが 1 ~ 65,535 の場合は NVARCHAR (バイト単位の長さ)。 長さが 65,536 ~ 2,147,483,647 の場合は NVARCHAR (65535)。
NVARCHAR2	長さが 4,000 バイトを超える場合は NVARCHAR (最大 LOB サイズ)。最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。 長さが 4,000 バイト以下の場合は NVARCHAR (バイト単位の長さ)。
RAW	VARCHAR (長さ)
REAL	FLOAT8
BLOB	VARCHAR (最大 LOB サイズ *2) 最大 LOB サイズが 31 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。

RDS for Oracle データタイプ	Amazon Redshift データタイプ
CLOB	VARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
NCLOB	NVARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
LONG	VARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
LONG RAW	VARCHAR (最大 LOB サイズ *2) 最大 LOB サイズが 31 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。
XMLTYPE	VARCHAR (最大 LOB サイズ) 最大 LOB サイズが 63 KB を超えることはできません。Amazon Redshift は 64 KB より大きい VARCHAR をサポートしていません。

RDS for PostgreSQL の DDL オペレーション

Amazon Redshift は PostgreSQL から派生しているため、共通の PostgreSQL アーキテクチャに基づくいくつかの機能を RDS for PostgreSQL と共有しています。ゼロ ETL 統合は、これらの類似点を活用して、RDS for PostgreSQL から Amazon Redshift へのデータレプリケーションを合理化し、データベースを名前でマッピングして、共有のデータベース、スキーマ、テーブル構造を利用します。

RDS for PostgreSQL ゼロ ETL 統合を管理するときは、以下の点を考慮してください。

- 分離はデータベースレベルで管理されます。
- レプリケーションはデータベースレベルで行われます。
- RDS for PostgreSQL データベースは Amazon Redshift データベースに名前でマッピングされるため、元の名前を変更すると、名前を変更した後の Redshift データベースにデータが流れます。

Amazon Redshift と RDS for PostgreSQL は類似していますが、重要な違いがあります。以下のセクションでは、一般的な DDL オペレーションに対する Amazon Redshift のシステムレスポンスについて概説します。

トピック

- データベースのオペレーション
- スキーマオペレーション
- テーブルの操作

データベースのオペレーション

次の表は、データベース DDL オペレーションのシステムレスポンスを示しています。

DDL オペレーション	Redshift システムレスポンス
`CREATE DATABASE`	オペレーションなし
`DROP DATABASE`	Amazon Redshift は、ターゲット Redshift データベース内のすべてのデータを削除します。
`RENAME DATABASE`	Amazon Redshift は、元のターゲットデータベース内のすべてのデータを削除し、新しいターゲットデータベース内のデータを再同期します。新しいデータベースが存在しない場合は、手動で作成する必要があります。手順については、「Amazon Redshift でのデスティネーションデータベースの作成」を参照してください。

スキーマオペレーション

次の表は、スキーマ DDL オペレーションのシステムレスポンスを示しています。

DDL オペレーション	Redshift システムレスポンス
`CREATE SCHEMA`	オペレーションなし
`DROP SCHEMA`	Amazon Redshift は元のスキーマを削除します。
`RENAME SCHEMA`	Amazon Redshift は元のスキーマを削除し、新しいスキーマ内のデータを再同期します。

テーブルの操作

次の表は、テーブル DDL オペレーションのシステムレスポンスを示しています。

DDL オペレーション	Redshift システムレスポンス
`CREATE TABLE`	Amazon Redshift はテーブルを作成します。 プライマリキーなしでテーブルを作成したり、宣言的パーティショニングを実行したりするなど、一部のオペレーションではテーブルの作成が失敗します。詳細については、「the section called “制限事項”」および「the section called “ゼロ ETL 統合のトラブルシューティング”」を参照してください。
`DROP TABLE`	Amazon Redshift はテーブルを削除します。
`TRUNCATE TABLE`	Amazon Redshift はテーブルを切り捨てます。
`ALTER TABLE (RENAME... )`	Amazon Redshift はテーブルまたは列の名前を変更します。

DDL オペレーション	Redshift システムレスポンス
ALTER TABLE (SET SCHEMA)	Amazon Redshift は元のスキーマのテーブルを削除し、新しいスキーマのテーブルを再同期します。
ALTER TABLE (ADD PRIMARY KEY)	Amazon Redshift はプライマリキーを追加し、テーブルを再同期します。
ALTER TABLE (ADD COLUMN)	Amazon Redshift はテーブルに列を追加します。
ALTER TABLE (DROP COLUMN)	Amazon Redshift は、プライマリキー列でない列を削除します。それ以外の場合は、テーブルを再同期します。
ALTER TABLE (SET LOGGED/UNLOGGED)	テーブルをログ記録するように変更すると、Amazon Redshift はテーブルを再同期します。テーブルをログ記録しないように変更すると、Amazon Redshift はテーブルを削除します。

Amazon RDS ゼロ ETL 統合の表示と監視

Amazon RDS ゼロ ETL 統合の詳細を表示して、その設定情報と現在のステータスを確認できます。Amazon Redshift の特定のシステムビューをクエリすることで、ゼロ ETL 統合をモニタリングすることもできます。さらに、Amazon Redshift は統合関連の特定のメトリクスを Amazon CloudWatch に公開します。これらは Amazon Redshift コンソールで確認できます。

トピック

- 統合の表示
- Amazon Redshift のシステムテーブルを使用して統合をモニタリングします。
- Amazon Redshift の Amazon EventBridge で統合をモニタリング

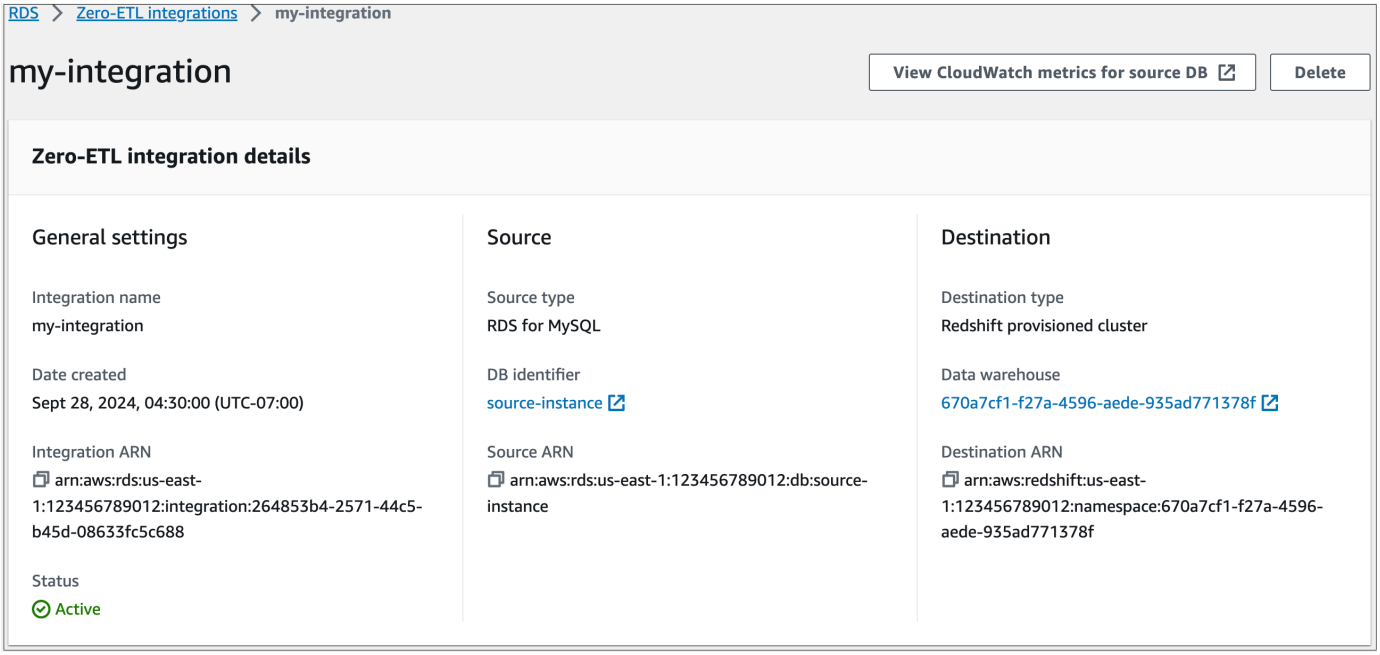
統合の表示

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、Amazon RDS ゼロ ETL 統合を表示できます。

コンソール

ゼロ ETL 統合の詳細を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 左側のナビゲーションペインから、[ゼロ ETL 統合] を選択します。
3. そのソースデータベースやターゲットのデータウェアハウスなどの詳細を表示するには、統合を選択します。



統合は、以下のようなステータスを持ちます。

- `Creating` — 統合は作成中です。
- `Active` — 統合はトランザクションデータをターゲットデータウェアハウスに送信中です。
- `Syncing` — 統合で回復可能なエラーが発生したため、データを再シードしています。影響を受けるテーブルは、再同期が完了するまでクエリを実行できません。
- `Needs attention` — 統合でイベントまたはエラーが発生したため、解決するには手動介入が必要です。問題を解決するには、統合詳細ページでエラーメッセージの指示に従ってください。

- `Failed` — 統合で、修正できない回復不能なイベントまたはエラーが発生しました。統合を手動で削除して再作成する必要があります。
- `Deleting` — 統合を削除中です。

AWS CLI

AWS CLIを使用して現在のアカウントの Zero-ETL 統合をすべて表示するには、describe-integrations コマンドを使用して `--integration-identifier` オプションを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Windows の場合:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Amazon RDS API を使用してゼロ ETL 統合を表示するには、`IntegrationIdentifier` パラメータを指定して `DescribeIntegrations` オペレーションを使用します。

Amazon Redshift のシステムテーブルを使用して統合をモニタリングします。

Amazon Redshift には、システムの動作に関する情報を含むシステムテーブルとビューがあります。これらのシステムテーブルとビューには、その他のデータベーステーブルと同じ方法でクエリを実行できます。Amazon Redshift のシステムテーブルとビューの詳細については、「Amazon Redshift データベースデベロッパーガイド」の「システムテーブルとビューのリファレンス」を参照してください。

以下のシステムビューとテーブルにクエリして、 ゼロ ETL 統合に関する情報を取得できます。

- SVV_INTEGRATION — 統合の設定の詳細を提供します。
- SVV_INTEGRATION_TABLE_STATE — 統合内の各テーブルの状態を記述します。

- SYS_INTEGRATION_TABLE_STATE_CHANGE — 統合のテーブルステート変更ログを表示します。
- SYS_INTEGRATION_ACTIVITY — 完了した統合実行に関する情報を提供します。

統合関連の CloudWatch メトリクスはすべて Amazon Redshift から生成されます。詳細については、「Amazon Redshift 管理ガイド」の「ゼロ ETL 統合のメトリクス」を参照してください。現在、Amazon RDS は統合関連のメトリクスを CloudWatch に公開していません。

Amazon Redshift の Amazon EventBridge で統合をモニタリング

Amazon Redshift は統合関連のイベントを Amazon EventBridge に送信します。イベントとそれに対応するイベント ID のリストについては、Amazon Redshift 管理ガイドの「Amazon EventBridge によるゼロ ETL 統合イベント通知」を参照してください。

Amazon RDS ゼロ ETL 統合の変更

サポートされるデータウェアハウスとのゼロ ETL 統合では、名前、説明、データフィルタリングオプションのみを変更できます。統合の暗号化に使用される AWS KMS キー、ソースデータベースまたはターゲットデータベースは変更できません。

既存の統合にデータフィルターを追加すると、Amazon RDS は、そのフィルターが本来存在していたものであるかのように再評価します。ターゲットのデータウェアハウスに現在あるデータのうち、新しいフィルタリング条件に一致しないものは削除されます。統合からデータフィルターを削除すると、以前はフィルター条件に一致していなかった (が現在は一致する) データがターゲットデータウェアハウスにレプリケートされます。詳細については、「the section called “ゼロ ETL 統合でのデータフィルタリング”」を参照してください。

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、ゼロ ETL 統合を変更できます。

RDS コンソール

ゼロ ETL 統合を変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[ゼロ ETL 統合] を選択して、変更する統合を選択します。
3. [変更] を選択して、変更可能な設定をすべて変更します。

4. すべての変更が完了したら、[変更] を選択します。

AWS CLI

AWS CLI を使用してゼロ ETL 統合を変更するには、modify-integration コマンドを呼び出します。`--integration-identifier` に加えて、次のいずれかのオプションを指定します。

- `--integration-name` — 統合の新しい名前を指定します。
- `--description` — 統合の新しい説明を指定します。
- `--data-filter` — 統合のデータフィルタリングオプションを指定します。詳細については、「the section called “ゼロ ETL 統合でのデータフィルタリング”」を参照してください。

Example

次のリクエストは既存のインテグレーションを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Windows の場合:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

RDS API

RDS API を使用してゼロ ETL 統合を変更するには、ModifyIntegration オペレーションを呼び出します。統合識別子と変更するパラメータを指定します。

AmazonRDS ゼロ ETL 統合の削除

ゼロ ETL 統合を削除すると、Amazon RDS はソースデータベースの からその ETL 統合を削除します。トランザクションデータが Amazon RDS または分析送信先から削除されることはありませんが、Amazon RDS は新しいデータを Amazon Redshift や Amazon SageMaker に送信しません。

統合は、ステータスが Active、Failed、Syncing、または Needs attention のときにのみ削除できます。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用してゼロ ETL 統合を削除できます。

コンソール

ゼロ ETL 統合を削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 左側のナビゲーションペインから、[ゼロ ETL 統合] を選択します。
3. 削除するゼロ ETL 統合を選択します。
4. [アクション]、[削除] を選択し、削除を確定します。

AWS CLI

ゼロ ETL 統合を削除するには、delete-integration コマンドを使用して `--integration-identifier` オプションを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Windows の場合:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Amazon RDS API を使用してゼロ ETL 統合を削除するには、`IntegrationIdentifier` パラメータを指定して `DeleteIntegration` オペレーションを使用します。

Amazon RDS ゼロ ETL 統合のトラブルシューティング

分析送信先の SVV_INTEGRATION システムテーブルにクエリを実行することで、ゼロ ETL 統合の状態を確認できます。`state` 列の値が `ErrorState` の場合、何か問題があることを意味します。詳細については、「the section called “システムテーブルを使ったモニタリング”」を参照してください。

以下の情報を使用して、Amazon RDS ゼロ ETL 統合に関する一般的な問題をトラブルシューティングしてください。

> **Important**
>
> 再同期および更新オペレーションは、Amazon SageMaker AI Lakehouse とのゼロ ETL 統合では使用できません。統合に問題がある場合は、統合を削除して新しい統合を作成する必要があります。既存の統合を更新または再同期することはできません。

トピック

- ゼロ ETL 統合を作成できない
- 統合が Syncing の状態でスタックしている
- テーブルが Amazon Redshift にレプリケートされない
- 1 つ以上の Amazon Redshift テーブルを再同期する必要がある
- Amazon SageMaker AI Lakehouse ゼロ ETL 統合の統合で失敗した問題

ゼロ ETL 統合を作成できない

ゼロ ETL 統合を作成できない場合は、ソースデータベースについて以下が正しいことを確認してください。

- ソースデータベースは、サポートされている DB エンジンバージョンを実行している必要があります。サポートされているバージョンのリストについては、「the section called “ゼロ ETL 統合 ”」を参照してください。
- DB パラメータを正しく設定しました。必須パラメータが正しく設定されていないか、データベースに関連付けられていない場合、作成は失敗します。「the section called “ステップ 1: カスタム DB のパラメータグループを作成する”」を参照してください。

さらに、ターゲットデータウェアハウスについて、以下が正しいことを確認してください。

- 大文字と小文字の区別が有効になっている。「データウェアハウスの大文字と小文字の区別を有効にする」を参照してください。
- 正しい承認済みプリンシパルと統合ソースを追加した。「Amazon Redshift データウェアハウスの認可を設定する」を参照してください。
- データウェアハウスは暗号化されている (プロビジョニングされたクラスターの場合)。「Amazon Redshift データベース暗号化」を参照してください。

統合が Syncing の状態でスタックしている

必須 DB パラメータのいずれかの値を変更すると、統合のステータスが常に Syncing と表示されることがあります。

この問題を解決するには、ソースデータベースに関連付けられているパラメータグループのパラメータの値をチェックして、必要な値と一致していることを確認します。詳細については、「the section called “ステップ 1: カスタム DB のパラメータグループを作成する”」を参照してください。

パラメータを変更した場合は、必ずデータベースを再起動して変更を適用してください。

テーブルが Amazon Redshift にレプリケートされない

Amazon Redshift に 1 つ以上のテーブルが反映されていない場合は、次のコマンドを実行してテーブルを再同期できます。

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

詳細については、「Amazon Redshift SQL リファレンス」の「ALTER DATABASE」を参照してください。

1 つ以上のソーステーブルにプライマリキーがないため、データがレプリケートされていない可能性があります。Amazon Redshift のモニタリングダッシュボードには、これらのテーブルのステータスが Failed と表示され、ゼロ ETL 統合全体のステータスが Needs attention に変わります。この問題を解決するには、プライマリキーとなる既存のキーをテーブル内で特定するか、合成プライマリキーを追加することができます。詳細な解決策については、「Amazon Aurora MySQL または Amazon RDS for MySQL と Amazon Redshift とのゼロ ETL 統合を作成する際に、プライマリキーがないテーブルを処理する」を参照してください。

1 つ以上の Amazon Redshift テーブルを再同期する必要がある

ソースデータベースに対して特定のコマンドを実行するには、テーブルの再同期が必要になる場合があります。このような場合、SVV_INTEGRATION_TABLE_STATE システムビューには table_state が ResyncRequired と表示されます。つまり、統合は MySQL から Amazon Redshift に特定のテーブルを完全にリロードする必要があります。

テーブルが再同期を開始すると、Syncing の状態はになります。テーブルを再同期するために手動で操作を行う必要はありません。テーブルデータの再同期中は、Amazon Redshift からデータにアクセスすることはできません。

以下に、テーブルを ResyncRequired 状態にする操作の例と、検討すべき代替案をいくつか示します。

Operation	例	代替

特定の位置への列の追加	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  INTEGER`
	`  NOT NULL first;`
		Amazon Redshift は、first または after キーワードを使用して特定の位置に列を追加することをサポートしていません。ターゲットテーブルの列の順序が重要でない場合は、より簡単なコマンドを使用してテーブルの最後に列を追加します。

Operation	例	代替
		`ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Operation	例	代替

デフォルトの CURRENT_TIMESTAMP でのタイムスタンプ列の追加	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  TIMESTAMP`
	`  NOT NULL DEFAULT CURRENT_TIMESTAMP;`
		既存のテーブル行の CURRENT_TIMESTAMP 値は RDS for MySQL によって計算されるため、テーブルデータを完全に再同期しない限り、Amazon Redshift でシミュレートすることはできません。
		
		可能であれば、テーブルが利用可能になるまでの待ち時間を避けるために、デフォルト値を 2023-01-01 00:00:15 などのリテラル定数に切り替えてください。

Operation	例	代替
1 つのコマンドで複数の列操作を実行する	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	コマンドを ADD と RENAME の 2 つの操作に分割することを検討してください。この場合、再同期は不要です。

Amazon SageMaker AI Lakehouse ゼロ ETL 統合の統合で失敗した問題

既存のゼロ ETL 統合と Amazon SageMaker AI Lakehouse で問題が発生した場合、唯一の解決策は統合を削除して新しい統合を作成することです。他の AWS サービスとは異なり、ゼロ ETL 統合は更新または再同期オペレーションをサポートしていません。

統合の問題を解決するには

1. コンソール、CLI、または API を使用して、問題のあるゼロ ETL 統合を削除します。
2. ソースデータベースとターゲットデータウェアハウスの設定が正しいことを確認します。
3. 同じ設定または更新された設定で新しいゼロ ETL 統合を作成します。

このプロセスにより、データパイプラインが完全に再初期化されます。再初期化の時間は、ソースデータベースのサイズによって異なる場合があります。

Amazon RDS for Db2

Amazon RDS は、以下のエディションおよびバージョンの IBM Db2 を実行する DB インスタンスをサポートしています。

エディション	Db2 のバージョン
Db2 Advanced Edition	v11.5.9、v12.1.4
Db2 Community Edition	v12.1.4
Db2 Standard Edition	v11.5.9、v12.1.4

マイナーバージョンのサポートの詳細については、「Amazon RDS での Db2 のバージョン」を参照してください。

DB インスタンスを作成する前に、このユーザーガイドの「Amazon RDS 環境のセットアップ」セクションの手順を完了してください。マスターユーザーを使用して DB インスタンスを作成すると、ユーザーは DBADM 権限を取得しますが、いくつかの制限があります。このユーザーは、追加のデータベースアカウントの作成などの管理タスクに使用します。SYSADM、SYSCTRL、SYSMAINT インスタンスレベルの権限、または SECADM データベースレベルの権限を使用することはできません。

以下を作成することができます。

- DB インスタンス
- DB スナップショット
- ポイントインタイムの復元
- 自動ストレージバックアップ
- 手動ストレージバックアップ

仮想プライベートクラウド (VPC)内で Db2 を実行する DB インスタンスを使用できます。また、さまざまなオプションを有効にして、Amazon RDS for Db2 DB インスタンスに機能を追加することもできます。Amazon RDS は、可用性の高いフェイルオーバーソリューションとして RDS for Db2 のマルチ AZ 配置をサポートしています。

> **Important**
>
> マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする、特定のシステムプロシージャやテーブルへのアクセスも制限しています。データベースへのアクセスには、IBM Db2 CLP などの標準の SQL クライアントを使用します。ただし、Telnet またはセキュアシェル (SSH) を使用してホストに直接アクセスすることはできません。

トピック

- Amazon RDS における Db2 の概要
- Amazon RDS for Db2 DB インスタンス作成の前提条件
- Amazon RDS for Db2 DB インスタンス上の複数のデータベース
- Db2 DB インスタンスへの接続
- Amazon RDS for Db2 DB インスタンス接続の保護
- Amazon RDS for Db2 DB インスタンスの管理
- Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合
- Amazon RDS for Db2 へのデータの移行
- Amazon RDS for Db2 フェデレーション
- Amazon RDS for Db2 のレプリカの使用
- RDS for Db2 DB インスタンスのオプション
- Amazon RDS for Db2 の外部ストアドプロシージャ
- Amazon RDS for Db2 の既知の問題と制限
- Amazon RDS for Db2 ストアドプロシージャリファレンス
- Amazon RDS for Db2 ユーザー定義関数リファレンス
- Amazon RDS for Db2 のトラブルシューティング

Amazon RDS における Db2 の概要

次のセクションを読むと、Amazon RDS における Db2 の概要を確認できます。

トピック

- Amazon RDS for Db2 の機能

- Amazon RDS での Db2 のバージョン
- Amazon RDS for Db2 のライセンスオプション
- Amazon RDS for Db2 インスタンスクラス
- Amazon RDS for Db2 のデフォルトロール
- Amazon RDS for Db2 パラメータ
- Amazon RDS 上の Db2 データベースの EBCDIC 照合
- Amazon RDS for Db2 DB インスタンスのローカルタイムゾーン

Amazon RDS for Db2 の機能

Amazon RDS for Db2 は、IBM Db2 データベースのほとんどの特徴と機能をサポートしています。一部の機能には、制限付きのサポートまたは制限された特権があります。特定の Db2 バージョンの Db2 データベース機能の詳細については、「IBM Db2 ドキュメント」を参照してください。

[What's New with Database?] (データベースの新機能) ページで新しい Amazon RDS 機能をフィルタリングできます。[製品] で [Amazon RDS] を選択します。その後、**`Db2 2023`** などのキーワードを使用して検索できます。

> **Note**
>
> 以下のリストは完全なものではありません。

トピック

- RDS for Db2 でサポートされる機能
- RDS for Db2 でサポートされていない機能

RDS for Db2 でサポートされる機能

RDS for Db2 は、IBM Db2 のネイティブな機能と Amazon RDS のコアとなる機能を含む機能をサポートしています。

IBM Db2 にネイティブな機能

RDS for Db2 では、次の Db2 データベース機能をサポートしています。

- お客様が定義したコードセット、照合順序、ページサイズ、および地域を使用する標準データベースの作成。Amazon RDS rdsadmin.create_database ストアドプロシージャを使用します。
- ローカルユーザーおよびグループの追加、削除、または変更。RDS for Db2 の権限の付与と取り消しに関するストアドプロシージャ のための Amazon RDS ストアドプロシージャを使用します。
- Amazon RDS rdsadmin.create_role ストアドプロシージャを使用したロールの作成。
- 標準の行で構成されたテーブルのサポート。
- 列で構成されたテーブルの分析ワークロードのサポート。
- Oracle や MySQL などの Db2 の互換性機能を定義する機能。
- Java ベースの外部ストアドプロシージャのサポート。
- SSL/TLS を使用した転送中のデータ暗号化のサポート。
- データベースのステータスのモニタリング (`ALIVE`、`DOWN`、`STORAGE_FULL`、`UNKNOWN`、`STANDBY_CONNECTABLE`)。
- お客様が提供するオフラインまたはオンライン Linux (LE) データベースの復元。RDS for Db2 のデータベースのストアドプロシージャ のための Amazon RDS ストアドプロシージャを使用します。
- データベースとセルフマネージド Db2 データベースの同期を維持するための、お客様が提供する Db2 アーカイブログの適用。RDS for Db2 のデータベースのストアドプロシージャ のための Amazon RDS ストアドプロシージャを使用します。
- Db2 インスタンスレベルおよびデータベースレベルの監査のサポート。
- 同種フェデレーションのサポート。
- Amazon Simple Storage Service (Amazon S3) のデータファイルからテーブルをロードする機能。
- `CONNECT`、`SYSMON`、`ACCESSCTRL`、`DATAACCESS`、`SQLADM`、`WLMADM`、`EXPLAIN`、`LOAD`、または `IMPLICIT_SCHEMA` などのユーザー、グループ、またはロールに付与されるアクセス承認。
- 複数のデータベースの作成。

> **Note**
>
> RDS for Db2 DB インスタンスには、最大 50 個のデータベースを含めることができます。詳細については、「the section called “複数の Db2 データベース”」を参照してください。

Amazon RDS のコアとなる機能

RDS for Db2 は、次の Amazon RDS のコア機能をサポートしています。

- DB インスタンスに割り当てるカスタムパラメータグループ
- DB インスタンスの作成、変更、削除
- セルフマネージド Db2 オフラインまたはオンライン Linux (LE) データベースバックアップの復元

> **Note**
>
> バックアップを復元するには、DB インスタンスの作成時にデータベースの名前を指定しないでください。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

- gp3 ストレージタイプ、io2 ストレージタイプ、io1 ストレージタイプのサポート。
- Kerberos 認証での AWS Managed Microsoft AD の使用と、RDS for Db2 での LDAP グループ認可の使用。
- 既存の Db2 インスタンスのセキュリティグループ、ポート、インスタンスタイプ、ストレージ、バックアップ保持期間、その他の設定の変更
- DB インスタンスの削除保護
- 暗号化されたバックアップを含む、クロスリージョンポイントインタイムリカバリ (PITR)。
- ストレージの暗号化と保管時の暗号化に対する AWS Key Management Service (AWS KMS) の使用
- 1 つのスタンバイによって高可用性を実現するマルチ AZ DB インスタンス
- DB インスタンスの再起動
- マスターパスワードの更新
- 特定の時点への DB インスタンスの復元
- ストレージレベルのバックアップを使用した DB インスタンスのバックアップと復元。
- インスタンスの開始と停止
- DB インスタンスのメンテナンス
- 同一リージョンおよびクロスリージョンのスタンバイおよびリードレプリカ

RDS for Db2 でサポートされていない機能

RDS for Db2 では、次の Db2 データベース機能はサポートしていません。

- マスターユーザーの `SYSADM`、`SECADM`、`SYSMAINT` アクセス。
- C、C++、または Cobol で記述された外部ストアドプロシージャ。

- 単一ホスト上の複数の Db2 DB インスタンス。
- 認証用の外部 GSS-API プラグイン。
- Db2 データベースをバックアップまたは復元するための外部サードパーティープラグイン。
- IBM Db2 Warehouse などのマルチノードの超並列処理 (MPP)。
- IBM Db2 pureScale.
- RDS for Db2 の高可用性ディザスタリカバリ (HADR) の手動セットアップ。

> **Note**
>
> Amazon RDS は、レプリカを通じて RDS for Db2 の HADR をサポートおよび管理します。詳細については、「Amazon RDS for Db2 のレプリカの使用」を参照してください。RDS for Db2 は、マルチ AZ 配置、クロスリージョン自動バックアップ、およびレプリケーションをサポートしています。詳細については、「Amazon RDS のマルチ AZ DB インスタンスデプロイ」および「別の AWS リージョン への自動バックアップのレプリケーション」を参照してください。

- ネイティブデータベースの暗号化。
- Informix、Sybase、および Teradata への異種フェデレーション。詳細については、「the section called “フェデレーション”」を参照してください。
- データをバックアップおよび復元することで、フェンスされていないルーチンの作成と既存のフェンスされていないルーチンの移行。詳細については、「フェンスされていないルーチン」を参照してください。
- 新しい非自動ストレージテーブルスペースの作成。詳細については、「移行中の非自動ストレージテーブルスペース」を参照してください。
- 外部テーブル。

Amazon RDS での Db2 のバージョン

Db2 の場合、バージョン番号は major.minor.build.revision という形式になります。例えば、11.5.9.0.sb00000000.r1 です。バージョンの実装は Db2 の実装と一致します。

major

メジャーバージョンの番号は、11.5 など、バージョン番号の整数と 1 つ目の小数の部分の両方です。メジャーバージョン番号が変更された場合 (11.5 から 12.1 へなど)、そのバージョン変更はメジャーと考えます。

minor

マイナーバージョンの番号は、バージョン番号の 3 番目と 4 番目の部分です。例えば、バージョン番号が 11.5.9.0 の場合、9.0 です。9.0 の 9 など、3 番目の部分は、Db2 modpack を示します。9.0 の 0 など、4 番目の部分は、Db2 fixpack を示します。Db2 modpack または Db2 fixpack のいずれかが変更された場合 (バージョン 11.5.9.0 から 11.5.9.1、または 11.5.9.0 から 11.5.10.0 に変更された場合など)、バージョンの変更はマイナーと見なされますが、カタログテーブルの更新を行う必要があります (Amazon RDS はこのような例外を処理します)。

ビルド

ビルド番号はバージョン番号の 5 番目の部分です。例えば、バージョン番号が 11.5.9.0.sb00000000 の場合、sb00000000 です。数値部分がすべてゼロであるビルド番号は、標準ビルドであることを示します。数値部分の一部がゼロではないビルド番号は、特別なビルドであることを示します。既存の Db2 バージョンのセキュリティ修正ビルドまたは特別なビルドがある場合、ビルド番号が変更されます。ビルド番号が変更された場合、Amazon RDS が新しいマイナーバージョンを自動的に適用したことも示しています。

revision

リビジョン番号は、バージョン番号の 6 番目の部分です。例えば、バージョン番号が 11.5.9.0.sb00000000.r1 場合、r1 です。リビジョンは、既存の Db2 リリースに対する Amazon RDS リビジョンです。リビジョン番号が変更された場合、Amazon RDS が新しいマイナーバージョンを自動的に適用したことを示しています。

トピック

- Amazon RDS Db2 インスタンスのアップグレード管理
- Amazon RDS でサポートされている Db2 のメジャーバージョン

Amazon RDS Db2 インスタンスのアップグレード管理

RDS でサポートされている Db2 マイナーバージョンの現在のリストを表示するには、次のいずれかのコマンドを使用します。

```
aws rds describe-db-engine-versions --engine db2-ae
aws rds describe-db-engine-versions --engine db2-ce
aws rds describe-db-engine-versions --engine db2-se
```

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

新しい DB インスタンスを作成するときは、現在サポートされているいずれかの Db2 バージョンを指定できます。メジャーバージョン (Db2 11.5 など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいずれかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合、Amazon RDS では、指定されたメジャーバージョンの最新リリースにデフォルトで設定されます。サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示するには、describe-db-engine-versions AWS Command Line Interface (AWS CLI) コマンドを使用します。

例えば、Amazon RDS for Db2 のサポートされているエンジンバージョンを一覧表示するには、次の AWS CLI コマンドを実行します。*#####*をお客様の AWS リージョンに置き換えてください。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
    --filters Name=engine,Values=db2-ae,db2-ce,db2-se \
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" \
    --region region
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
    --filters Name=engine,Values=db2-ae,db2-ce,db2-se ^
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" ^
    --region region
```

このコマンドでは、次の例のような出力が生成されます。

```
[
    {
    "Engine": "db2-ae",
    "EngineVersion": "11.5.9.0.sb00000000.r1",
    "DBParameterGroupFamily": "db2-ae-11.5"
```

```
    },
    {
    "Engine": "db2-ce",
    "EngineVersion": "12.1.4.0.sb00080714.r1",
    "DBParameterGroupFamily": "db2-ce-12.1"
    },
    {
    "Engine": "db2-se",
    "EngineVersion": "11.5.9.0.sb00000000.r1",
    "DBParameterGroupFamily": "db2-se-11.5"
    }

]
```

デフォルトの Db2 バージョンは、AWS リージョンによって異なる場合があります。特定のマイナーバージョンで DB インスタンスを作成するには、DB インスタンスの作成中にマイナーバージョンを指定します。 describe-db-engine-versions コマンドを実行すると、AWS リージョンにおける db2-ae、db2-ce、および db2-se データベースエンジンのデフォルトバージョンを確認できます。次の例では、米国東部 (バージニア北部) の db2-ae のデフォルトバージョンを返します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
    --default-only --engine db2-ae \
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" \
    --region us-east-1
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
    --default-only --engine db2-ae ^
    --query "DBEngineVersions[].{Engine:Engine, EngineVersion:EngineVersion,
 DBParameterGroupFamily:DBParameterGroupFamily}" ^
    --region us-east-1
```

このコマンドでは、次の例のような出力が生成されます。

```
[
    {
    "Engine": "db2-ae",
```

```
    "EngineVersion": "12.1.4.0.sb00080714.r1",
    "DBParameterGroupFamily": "db2-ae-12.1"
    }
]
```

マイナーバージョンの自動アップグレードを有効にすると、Amazon RDS によって、Amazon RDS でサポートされる新しい Db2 マイナーバージョンに DB インスタンスが自動的にアップグレードされます。このパッチ適用は、予定されたメンテナンスウィンドウ内で行われます。DB インスタンスを変更して、マイナーバージョンの自動アップグレードを有効または無効にすることができます。

スケジュールされた自動アップグレードを解除している場合は、サポートされているマイナーバージョンリリースに手動でアップグレードできます。その手順はメジャーバージョンの更新の場合と同じです。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

Amazon RDS でサポートされている Db2 のメジャーバージョン

RDS for Db2 メジャーバージョンは、少なくとも対応する IBM バージョンの IBM サポート (基本サポート) 終了日までの標準サポートの対象となります。次の表に、テストおよびアップグレードのサイクルを計画するのに使用できる日付を示します。Amazon は、RDS for Db2 バージョンのサポートを当初発表よりも長く延長した場合、新しい日付を反映してこの表を更新するようにします。

次の日付を参考にすると、テストおよびアップグレードのサイクルを計画することができます。

> **Note**
>
> 月と年のみの日付はおおよその日付であり、確定後に正確な日付で更新されます。
> describe-db-major-engine-versions AWS CLI コマンドを実行する
> か、DescribeDBMajorEngineVersions RDS API オペレーションを使用して、Db2 データベースのメジャーバージョンを表示できます。

Db2 のメジャーバージョン	IBM でのリリース日	RDS リリース日	IBM サポート終了 (Standard Edition および Advanced Edition)	IBM サポート (延長サポート) 終了日	
Db2 11.5	2019 年 6 月 27 日	2023 年 11 月 27 日	2027 年 4 月 30 日	2031 年 4 月 30 日	
Db2 12.1	2024 年 11 月 14 日	2026 年 5 月 29 日	TBD	TBD	

> **Note**
>
> API またはコンソールを使用したメジャーバージョンアップグレードは、現在サポートされていません。この機能は、近い将来のリリースで利用可能になります。
> RDS Db2 v11.5.9 から RDS Db2 v12.1.4 へのメジャーバージョンアップグレードを手動で実行するには、rdsadmin.backup_database ストアドプロシージャを使用してフル OFFLINE バックアップを実行し、rdsadmin.restore_database ストアドプロシージャを使用して既存の RDS for Db2 v12.1.4 インスタンスにそれを復元します。

Amazon RDS for Db2 のライセンスオプション

Amazon RDS for Db2 には、Bring-Your-Own-License (BYOL) と AWS Marketplace を介した Db2 ライセンスという 2 つのライセンスオプションがあります。

トピック

- Db2 の Bring-Your-Own-License (BYOL)
- AWS Marketplace 経由の Db2 ライセンス
- Db2 ライセンスの切り替え

Db2 の Bring-Your-Own-License (BYOL)

BYOL モデルでは、既存の Db2 データベースのライセンスを使用して Amazon RDS でデータベースをデプロイできます。実行する DB インスタンスクラスと Db2 データベースエディションに適切な Db2 データベースライセンスがあることを確認します。また、クラウドコンピューティング環境での IBM データベースソフトウェアのライセンス化に関する IBM のポリシーに従う必要があります。

> **Note**
>
> Db2 Community Edition (Db2 CE) v12.1 ライセンスは IBM から追加料金なしで入手できます。Amazon RDS インフラストラクチャリソースに対してのみ料金が発生します。Db2 CE インスタンスを作成する前に、IBM ウェブサイトから IBM Customer ID と IBM Site ID を取得する必要があります。

このモデルでは、アクティブな IBM サポートアカウントを継続して使用できます。Db2 データベースのサービスリクエストについては、直接 IBM にお問い合わせください。ケースサポート付きの サポート アカウントをお持ちであれば、Amazon RDS の問題については サポート にお問い合わせいただけます。Amazon Web Services と IBM では、双方の組織からのサポートが必要なケースのために、マルチベンダーサポートプロセスを用意しています。

Amazon RDS は、Db2 Advanced Edition、Db2 Community Edition、および Db2 Standard Edition の BYOL モデルをサポートしています。

> **Note**
>
> Db2 データベースがインストールされているが実行されていないため、マルチ AZ DB インスタンスはコールドスタンバイです。スタンバイでは、リクエストの読み取り、実行、処理を行うことはできません。詳細については、IBM ウェブサイトの IBM Db2 ライセンス情報を参照してください。

トピック

- Db2 の Bring-Your-Own-License (BYOL) の IBM ID
- RDS for Db2 DB インスタンスのパラメータグループに IBM ID を追加する
- AWS License Manager との統合

Db2 の Bring-Your-Own-License (BYOL) の IBM ID

BYOL モデルでは、RDS for Db2 DB インスタンスを作成、変更、または復元するには、IBM Customer ID と IBM Site ID が必要です。

RDS for Db2 DB インスタンスを作成する前に、IBM Customer ID と IBM Site ID を使用してカスタムパラメータグループを作成する必要があります。詳細については、「RDS for Db2 DB インスタンスのパラメータグループに IBM ID を追加する」を参照してください。複数の RDS for Db2 DB インスタンスは、同じ AWS アカウント または AWS リージョン で異なる IBM Customer IDs と IBM Site IDs を使用して実行できます。

> **Important**
>
> お客様の IBM Customer ID と IBM Site ID でライセンスを検証できない場合、未検証のライセンスで実行されているすべての DB インスタンスは終了される場合があります。

IBM Db2 の新規のお客様は、まず IBM から Db2 ソフトウェアライセンスを購入する必要があります。Db2 ソフトウェアライセンスの購入後、IBM Customer ID と IBM Site ID が記載されたライセンス証書が IBM から届きます。

IBM Db2 の既存のお客様は、IBM のライセンス証書で IBM Customer ID と IBM Site ID を確認できます。

また、IBM Passport Advantage Online アカウントで IBM Customer ID と IBM Site ID を探すこともできます。ログイン後、メインページまたはソフトウェアのダウンロードページで両方の IDs を表示します。

RDS for Db2 DB インスタンスのパラメータグループに IBM ID を追加する

デフォルトのパラメータグループを変更することはできないため、カスタムパラメータグループを作成し、IBM Customer ID と IBM Site ID の値を含めるように変更する必要があります。パラメータグループの詳細については、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。

> **Important**
>
> RDS for Db2 DB インスタンスを作成する前に、IBM Customer ID と IBM Site ID を使用してカスタムパラメータグループを作成する必要があります。

次の表のパラメータ設定を使用します。

パラメータ	値
rds.ibm_customer_id	<your IBM Customer ID>
rds.ibm_site_id	<your IBM Site ID>
ApplyMethod	immediate , pending-reboot

これらのパラメータは動的です。つまり、これらのパラメータが変更されるとすぐに有効になるため、DB インスタンスを再起動する必要はありません。変更をすぐに有効にしたくない場合は、ApplyMethod を pending-reboot に設定し、メンテナンスウィンドウ中にこれらの変更が行われるようにスケジュールできます。

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、カスタムパラメータグループを作成および変更することができます。

コンソール

IBM Customer ID と IBM Site ID をパラメータグループに追加するには

1. 新しい DB パラメータグループを作成します。DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. 先ほど作成したパラメータグループを変更します。パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

AWS CLI

IBM Customer ID と IBM Site ID をパラメータグループに追加するには

1. create-db-parameter-group コマンドを実行して、カスタムパラメータグループを作成します。

   次の必須パラメータを含めます。

   - --db-parameter-group-name - 作成するパラメータグループの名前。
   - --db-parameter-group-family - Db2 エンジンエディションとメジャーバージョン。有効な値:db2-se-11.5 、db2-ae-11.5、db2-ae-12.1、db2-ce-12.1、db2-se-12.1

- `--description` - このパラメータグループの説明。

DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。

2. modify-db-parameter-group コマンドを実行して、作成したカスタムパラメータグループのパラメータを変更します。

次の必須パラメータを含めます。

- `--db-parameter-group-name` – 作成したパラメータグループの名前。
- `--parameters` – パラメータ名、値、およびパラメータ更新用のアプリケーションメソッドの配列。

パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

RDS API

IBM Customer ID と IBM Site ID をパラメータグループに追加するには

1. DB パラメータグループを作成するには、Amazon RDS API CreateDBParameterGroup オペレーションを使用します。

以下の必須パラメータを含めます。

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。

2. RDS API ModifyDBParameterGroup オペレーションを使用して、作成したカスタムパラメータグループのパラメータを変更します。

以下の必須パラメータを含めます。

- `DBParameterGroupName`

- `Parameters`

  パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

これで、DB インスタンスを作成し、カスタムパラメータグループを DB インスタンスにアタッチする準備が整いました。詳細については、「Amazon RDS DB インスタンスの作成」および「Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける」を参照してください。

AWS License Manager との統合

BYOL モデルで RDS の Db2 ライセンス使用状況のモニタリングを簡単に行えるよう、AWS License Manager は RDS for Db2 と統合されています。License Manager は、仮想 CPU (vCPU) に基づいた RDS for Db2 エンジンエディションの追跡をサポートしています。また AWS Organizations で License Manager を使用して、すべての組織アカウントを一元管理することもできます。

RDS for Db2 DB インスタンスのライセンス使用状況を追跡するには、セルフマネージドライセンスを作成する必要があります。セルフマネージドライセンスを作成するには、AWS マネジメントコンソール、AWS License Manager CLI、および AWS License Manager API を使用します。または、CloudFormation および Terraform テンプレートを使用して、セルフマネージドライセンスの作成を自動化することもできます。

製品情報フィルターに一致する RDS for Db2 リソースは、セルフマネージドライセンスに自動的に関連付けられます。RDS for Db2 DB インスタンスの検出には最長で 24 時間かかる場合があります。

次の表は、RDS for Db2 のエンジンエディション製品情報フィルターとして使用可能な値を示しています。

値	説明
`db2-ae`	Db2 Advanced Edition
`db2-ce`	Db2 Community Edition
`db2-se`	Db2 Standard Edition

トピック

- 用語
- AWS License Manager でのセルフマネージドライセンスの作成
- AWS License Manager テンプレートを使用したセルフマネージドライセンス作成の自動化
- セルフマネージドライセンスを作成するための設定

用語

このページでは、Amazon RDS と AWS License Manager との統合について説明する際に、以下の用語を使用します。

セルフマネージドライセンス

> セルフマネージドライセンスは、AWS License Manager で使用される用語です。Amazon RDS コンソールは、ライセンスを AWS License Manager 設定として参照します。セルフマネージドライセンスにはエンタープライズ契約の条項に基づくライセンスルールが含まれています。作成したルールは、ライセンスを消費するコマンドのAWS処理方法を決定します。セルフマネージドライセンスを作成する際には、組織のコンプライアンスチームと密接に協力して、お使いのエンタープライズ契約を確認してください。詳細については、「License Manager のセルフマネージドライセンス」を参照してください。

AWS License Manager でのセルフマネージドライセンスの作成

セルフマネージドライセンスを作成するには、AWS マネジメントコンソール、AWS License Manager CLI、および AWS License Manager API を使用します。

> **Note**
>
> AWS マネジメントコンソール を使用して RDS for Db2 DB インスタンスを作成する場合は、ライセンスの名前を入力してセルフマネージドライセンスを作成します。次に、Amazon RDS は DB インスタンスをこのライセンスに関連付けます。(Amazon RDS コンソールでは、このライセンスは AWS License Manager 設定と呼ばれます。) AWS License Manager CLI または AWS License Manager API を使用して RDS for Db2 DB インスタンスを作成する場合は、まず次の手順でセルフマネージドライセンスを作成する必要があります。同じ状況は、RDS for Db2 DB インスタンスを特定の時点またはスナップショットから復元する場合にも当てはまります。

コンソール

RDS for Db2 DB インスタンスのライセンス使用状況を追跡するセルフマネージドライセンスを作成するには

1. https://console.aws.amazon.com/license-manager/ に移動します。
2. セルフマネージドライセンスを作成する

   手順については、「AWS License Manager ユーザーガイド」の「セルフマネージドライセンスを作成する」を参照してください。

   [Product Information (製品情報)] パネルで [RDS Product Information Filter (RDS 製品情報フィルター)] のルールを追加します。

   詳細については、AWS License Manager API リファレンスの「ProductInformation」を参照してください。

AWS License Manager CLI

> **Note**
>
> この手順では AWS License Manager CLI コマンドを使用します。

AWS CLI を使用してセルフマネージドライセンスを作成するには、AWS License Manager create-license-configuration コマンドを実行します。`--cli-input-json` または `--cli-input-yaml` のオプションを使用して、コマンドにオプションを渡します。

詳細については、「the section called “セルフマネージドライセンスを作成するための設定”」を参照してください。

次のコマンドは、Db2 Standard Edition のセルフマネージドライセンスを作成します。

```
aws license-manager create-license-configuration --cli-input-json file://rds-db2-se.json
```

次の JSON は、前のコマンドで使用された `rds-db2-se.json` ファイルの内容です。

```
{
```

```
    "Name": "rds-db2-se",
    "Description": "RDS Db2 Standard Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["db2-se"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

製品情報の詳細については、AWS License Manager ユーザーガイドの「リソースインベントリの自動検出」を参照してください。

`--cli-input` パラメータの詳細については、AWS CLI ユーザーガイドの「JSON または YAML 入力ファイルからの AWS CLI スケルトンと入力パラメータの生成」を参照してください。

AWS License Manager API

> **Note**
>
> この手順では AWS License Manager API コマンドを使用します。

セルフマネージドライセンスを作成するには、次の必須パラメータで `CreateLicenseConfiguration` AWS License Manager API オペレーションを使用します。

- `Name`
- `LicenseCountingType`
- `ProductInformationList`
- `ResourceType`
- `ProductInformationFilterList`
- `ProductInformationFilterName`

- `ProductInformationFilterValue`
- `ProductInformationFilterComparator`

パラメータの詳細については、「」を参照してくださいthe section called “セルフマネージドライセンスを作成するための設定”

AWS License Manager テンプレートを使用したセルフマネージドライセンス作成の自動化

CloudFormation および Terraform テンプレートを使用して、セルフマネージドライセンスの作成を自動化できます。

次のサンプルテンプレート CloudFormation は、RDS for Db2 で Db2 Standard Edition のセルフマネージドライセンスを作成します。Db2 Advanced Edition のテンプレートの場合は、`Name`、`Description`、および `ProductInformationFilter` の値を更新します。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: CloudFormation template to create a License Configuration for Db2 Standard
 Edition on RDS for Db2.

Resources:
  Db2LicenseConfiguration:
    Type: "AWS::LicenseManager::LicenseConfiguration"
    Properties:
      Name: "rds-db2-se"
      Description: "Db2 Standard Edition on RDS for Db2"
      LicenseCountingType: "vCPU"
      LicenseCountHardLimit: false
      ProductInformationList:
        - ResourceType: "RDS"
          ProductInformationFilterList:
            - ProductInformationFilterName: "Engine Edition"
              ProductInformationFilterValue:
                - "db2-se"
              ProductInformationFilterComparator: "EQUALS"
```

Amazon RDS での CloudFormation の使用に関する詳細は、「AWS CloudFormation での Amazon RDS リソースの作成」を参照してください。

次のサンプルテンプレート Terraform は、RDS for Db2 で Db2 Standard Edition のセルフマネージドライセンスを作成します。*us-east-1* を、ご利用の AWS リージョン に置き換えて

ください。Db2 Advanced Edition のテンプレートの場合は、name、description、および product_information_filter の値を更新します。

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_licensemanager_license_configuration" "rds_db2_license_config" {
  name                     = "rds-db2-se"
  description              = "Db2 Standard Edition on RDS for Db2
  license_counting_type    = "vCPU"
  license_count_hard_limit = false

  product_information_list {
    resource_type = "RDS"

    product_information_filter {
      name       = "Engine Edition"
      comparator = "EQUALS"
      value      = ["db2-se"]
    }
  }
}
```

Terraform と Amazon RDS の使用の詳細については、「AWS 規範ガイダンス」の「Using Terraform as an IaC tool for the AWS クラウド」および「Best practices for using the Terraform AWS Provider」を参照してください。

セルフマネージドライセンスを作成するための設定

次の表では、AWS License Manager CLI、AWS License Manager API、CloudFormation テンプレート、および Terraform テンプレートを使用してセルフマネージドライセンスを作成する際の設定の情報を示します。次の表のパラメータ名は、AWS License Manager API と CloudFormation テンプレートで使用される名前の形式で表示されます。

パラメータ名	データ型	必要	説明
Name	string	はい	ライセンス設定の名前。

パラメータ名	データ型	必要	説明
説明	string	いいえ	ライセンス設定の説明。
LicenseCountingType	string	はい	ライセンスインベントリの追跡に使用されるディメンション。有効な値: vCPU。
LicenseCountHardLimit	boolean	いいえ	ハードライセンス適用とソフトライセンス適用のどちらを使用するかを示します。ハード制限を超えると、新しいインスタンスの起動がブロックされます。
ProductInformationList	オブジェクトの配列	はい	ライセンス設定の製品情報のリスト。
ResourceType	string	はい	リソースタイプ。有効な値: RDS。
ProductInformationFilterList	オブジェクトの配列	はい	ライセンス設定の製品情報フィルターのリスト。
ProductInformationFilterName	string	はい	宣言されるフィルターのタイプの名前。有効な値: Engine Edition。

パラメータ名	データ型	必要	説明
ProductInformation FilterValue	文字列の配列	はい	フィルターをオンにする値。設定できる値は 1 つだけです。有効な値: db2-ae、db2-ce、または db2-se
ProductInformation FilterComparator	string	はい	ProductInformation FilterName の論理演算子。有効な値: EQUALS。

AWS Marketplace 経由の Db2 ライセンス

AWS Marketplace モデル経由の Db2 ライセンスでは、Db2 ライセンスのサブスクリプション料金が時間単位で請求されます。このモデルは、ライセンスを購入することなく、RDS for Db2 をすぐに使い始めることができます。

AWS Marketplace 経由で Db2 ライセンスを使用するには、使用する IBM Db2 エディションの AWS Marketplace サブスクリプションを有効にする必要があります。まだお持ちでない場合は、IBM Db2 エディションのサブスクリプションを AWS Marketplace で購入する必要があります。

Amazon RDS は、IBM Db2 Standard Edition と IBM Db2 Advanced Edition の AWS Marketplace 経由 Db2 ライセンスをサポートしています。

トピック

- 用語
- 支払いと請求
- Db2 Marketplace サブスクリプションの購入と IBM での登録
- プライベートオファーを受ける方法について

用語

このページでは、Amazon RDS と AWS Marketplace との統合について説明する際に、以下の用語を使用します。

SaaS サブスクリプション

AWS Marketplace では、従量制料金のライセンスモデル Software-as-a-Service (SaaS) 製品は、利用ベースのサブスクリプションモデルを採用しています。Db2 の販売者である IBM が使用量を確認し、ユーザーは使用した分の料金だけを支払います。

パブリックオファー

パブリックオファーでは、ユーザーは AWS Marketplace 製品を AWS マネジメントコンソールから直接購入できます。

プライベートオファー

プライベートオファーとは、販売者と購入者が AWS Marketplace での購入のカスタム料金とエンドユーザー使用許諾契約 (EULA) の条件を交渉できる購入プログラムのことです。

Db2 Marketplace 料金

IBM が請求する Db2 ソフトウェアライセンスの使用料金。これらのサービス料金は AWS Marketplace で計算され、AWS 請求書の AWS Marketplace セクションに記載されます。

Amazon RDS 料金

RDS for Db2 サービスに対して AWS が請求する料金。AWS Marketplace での Db2 ライセンスは除きます。料金は、使用された Amazon RDS に基づいて計算され、AWS 請求書に記載されます。

支払いと請求

RDS for Db2 は AWS Marketplace と統合され、Db2 の時間単位の従量制料金ライセンスを提供します。Db2 Marketplace 料金には Db2 ソフトウェアのライセンス費用が含まれ、Amazon RDS 料金には RDS for Db2 DB インスタンスの使用料金が含まれます。料金の詳細については、「Amazon RDS for Db2 の料金」を参照してください。

これらの料金の請求を停止するには、RDS for Db2 DB インスタンスをすべて削除する必要があります。また、AWS Marketplace for Db2 ライセンスのサブスクリプションを削除することもできます。DB インスタンスを削除せずにサブスクリプションを削除した場合、Amazon RDS は DB インス

タンスの使用に対して引き続き課金します。詳細については、「the section called “DB インスタンスを削除する”」を参照してください。

AWS Marketplace 経由の Db2 ライセンスを使用する RDS for Db2 DB インスタンスの料金の表示と管理は、AWS Billing コンソールで行うことができます。請求書には、AWS Marketplace 経由の Db2 ライセンスの使用料と Amazon RDS の使用料の 2 つの料金が含まれます。詳細については、「AWS Billing and Cost Management ユーザーガイド」の「Viewing your bill」を参照してください。

Db2 Marketplace サブスクリプションの購入と IBM での登録

AWS Marketplace 経由の Db2 ライセンスを使用するには、AWS マネジメントコンソール を使用して以下の 2 つのタスクを完了する必要があります。AWS CLI または RDS API を使用してこれらのタスクを完了することはできません。

> **Note**
>
> AWS CLI または RDS API を使用して DB インスタンスを作成する場合は、まずこれら 2 つのタスクを完了する必要があります。

> **Note**
>
> Marketplace ライセンスは Db2 Community Edition (db2-ce) では使用できません。

トピック

- タスク 1: AWS Marketplace で Db2 サブスクリプションを購入する
- タスク 2: サブスクリプションを IBM に登録する

タスク 1: AWS Marketplace で Db2 サブスクリプションを購入する

AWS Marketplace で Db2 ライセンスを使用するには、Db2 の有効な AWS Marketplace サブスクリプションが必要です。サブスクリプションは特定の IBM Db2 エディションに関連付けられているため、使用する Db2 のエディションごと (IBM Db2 Advanced Edition 、IBM Db2Standard Edition) に AWS Marketplace で Db2 のサブスクリプションを購入する必要があります。AWS Marketplace サブスクリプションの詳細については、「AWS Marketplace 購入者ガイド」の「SaaS 製品の料金モデル」を参照してください。

DB インスタンスの作成を開始する前に、AWS Marketplace で Db2 のサブスクリプションを購入することをお勧めします。

タスク 2: サブスクリプションを IBM に登録する

AWS Marketplace で Db2 のサブスクリプションを購入したら、選択した Db2 サブスクリプションタイプの AWS Marketplace ページから IBM 注文の登録を完了します。AWS Marketplace ページで、[購入オプションの表示] を選択し、[アカウントの設定] を選択します。既存の IBM アカウントに登録することも、無料の IBM アカウントを作成することもできます。

プライベートオファーを受ける方法について

Db2 の AWS Marketplace プライベートオファーを IBM にリクエストできます。詳細については、「AWS Marketplace 購入者ガイド」の「プライベートオファー」を参照してください。

> **Note**
>
> AWS Organizations ユーザーであり、支払いアカウントとメンバーアカウントに発行されたプライベートオファーを受け取った場合は、以下の手順に従って、組織内の各アカウントで直接、Db2 にサブスクライブしてください。

Db2 プライベートオファーを受けるには

1. プライベートオファーが発行されたら、AWS Marketplace コンソールにサインインします。
2. Db2 プライベートオファーのリンクが記載されたメールを開きます。
3. リンクに従い、プライベートオファーに直接アクセスします。

   > **Note**
   >
   > 正しいアカウントにログインする前にこのリンクをクリックすると、[ページが見つかりません] (404) というエラーが表示されます。

4. 利用規約を読みます。
5. 次に [規約の受諾] を選択します。

> **Note**
>
> AWS Marketplace プライベートオファーが受け入れられない場合、AWS Marketplace からの Db2 サービス料は引き続きパブリック時間料金で請求されます。

6. オファーの詳細を確認するには、製品リストの [詳細を表示] を選択します。

手順を完了したら、「the section called “DB インスタンスの作成”」の手順に従って、DB インスタンスを作成できます。[ライセンス] の [AWS マネジメントコンソール] で、[AWS Marketplace から] が選択されていることを確認します。

Db2 ライセンスの切り替え

RDS for Db2 では、Db2 ライセンスを切り替えることができます。例えば、Bring-Your-Own-License (BYOL) から始めて、後で AWS Marketplace 経由の Db2 ライセンスに切り替えることができます。

> **Important**
>
> AWS Marketplace 経由の Db2 ライセンスに切り替える場合は、使用する IBM Db2 エディションの有効な AWS Marketplace サブスクリプションがあることを確認してください。有効なサブスクリプションがない場合は、まず Db2 エディション用に AWS Marketplace で Db2 のサブスクリプションを購入してから、復元操作を行います。

コンソール

Db2 ライセンスを切り替えるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Automated backups] (自動バックアップ) を選択します。

   [Current Region] (現在のリージョン) タブに自動バックアップが表示されます。
3. 復元する DB インスタンスを選択します。
4. 「アクション」 で、「特定時点への復元」 を選択します。

   [特定時点への復元] ウィンドウが表示されます。

5. 「Latest restorable time」 を選択してできるだけ最新の時点に復元するか、「カスタム」 を選択して時刻を選択します。

   [カスタム] を選択した場合、インスタンスを復元する日時を入力します。

   > **Note**
   >
   > 時刻は、協定世界時 (UTC) からのオフセットとしてローカルタイムゾーンで表示されます。例えば、UTC-5 は東部スタンダード時/中部夏時間です。

6. [DB エンジン] で、使用する Db2 ライセンスを選択します。
7. [DB インスタンス識別子] に、ターゲットが復元された DB インスタンスの名前を入力します。名前は一意である必要があります。
8. 必要に応じて、DB インスタンスクラス、ストレージ、ストレージのオートスケーリングを使用するかどうかなど、他のオプションを選択します。

   各設定の詳細については、「DB インスタンスの設定」を参照してください。
9. [Restore to point in time] (特定時点への復元) を選択します。

詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

AWS CLI

Db2 ライセンスを切り替えるには、restore-db-instance-to-point-in-time コマンド を使用します。次の例では、最新のポイントインタイムバージョンの復元、IBM Db2 Advanced Edition への DB エンジンの設定、AWS Marketplace 経由の Db2 ライセンスへのライセンスモデルの設定を行います。

他の設定を指定できます。各設定の詳細については、「DB インスタンスの設定」を参照してください。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier my_source_db_instance \
    --target-db-instance-identifier my_target_db_instance \
    --use-latest-restorable-time \
    --engine db2-ae \
```

```
    --license-model marketplace-license
```

Windows の場合:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier my_source_db_instance ^
    --target-db-instance-identifier my_target_db_instance ^
    --use-latest-restorable-time ^
    --engine db2-ae ^
    --license-model marketplace-license
```

詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

RDS API

Db2 ライセンスを切り替えるには、以下のパラメータを指定して Amazon RDS API `RestoreDBInstanceToPointInTime` オペレーションを呼び出します。

- `SourceDBInstanceIdentifier`
- `TargetDBInstanceIdentifier`
- `RestoreTime`
- `Engine`
- `LicenseModel`

詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

Amazon RDS for Db2 インスタンスクラス

DB インスタンスの計算とメモリの容量は、インスタンスクラスによって決まります。必要な DB インスタンスクラスは、処理能力とメモリの要件によって異なります。

サポートされている RDS for Db2 インスタンスクラス

サポートされる Amazon RDS for Db2 インスタンスクラスは、Amazon RDS DB インスタンスクラスのサブセットです。Amazon RDS インスタンスクラスの完全なリストについては、「DB インスタンスクラス」を参照してください。

トピック

- Db2 Advanced Edition のサポートされている RDS for Db2 インスタンスクラス
- Db2 Community Edition のサポートされている RDS for Db2 インスタンスクラス
- Db2 Standard Edition のサポートされている RDS for Db2 インスタンスクラス

Db2 Advanced Edition のサポートされている RDS for Db2 インスタンスクラス

次の表は、Db2 Advanced Edition でサポートされているすべてのインスタンスクラスのリストです。これらのインスタンスクラスは、Bring-Your-Own-License (BYOL) と Db2 ライセンス (AWS Marketplace 経由) の両方で使用できます。

インスタンスクラスのタイプ	インスタンスクラス
第 3 世代の Intel Xeon Scalable プロセッサ、SSD ストレージ、およびネットワーク最適化を搭載した汎用インスタンスクラス	db.m6idn.large–db.m6idn.32xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載した汎用インスタンスクラス	db.m6in.large–db.m6in.32xlarge
汎用インスタンスクラス	db.m6i.large–db.m7i.48xlarge db.m7i.large–db.m7i.48xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載し、ローカル NVMe ベースの SSD を備えたメモリ最適化インスタンスクラス	db.x2iedn.xlarge–db.x2iedn.32xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載したメモリ最適化インスタンスクラス	db.r6idn.large–db.r6idn.32xlarge db.r6in.8xlarge–db.r6in.32xlarge
メモリ最適化インスタンスクラス	db.r6i.large–db.r7i.48xlarge db.r7i.large–db.r7i.48xlarge
バーストパフォーマンスインスタンスクラス	db.t3.small–db.t3.2xlarge

Db2 Community Edition のサポートされている RDS for Db2 インスタンスクラス

次の表は、Db2 Community Edition でサポートされているすべてのインスタンスクラスのリストです。これらのインスタンスクラスは、Bring-Your-Own-License (BYOL) でのみ使用できます。

インスタンスクラスのタイプ	インスタンスクラス
汎用インスタンスクラス	db.m6i.large、db.m7i.large
バーストパフォーマンスインスタンスクラス	db.t3.small–db.t3.large

汎用インスタンスクラスは、すべての AWS リージョンで利用できるとは限りません。特定のリージョンにおけるインスタンスクラスのサポート状況を確認するには、「AWS リージョン での DB インスタンスクラスのサポートを決定する」を参照してください。

Db2 Standard Edition のサポートされている RDS for Db2 インスタンスクラス

次の表は、Db2 Standard Edition でサポートされているすべてのインスタンスクラスのリストです。これらのインスタンスクラスは、Bring-Your-Own-License (BYOL) と Db2 ライセンス (AWS Marketplace 経由) の両方で使用できます。

インスタンスクラスのタイプ	インスタンスクラス
第 3 世代の Intel Xeon Scalable プロセッサ、SSD ストレージ、およびネットワーク最適化を搭載した汎用インスタンスクラス	db.m6idn.large–db.m6idn.8xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載した汎用インスタンスクラス	db.m6in.large–db.m6in.8xlarge
汎用インスタンスクラス	db.m7i.large–db.m7i.8xlarge db.m6i.large–db.m6i.8xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載し、ローカル NVMe ベースの SSD を備えたメモリ最適化インスタンスクラス	db.x2iedn.xlarge
第 3 世代 Intel Xeon Scalable プロセッサを搭載したメモリ最適化インスタンスクラス	db.r6idn.large–db.r6idn.4xlarge

インスタンスクラスのタイプ	インスタンスクラス
	db.r6in.large–db.r6in.4xlarge
メモリ最適化インスタンスクラス	db.r7i.large–db.r7i.8xlarge db.r6i.large–db.r6i.4xlarge
バーストパフォーマンスインスタンスクラス	db.t3.small–db.t3.2xlarge

Amazon RDS for Db2 のデフォルトロール

RDS for Db2 は、次の 6 つのロールを追加し、ADMIN オプションで master_user_role に付与します。データベースがプロビジョニングされると、RDS for Db2 はマスターユーザーに master_user_role を付与します。マスターユーザーは、データベースに接続することで、ネイティブ GRANT ステートメントを持つ他のユーザー、グループ、またはロールにこれらのロールを付与できます。

- DBA – RDS for Db2 は、この空のロールを DATAACCESS 承認で作成します。マスターユーザーは、このロールにさらに承認または権限を追加し、そのロールを他のユーザー、グループ、またはロールに付与できます。
- DBA_RESTRICTED – RDS for Db2 は、この空のロールを作成します。マスターユーザーは、このロールに権限を追加でき、そのロールを他のユーザー、グループ、またはロールに付与できます。
- DEVELOPER – RDS for Db2 は、この空のロールを DATAACCESS 承認で作成します。マスターユーザーは、このロールにさらに承認または権限を追加し、そのロールを他のユーザー、グループ、またはロールに付与できます。
- ROLE_NULLID_PACKAGES – RDS for Db2 は、CREATE DATABASE の実行時に Db2 によってバインドされた ALL NULLID パッケージに対して、このロールに EXECUTE 権限を付与します。
- ROLE_PROCEDURES – RDS for Db2 は、すべての SYSIBM プロシージャでこのロールに EXECUTE 権限を付与します。
- ROLE_TABLESPACES – RDS for Db2 は、CREATE DATABASE コマンドによって作成されたテーブルスペースに対する USAGE 権限を付与します。

Amazon RDS for Db2 パラメータ

Amazon RDS for Db2 は、データベースマネージャー設定パラメータ、レジストリ変数、データベース設定パラメータの 3 種類のパラメータを使用します。最初の 2 つのタイプはパラメータグループで管理でき、最後のタイプは rdsadmin.update_db_param ストアドプロシージャで管理できます。

デフォルトでは、RDS for Db2 DB インスタンスは Db2 データベースと DB インスタンスに固有の DB パラメータグループを使用します。このパラメータグループには、IBM Db2 データベースエンジンのパラメータ、特にデータベースマネージャーの設定パラメータとレジストリ変数が含まれます。パラメータグループの操作については、「Amazon RDS のパラメータグループ」を参照してください。

RDS for Db2 データベース設定パラメータは、選択したストレージエンジンのデフォルト値に設定されます。Db2 パラメータに関する詳細については、IBM Db2 ドキュメントの「Db2 database configuration parameters」を参照してください。

トピック

- パラメータグループのパラメータの表示
- Db2 コマンドを使用したすべてのパラメータの表示
- パラメータグループのパラメータの変更
- Db2 コマンドによるデータベース設定パラメータの変更

パラメータグループのパラメータの表示

データベースマネージャー設定パラメータとレジストリ変数は、パラメータグループで設定されます。AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、特定の Db2 バージョンのデータベースマネージャー設定パラメータとレジストリ変数を表示できます。

コンソール

DB パラメータグループのパラメータ値を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。

   DB パラメータグループがリストに表示されます。
3. パラメータを一覧表示するパラメータグループの名前を選択します。

AWS CLI

describe-engine-default-parameters コマンドを実行して、Db2 バージョンのデータベースマネージャー設定パラメータとレジストリ変数を表示できます。--db-parameter-group-family オプションには、次の値のうち 1 つを指定します。

- db2-ae-11.5
- db2-se-11.5
- db2-ae-12.1
- db2-ce-12.1
- db2-se-12.1

例えば、Db2 Standard Edition 11.5 のパラメータを表示するには、次のコマンドを実行します。

```
aws rds describe-engine-default-parameters --db-parameter-group-family db2-se-11.5
```

このコマンドでは、次の例のような出力が生成されます。

```
{
    "EngineDefaults": {
        "Parameters": [
            {
                "ParameterName": "agent_stack_sz",
                "ParameterValue": "1024",
                "Description": "You can use this parameter to determine the amount of
 memory that is allocated by Db2 for each agent thread stack.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "256-32768",
                "IsModifiable": false
            },
            {
                "ParameterName": "agentpri",
                "ParameterValue": "-1",
                "Description": "This parameter controls the priority given to all
 agents and to other database manager instance processes and threads by the operating
 system scheduler. This priority determines how CPU time is allocated to the database
 manager processes, agents, and threads relative to other processes and threads running
 on the machine.",
```

```
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "1-99",
                "IsModifiable": false
            },
            ...
        ]
    }
}
```

Db2 Standard Edition 11.5 の変更可能なパラメータのみを一覧表示するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family db2-se-11.5 \
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

Windows の場合:

```
aws rds describe-engine-default-parameters ^
    --db-parameter-group-family db2-se-11.5 ^
    --query 'EngineDefaults.Parameters[?IsModifiable==`true`].
{ParameterName:ParameterName, DefaultValue:ParameterValue}'
```

RDS API

DB パラメータグループのパラメータ値を表示するには、以下の必須パラメータを指定して、DescribeDBParameters オペレーションを使用します。

- DBParameterGroupName

Db2 コマンドを使用したすべてのパラメータの表示

Db2 コマンドを使用して、データベースマネージャー設定パラメータ、データベース設定パラメータ、およびレジストリ変数の設定を表示できます。

設定を表示するには

1. Db2 データベースに接続します。次の例で、*database_name*、*master_username*、*master_password* を自分の情報に置き換えます。

```
db2 "connect to database_name user master_username using master_password"
```

2. サポートされている Db2 バージョンを検索します。

```
db2 "select service_level, fixpack_num from table(sysproc.env_get_inst_info()) as
 instanceinfo"
```

3. 特定の Db2 バージョンのパラメータを表示します。

   - 次のコマンドを実行して、データベースマネージャー設定パラメータを表示します。

```
db2 "select cast(substr(name,1,24) as varchar(24)) as name, case
    when value_flags = 'NONE' then '' else value_flags end flags,
    cast(substr(value,1,64) as varchar(64)) as current_value
    from sysibmadm.dbmcfg
    order by name asc with UR"
```

   - 次のコマンドを実行して、すべてのデータベース設定パラメータを表示します。

```
db2 "select cast(substr(name,1,24) as varchar(24)) as name, case
    when value_flags = 'NONE' then '' else value_flags end flags,
    cast(substr(value,1,64) as varchar(64)) as current_value
    from table(db_get_cfg(null)) order by name asc, member asc with UR"
```

   - 次のコマンドを実行して、現在設定されているレジストリ変数を表示します。

```
db2 "select cast(substr(reg_var_name,1,50) as varchar(50)) as reg_var_name,
    cast(substr(reg_var_value,1,50) as varchar(50)) as reg_var_value,
    level from table(env_get_reg_variables(null))
    order by reg_var_name,member with UR"
```

パラメータグループのパラメータの変更

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、カスタムパラメータグループのデータベースマネージャー設定パラメータとレジストリ変数を変更できます。詳細については、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。

コンソール

データベースマネージャー設定パラメータとレジストリ変数を変更するには

1. カスタムパラメータグループを作成します。DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. そのカスタムパラメータグループ内のパラメータを変更します。パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

AWS CLI

データベースマネージャー設定パラメータとレジストリ変数を変更するには

1. create-db-parameter-group コマンドを実行して、カスタムパラメータグループを作成します。

   次の必須パラメータを含めます。

   - `--db-parameter-group-name` - 作成するパラメータグループの名前。
   - `--db-parameter-group-family` - Db2 エンジンエディションとメジャーバージョン。有効な値:`db2-ae-11.5` 、`db2-se-11.5`、`db2-ae-12.1`、`db2-ce-12.1`、`db2-se-12.1`
   - `--description` - このパラメータグループの説明。

   DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. modify-db-parameter-group コマンドを実行して、作成したカスタムパラメータグループのパラメータを変更します。

   次の必須パラメータを含めます。

   - `--db-parameter-group-name` – 作成したパラメータグループの名前。

- `--parameters` – パラメータ名、値、およびパラメータ更新用のアプリケーションメソッドの配列。

パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

RDS API

データベースマネージャー設定パラメータとレジストリ変数を変更するには

1. CreateDBParameterGroup 操作を使用して、カスタム DB パラメータグループを作成します。

   以下の必須パラメータを含めます。

   - `DBParameterGroupName`
   - `DBParameterGroupFamily`
   - `Description`

   DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. ModifyDBParameterGroup 操作を使用して、作成したカスタムパラメータグループのパラメータを変更します。

   以下の必須パラメータを含めます。

   - `DBParameterGroupName`
   - `Parameters`

   パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

Db2 コマンドによるデータベース設定パラメータの変更

Db2 コマンドでデータベース設定パラメータを変更できます。

データベース設定パラメータを変更するには

1. rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.update_db_param ストアドプロシージャを呼び出して、データベース設定パラメータを変更します。詳細については、「rdsadmin.update_db_param」を参照してください。

```
db2 "call rdsadmin.update_db_param(
    'database_name',
    'parameter_to_modify',
    'changed_value',
    'restart_database')"
```

Amazon RDS 上の Db2 データベースの EBCDIC 照合

Amazon RDS for Db2 は、Db2 データベースの EBCDIC 照合をサポートしています。データベースの EBCDIC 照合順序は、Amazon RDS the section called “rdsadmin.create_database” ストアドプロシージャを使用してデータベースを作成するときのみ指定できます。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して RDS for Db2 DB インスタンスを作成する場合、データベース名を指定できます。データベース名を指定すると、Amazon RDS は SYSTEM のデフォルトの照合を使用してデータベースを作成します。EBCDIC 照合を使用してデータベースを作成する必要がある場合は、DB インスタンスの作成時にデータベース名を指定しないでください。

RDS for Db2 のデータベースの照合は、作成時に設定され、変更できません。

EBCDIC 照合を使用して Db2 データベースを作成するには

1. RDS for Db2 DB インスタンスがない場合は、DB インスタンスを作成しても、データベース名は指定しないでください。AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB インスタンスを作成してください。詳細については、「DB インスタンスの作成」を参照してください。
2. Db2 データベースを作成し、rdsadmin.create_database ストアドプロシージャを呼び出して照合オプションを EBCDIC 値に設定します。詳細については、「rdsadmin.create_database」を参照してください。

> **Important**
>
> ストアドプロシージャを使用してデータベースを作成した後で、照合順序を変更することはできません。データベースで別の照合順序を使用する場合は、the section called “rdsadmin.drop_database” ストアドプロシージャを呼び出してデータベースを削除します。次に、必要な照合順序でデータベースを作成します。

Amazon RDS for Db2 DB インスタンスのローカルタイムゾーン

Db2 を実行している Amazon RDS DB インスタンスのタイムゾーンは、デフォルトで設定されます。デフォルトは協定世界時 (UTC) です。アプリケーションのタイムゾーンを一致させるには、代わりに DB インスタンスのタイムゾーンをローカルタイムゾーンに設定できます。

DB インスタンスを最初に作成するときにタイムゾーンを設定します。DB インスタンスを作成するには、AWS マネジメントコンソール、RDS API、または AWS CLI を使用できます。詳細については、「DB インスタンスの作成」を参照してください。

DB インスタンスがマルチ AZ 配置の一部である場合は、フェイルオーバーしても、タイムゾーンは設定したローカルタイムゾーンのままになります。

DB インスタンスは、指定した時点に復元できます。時間は、ローカルタイムゾーンで表示されます。詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

DB インスタンスでローカルタイムゾーンを設定する場合は、以下の制限事項があります。

- 既存の Amazon RDS for Db2 DB インスタンスのタイムゾーンを変更することはできません。
- あるタイムゾーンの DB インスタンスのスナップショットを、タイムゾーンの異なる DB インスタンスに復元することはできません。
- あるタイムゾーンのバックアップファイルを、別のタイムゾーンに復元しないことを強くお勧めします。バックアップファイルを別のタイムゾーンに復元した場合は、タイムゾーンの変更によるクエリとアプリケーションへの影響を監査する必要があります。

利用可能なタイムゾーン

タイムゾーン設定には、以下の値を使用できます。

ゾーン	タイムゾーン
アフリカ	Africa/Cairo、Africa/Casablanca、Africa/Harare、Africa/Lagos、Africa/Luanda、Africa/Monrovia、Africa/Nairobi、Africa/Tripoli、Africa/Windhoek
南北アメリカ大陸	America/Araguaina、America/Argentina/Buenos_Aires、America/Asuncion、America/Bogota、America/Caracas、America/Chicago、America/Chihuahua、America/Cuiaba、America/Denver、America/Detroit、America/Fortaleza、America/Godthab、America/Guatemala、America/Halifax、America/Lima、America/Los_Angeles、America/Manaus、America/Matamoros、America/Mexico_City、America/Monterrey、America/Montevideo、America/New_York、America/Phoenix、America/Santiago、America/Sao_Paulo、America/Tijuana、America/Toronto
アジア	Asia/Amman、Asia/Ashgabat、Asia/Baghdad、Asia/Baku、Asia/Bangkok、Asia/Beirut、Asia/Calcutta、Asia/Damascus、Asia/Dhaka、Asia/Hong_Kong、Asia/Irkutsk、Asia/Jakarta、Asia/Jerusalem、Asia/Kabul、Asia/Karachi、Asia/Kathmandu、Asia/Kolkata、Asia/Krasnoyarsk、Asia/Magadan、Asia/Manila、Asia/Muscat、Asia/Novosibirsk、Asia/Rangoon、Asia/Riyadh、Asia/Seoul、Asia/Shanghai、Asia/Singapore、Asia/Taipei、Asia/Tehran、Asia/Tokyo、Asia/Ulaanbaatar、Asia/Vladivostok、Asia/Yakutsk、Asia/Yerevan
大西洋	Atlantic/Azores、Atlantic/Cape_Verde
オーストラリア	Australia/Adelaide、Australia/Brisbane、Australia/Darwin、Australia/Eucla、Australia/Hobart、Australia/Lord_Howe、Australia/Perth、Australia/Sydney
ブラジル	Brazil/DeNoronha、Brazil/East
カナダ	Canada/Newfoundland、Canada/Saskatchewan
ETC	Etc/GMT-3
欧州	Europe/Amsterdam、Europe/Athens、Europe/Berlin、Europe/Dublin、Europe/Helsinki、Europe/Kaliningrad、Europe/London、Europe/Madrid、Europe/

ゾーン	タイムゾーン
	Moscow、Europe/Paris、Europe/Prague、Europe/Rome、Europe/Sarajevo、Europe/Stockholm
太平洋	Pacific/Apia、Pacific/Auckland、Pacific/Chatham、Pacific/Fiji、Pacific/Guam、Pacific/Honolulu、Pacific/Kiritimati、Pacific/Marquesas、Pacific/Samoa、Pacific/Tongatapu、Pacific/Wake
米国	US/Alaska、US/Central、US/East-Indiana、US/Eastern、US/Pacific
UTC	UTC

Amazon RDS for Db2 DB インスタンス作成の前提条件

以下は DB インスタンスを作成するための前提条件です。

トピック

- 管理者アカウント
- その他の考慮事項

管理者アカウント

DB インスタンスを作成するときは、インスタンスの管理者アカウントを指定する必要があります。Amazon RDS は、このローカルデータベース管理者アカウントに DBADM 権限を付与します。

管理者アカウントには、次の特性、機能、および制限があります。

- ローカルユーザーであり、AWS アカウント ではない。
- SYSADM、SYSMAINT、 SYSCTRL など、Db2 インスタンスレベルの権限は持たない。
- Db2 インスタンスを停止または開始することはできない。
- DB インスタンスの作成時に名前を指定した場合、Db2 データベースを削除することはできない。
- カタログテーブルとビューを含む Db2 データベースへのフルアクセス権を持つ。
- Amazon RDS ストアドプロシージャを使用して、ローカルユーザーとグループを作成できる。
- 権限と特権を付与および取り消すことができる。

管理者アカウントは以下のタスクを実行できます。

- DB インスタンスの作成、変更、または削除。
- DB スナップショットの作成。
- ポイントインタイム復元の開始。
- DB スナップショットの自動バックアップの作成。
- DB スナップショットの手動バックアップの作成。
- 他の Amazon RDS 機能の使用。

その他の考慮事項

DB インスタンスを作成する前に、以下を考慮してください。

- 各 Amazon RDS for Db2 DB インスタンスは、最大 50 の Db2 データベースをホストできます。詳細については、「Amazon RDS for Db2 DB インスタンス上の複数のデータベース」を参照してください。
- 初期データベース名
  - DB インスタンスの作成時にデータベース名を指定しないと、Amazon RDS はデータベースを作成しません。
  - 以下の状況では、データベース名を指定しないでください。
    - `db2_compatibility_vector` パラメータを変更してください。詳細については、「db2_compatibility_vector パラメータの設定」を参照してください。
- Bring-Your-Own-License (BYOL) モデルでは、まず IBM Customer ID と IBM Site ID を含むカスタムパラメータグループを作成する必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL)」を参照してください。
- AWS Marketplace 経由の Db2 ライセンスモデルでは、使用する特定の IBM Db2 エディションの有効な AWS Marketplace サブスクリプションが必要です。まだお持ちでない場合は、使用する IBM Db2 エディションの Db2 サブスクリプションを AWS Marketplace で購入する必要があります。詳細については、「AWS Marketplace 経由の Db2 ライセンス」を参照してください。

Amazon RDS for Db2 DB インスタンス上の複数のデータベース

rdsadmin.create_database ストアドプロシージャを呼び出すことで、単一の RDS for Db2 DB インスタンスに複数のデータベースを作成できます。1 つの RDS for Db2 DB インスタンスに作成できるデータベースの数は 50 に制限されています。この数には、アクティブ状態と非アクティブ状態の両方のデータベースが含まれます。

> **Note**
>
> 2024 年 11 月 15 日より前に作成された RDS for Db2 DB インスタンスに複数のデータベースを作成する場合は、DB インスタンスを再起動して、複数のデータベースのサポートを有効にする必要があります。

デフォルトでは、Amazon RDS はデータベースの作成時にデータベースをアクティブ化します。メモリリソースを最適化するために、使用頻度の低いデータベースを非アクティブ化したり、必要に応じて後でアクティブ化したりできます。詳細については、「the section called “データベースの非アクティブ化”」および「the section called “データベースのアクティブ化”」を参照してください。

DB インスタンスでアクティブ化されるデータベースの数は、サーバーで使用可能なメモリリソースによって異なります。メモリリソースは、DB インスタンスクラスと、データベースに設定されたメモリの量によって異なります。DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。RDS for Db2 データベースのメモリを更新する方法については、「the section called “rdsadmin.update_db_param”」を参照してください。

一般的なデータベースタスク、オペレーティングシステムの要件、バックアップなどの他の Amazon RDS オートメーションタスクには、メモリが 2 GB に設定された DB インスタンスクラスを選択することをお勧めします。DB インスタンスクラスの変更について詳しくは、「the section called “DB インスタンスを変更する”」を参照してください。

さらに、IBM では、アクティブなデータベースごとに最低 1 GB のメモリを推奨しています。詳細については、IBM ドキュメントの「ディスクとメモリの要件」を参照してください。

DB インスタンスが保持できるアクティブなデータベースの最大数は、次の式を使用して計算できます。

```
Active database limit = (total server memory - 2 GB) / 1 GB
```

次の例は、db.m6i.xlarge DB インスタンスクラスを持つ DB インスタンスのアクティブなデータベースの最大数を示しています。

```
Active database limit = (total server memory - 2 GB) / 1 GB
                      = (16 GB - 2 GB) / 1 GB
                      = 14 databases
```

クラッシュ後に Amazon RDS がデータベースを復旧すると、以前にアクティブだったデータベースがアクティブ化されます。DB インスタンスクラスをより低いメモリ設定に変更する場合など、特定のケースでは、DB インスタンス上のすべてのデータベースをアクティブ化するのに十分なメモリがない可能性があります。このような場合、Amazon RDS はデータベースを作成された順序でアクティブ化します。

> **Note**
>
> メモリ不足のために Amazon RDS がアクティブ化できないデータベースは、非アクティブ化された状態のままになります。

Db2 DB インスタンスへの接続

Amazon RDS が Amazon RDS for Db2 DB インスタンスをプロビジョニングした後、標準の SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。Amazon RDS はマネージドサービスであるため、SYSADM、SYSCTRL、SECADM、または SYSMAINT としてサインインすることはできません。

IBM Db2 CLP、IBM CLPPlus、DBeaver、または IBM Db2 Data Management Console を使用して、IBM Db2 データベースエンジンを実行している DB インスタンスに接続できます。

> **Note**
>
> RDS for Db2 DB インスタンスに十分なメモリがない場合、Db2 データベースへの接続が失敗する可能性があります。詳細については、「the section called “データベース接続エラー”」を参照してください。

トピック

- Amazon RDS for Db2 DB インスタンスのエンドポイントを発見する
- IBM Db2 CLP を使用して Amazon RDS for Db2 DB インスタンスに接続する
- IBM CLPPlus を使用して Amazon RDS for Db2 DB インスタンスに接続する
- DBeaver を使用して Amazon RDS for Db2 DB インスタンスに接続する
- IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続する
- Amazon RDS for Db2 のセキュリティグループの考慮事項

Amazon RDS for Db2 DB インスタンスのエンドポイントを発見する

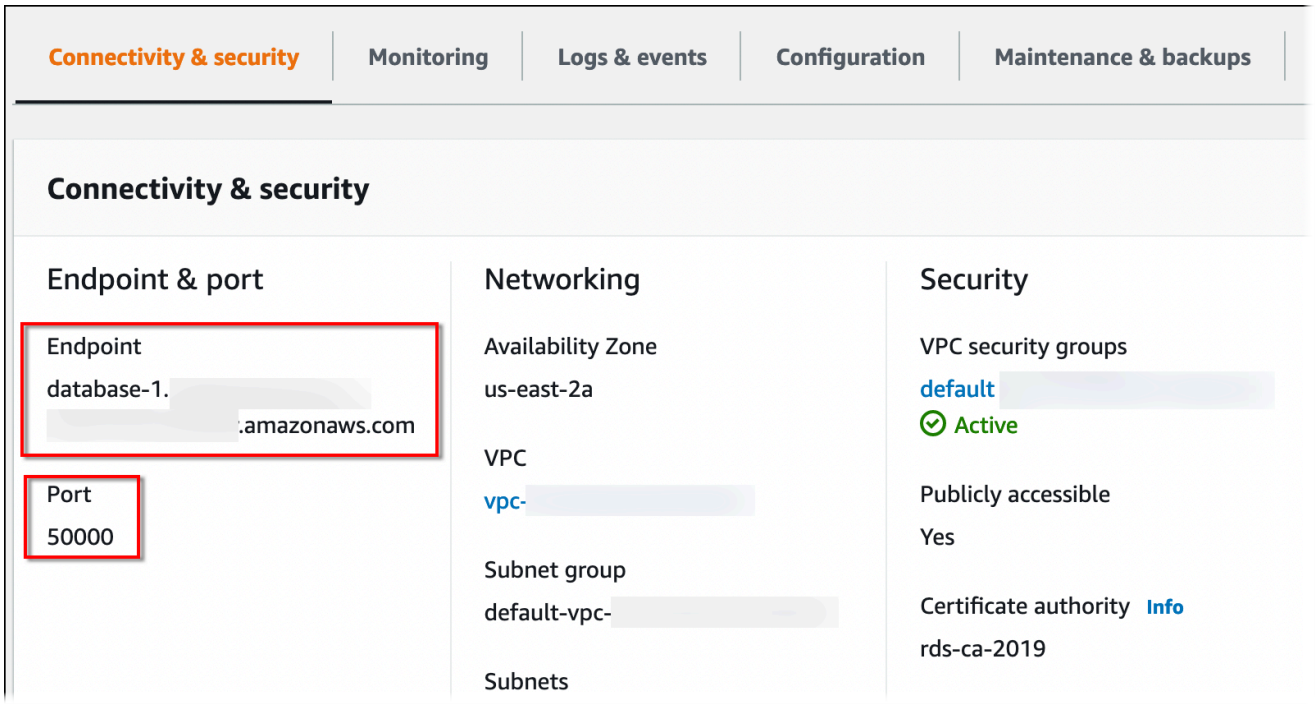
各 Amazon RDS DB インスタンスにはエンドポイントがあり、各エンドポイントに DB インスタンスの DNS 名とポート番号があります。SQL クライアントアプリケーションを使用して Amazon RDS for Db2 DB インスタンスに接続するには、DB インスタンスの DNS 名とポート番号が必要です。

AWS マネジメントコンソール コンソールまたは AWS CLI を使用して DB インスタンスのエンドポイントを確認できます。

コンソール

RDS for Db2 DB インスタンスのエンドポイントを確認するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. コンソールで、DB インスタンスの AWS リージョンを選択します。
3. RDS for Db2 DB インスタンスの DNS 名とポート番号を見つけます。
   a. [データベース] を選択して DB インスタンスを一覧表示します。
   b. 詳細を表示する RDS for Db2 DB インスタンスの名前を選択します。
   c. 接続とセキュリティ タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



AWS CLI

RDS for Db2 DB インスタンスのエンドポイントを確認するには、describe-db-instances コマンドを実行します。次の例では、*database-1* を DB インスタンスの名前に置き換えます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
    --db-instance-identifier database-1 \
    --query 'DBInstances[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' \
    --output json
```

Windows の場合:

```
aws rds describe-db-instances ^
    --db-instance-identifier database-1 ^
    --query 'DBInstances[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBName:DBName,Endpoint:Endpoint}' ^
    --output json
```

このコマンドでは、次の例のような出力が生成されます。出力の Address 行には DNS 名が含まれています。

```
[
    {
        "DBInstanceIdentifier": "database-1",
        "DBName": "DB2DB",
        "Endpoint": {
            "Address": "database-1.123456789012.us-east-2.amazonaws.com",
            "Port": 50000,
            "HostedZoneId": "Z2OC4A7DETW6VH"
        }
    }
]
```

IBM Db2 CLP を使用して Amazon RDS for Db2 DB インスタンスに接続する

IBM Db2 CLP などのコマンドラインユーティリティを使用して、Amazon RDS for Db2 DB インスタンスに接続できます。このユーティリティは IBM Data Server Runtime Client の一部です。IBM Fix Central からクライアントをダウンロードするには、IBM サポートの「IBM Data Server Client Packages Version 11.5 Mod 8 Fix Pack 0」を参照してください。

トピック

- 用語

- クライアントをインストールする
- DB インスタンスに接続する
- RDS for Db2 DB インスタンスへの接続のトラブルシューティング

用語

次の用語は、RDS for Db2 DB インスタンス に接続する際に使用するコマンドの説明として役立ちます。

catalog tcpip node

このコマンドは、リモートデータベースノードをローカル Db2 クライアントに登録し、クライアントアプリケーションがノードにアクセスできるようにします。ノードをカタログ化するには、サーバーのホスト名、ポート番号、通信プロトコルなどの情報を指定します。カタログ化されたノードは、1 つ以上のリモートデータベースが存在するターゲットサーバーを表します。詳細については、IBM Db2 ドキュメントの「CATALOG TCPIP/TCPIP4/TCPIP6 NODE command」を参照してください。

catalog database

このコマンドは、リモートデータベースをローカル Db2 クライアントに登録し、クライアントアプリケーションがデータベースにアクセスできるようにします。データベースをカタログ化するには、データベースのエイリアス、データベースが存在するノード、データベースへの接続に必要な認証タイプなどの情報を提供します。詳細については、IBM Db2 ドキュメントの「CATALOG DATABASE command」を参照してください。

クライアントをインストールする

downloading the package for Linux の後に、ルート権限または管理者権限を使用してクライアントをインストールします。

> **Note**
>
> クライアントを AIX または Windows にインストールするには、同じ手順に従いますが、オペレーティングシステムに応じてコマンドを変更します。

Linux にクライアントをインストールするには

1. **./db2_install -f sysreq** を実行し、**yes** を選択してライセンス契約に同意します。
2. クライアントをインストールする場所を選択します。
3. **clientInstallDir/instance/db2icrt -s client *instance_name*** を実行します。*instance_name* は、Linux の有効なオペレーティングシステムユーザーに置き換えます。Linux では、Db2 DB インスタンス名はオペレーティングシステムのユーザー名に関連付けられます。

   このコマンドは、Linux の指定ユーザーのホームディレクトリに **sqllib** ディレクトリを作成します。

DB インスタンスに接続する

RDS for Db2 DB インスタンスに接続するには、DNS 名とポート番号が必要です。これを確認する方法の詳細については、「エンドポイントの検索」を参照してください。また、RDS for Db2 DB インスタンスの作成時に定義したデータベース名、マスターユーザー名、マスターパスワードも把握しておく必要があります。これを確認する方法の詳細については、「DB インスタンスの作成」を参照してください。

IBM Db2 CLP を使用して RDS for Db2 DB インスタンスに接続するには

1. IBM Db2 CLP クライアントのインストール時に指定したユーザー名でサインインします。
2. RDS for Db2 DB インスタンスをカタログ化します。次の例では、*node_name*、*dns_name*、*port* をローカルカタログ内のノード名、DB インスタンスの DNS 名、ポート番号に置き換えます。

   ```
   db2 catalog TCPIP node node_name remote dns_name server port
   ```

   例

   ```
   db2 catalog TCPIP node remnode remote database-1.123456789012.us-
   east-1.amazonaws.com server 50000
   ```
3. rdsadmin データベースと自身のデータベースをカタログ化します。これにより、Amazon RDS ストアドプロシージャを使用して rdsadmin データベースに接続し、一部の管理タスクを実行できるようになります。詳細については、「RDS for Db2 DB インスタンスの管理」を参照してください。

次の例では、*database_alias*、*node_name*、*database_name* をこのデータベースのエイリアス、前のステップで定義したノード名、データベースの名前に置き換えます。server_encrypt は、ネットワーク経由でユーザー名とパスワードを暗号化します。

```
db2 catalog database rdsadmin [ as database_alias ] at node node_name
 authentication server_encrypt

db2 catalog database database_name [ as database_alias ] at node node_name
 authentication server_encrypt
```

例

```
db2 catalog database rdsadmin at node remnode authentication server_encrypt

db2 catalog database testdb as rdsdb2 at node remnode authentication server_encrypt
```

4. RDS for Db2 データベースに接続します。次の例では、*rds_database_alias*、*master_username*、*master_password* を、データベース名、RDS for Db2 DB インスタンスのマスターユーザー名、マスターパスワードに置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

このコマンドでは、次の例のような出力が生成されます。

```
Database Connection Information

    Database server        = DB2/LINUXX8664 11.5.9.0
    SQL authorization ID   = ADMIN
    Local database alias   = TESTDB
```

5. クエリを実行し、結果を表示します。次の例は、作成したデータベースを選択する SQL ステートメントを示しています。

```
db2 "select current server from sysibm.dual"
```

このコマンドでは、次の例のような出力が生成されます。

```
1
```

```
------------------
TESTDB

1 record(s) selected.
```

RDS for Db2 DB インスタンスへの接続のトラブルシューティング

次の NULLID エラーが表示される場合は、通常、クライアントと RDS for Db2 サーバーのバージョンが一致していないことを示しています。サポートされている Db2 クライアントバージョンについては、IBM Db2 ドキュメントの「Supported combinations of clients, drivers and server levels」を参照してください。

```
db2 "select * from syscat.tables"
SQL0805N Package "NULLID.SQLC2O29 0X4141414141454A69" was not found.
SQLSTATE=51002
```

このエラーが表示された場合は、古い Db2 クライアントから RDS for Db2 でサポートされている Db2 サーバーバージョンにパッケージをバインドする必要があります。

古い Db2 クライアントから新しい Db2 サーバーにパッケージをバインドするには

1. クライアントマシン上のバインドファイルを見つけます。通常、これらのファイルは Db2 クライアントのインストールパスの bnd ディレクトリにあり、.bnd という拡張子が付いています。
2. Db2 サーバーに接続します。次の例では、*database_name* を Db2 サーバーの名前に置き換えます。*master_username* と *master_password* をユーザー自身の情報に置き換えます。このユーザーには DBADM 権限があります。

   ```
   db2 connect to database_name user master_username using master_password
   ```
3. bind コマンドを実行して、パッケージをバインドします。

   a. クライアントマシン上のバインドファイルが存在するディレクトリに移動します。

   b. 各ファイルに対して bind コマンドを実行します。

      以下のオプションは必須です。

      - blocking all - バインドファイル内のすべてのパッケージを 1 つのデータベースリクエストにバインドします。
      - grant public - パッケージを実行するアクセス許可を public に付与します。

- `sqlerror continue` - エラーが発生した場合でも bind プロセスが継続するよう指定します。

  bind コマンドの詳細については、IBM Db2 ドキュメントの「BIND command」を参照してください。

4. `syscat.package` カタログビューにクエリを実行するか、bind コマンドの後に返されたメッセージを確認して、正常にバインドされたことを確認します。

詳細については、IBM サポートの「DB2 v11.5 Bind File and Package Name List」を参照してください。

IBM CLPPlus を使用して Amazon RDS for Db2 DB インスタンスに接続する

IBM CLPPlus などのユーティリティを使用して、Amazon RDS for Db2 DB インスタンスに接続できます。このユーティリティは IBM Data Server Runtime Client の一部です。IBM Fix Central からクライアントをダウンロードするには、IBM サポートの「IBM Data Server Client Packages Version 11.5 Mod 8 Fix Pack 0」を参照してください。

> **Important**
>
> グラフィカルユーザーインターフェイスをサポートする macOS、Windows、または Linux with Desktop などのオペレーティングシステムで IBM CLPPlus を実行することをお勧めします。ヘッドレス Linux を実行している場合は、CLPPlus コマンドでスイッチ -nw を使用します。

トピック

- クライアントをインストールする
- DB インスタンスに接続する
- DB2 ストアドプロシージャから CLOB データを取得する

クライアントをインストールする

Linux のパッケージをダウンロードしたら、クライアントをインストールします。

> **Note**
>
> クライアントを AIX または Windows にインストールするには、同じ手順に従いますが、オペレーティングシステムに応じてコマンドを変更します。

Linux にクライアントをインストールするには

1. を実行します。。**./db2_install**
2. **clientInstallDir/instance/db2icrt -s client *instance_name*** を実行します。*instance_name* は、Linux の有効なオペレーティングシステムユーザーに置き換えます。Linux では、Db2 DB インスタンス名はオペレーティングシステムのユーザー名に関連付けられます。

   このコマンドは、Linux の指定ユーザーのホームディレクトリに **sqllib** ディレクトリを作成します。

DB インスタンスに接続する

RDS for Db2 DB インスタンスに接続するには、DNS 名とポート番号が必要です。これを確認する方法の詳細については、「エンドポイントの検索」を参照してください。また、RDS for Db2 DB インスタンスの作成時に定義したデータベース名、マスターユーザー名、マスターパスワードも把握しておく必要があります。これを確認する方法の詳細については、「DB インスタンスの作成」を参照してください。

IBM CLPPlus を使用して RDS for Db2 DB インスタンスに接続するには

1. コマンドの構文を確認します。次の例では、*clientDir* をクライアントがインストールされている場所に置き換えます。

   ```
   cd clientDir/bin
       ./clpplus -h
   ```

2. Db2 サーバーを設定します。次の例では、*dsn_name*、*database_name*、*endpoint*、*port* を DSN 名、データベース名、エンドポイント、RDS for Db2 DB インスタンスのポートに置き換えます。詳細については、「Amazon RDS for Db2 DB インスタンスのエンドポイントを発見する」を参照してください。

```
db2cli writecfg add -dsn dsn_name -database database_name -host endpoint -port port
 -parameter "Authentication=SERVER_ENCRYPT"
```

3. RDS for Db2 DB インスタンスに接続します。次の例では、*master_username* と *dsn_name* をマスターユーザー名と DSN 名に置き換えます。

```
./clpplus -nw master_username@dsn_name
```

4. Java Shell ウィンドウが開きます。RDS for Db2 DB インスタンスのマスターパスワードを入力します。

> **Note**
>
> Java Shell ウィンドウが開かない場合は、**./clpplus -nw** を実行して同じコマンドラインウィンドウを使用します。

```
Enter password: *********
```

接続が確立され、次の例のような出力が生成されます。

```

Database Connection Information :
---------------------------------
Hostname = database-1.abcdefghij.us-east-1.rds.amazonaws.com
Database server = DB2/LINUXX8664  SQL110590
SQL authorization ID = admin
Local database alias = DB2DB
Port = 50000
```

5. クエリを実行し、結果を表示します。次の例は、作成したデータベースを選択する SQL ステートメントを示しています。

```
SQL > select current server from sysibm.dual;
```

このコマンドでは、次の例のような出力が生成されます。

```
1
    --------------------
```

```
    DB2DB
    SQL>
```

DB2 ストアドプロシージャから CLOB データを取得する

rdsadmin.db2pd_command などのストアドプロシージャは、最大 2 GB のデータをサポートする CLOB 列で結果を返します。ただし、DB2 CLP は CLOB 出力を 8 KB (8192 バイト) に制限し、このしきい値を超えるデータは切り捨てられます。完全な出力を取得するには、代わりに CLPPLUS を使用します。

1. タスク ID (task_id) の取得

```
db2 "select task_id, task_type, database_name, lifecycle,
 varchar(bson_to_json(task_input_params), 500) as task_params,
cast(task_output as varchar(500)) as task_output, CREATED_AT, LAST_UPDATED_AT from
 table(rdsadmin.get_task_status(null,null,null))"
```

2. CLPPLUS コマンドを実行する

   task_id を取得したら、Unix プロンプトから次のコマンドを実行します (TASK_ID を実際の数値タスク ID に置き換えます)。

```
$ (echo "select task_output from
 table(rdsadmin.get_task_status(task_id,null,null));" ; echo "disconnect;" ; echo
 "exit;") | clpplus -nw -silent masteruser/MasterUserPassword@hostname:port_num/
rdsadmin
```

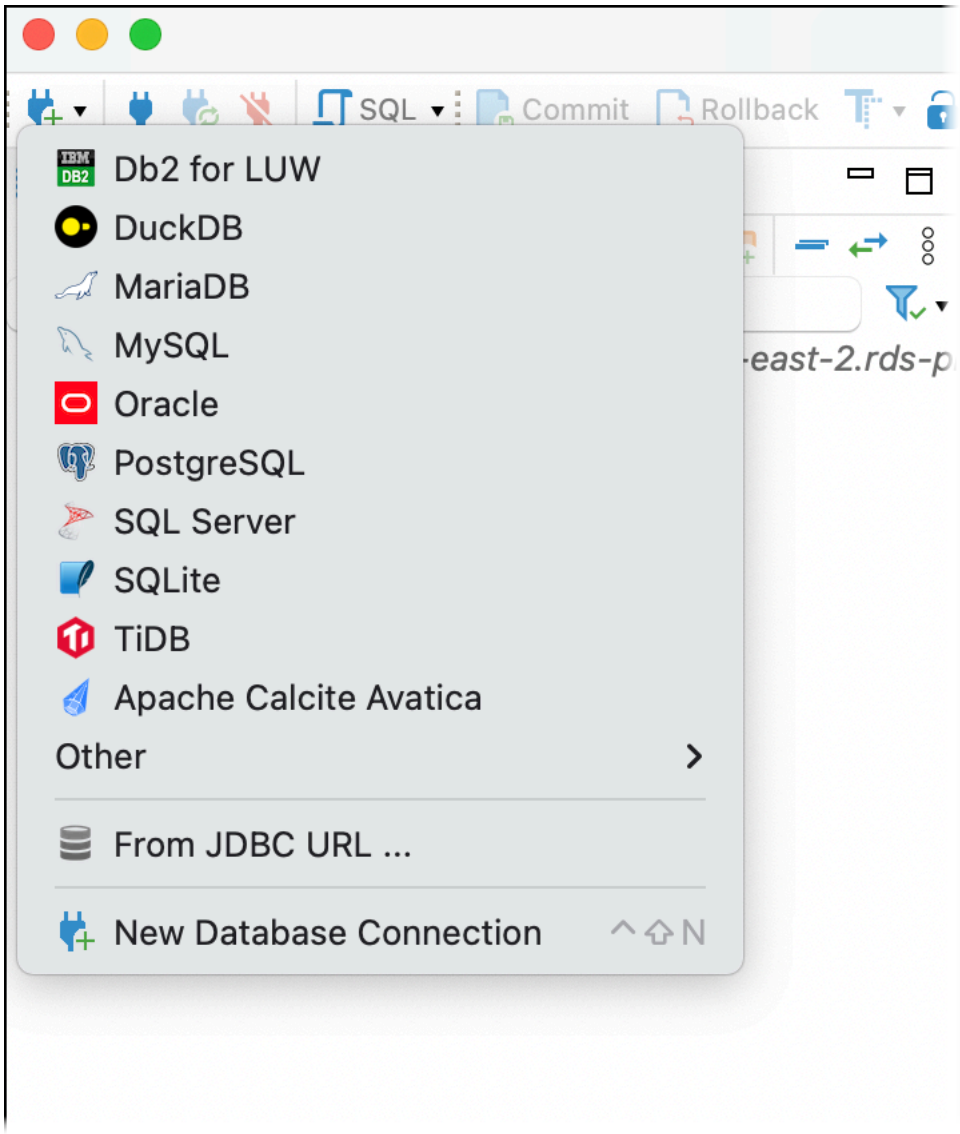
DBeaver を使用して Amazon RDS for Db2 DB インスタンスに接続する

DBeaver などのサードパーティーツールを使用して、Amazon RDS for Db2 DB インスタンスに接続できます。このユーティリティをダウンロードするには、「DBeaver Community」を参照してください。

RDS for Db2 DB インスタンスに接続するには、DNS 名とポート番号が必要です。これを確認する方法の詳細については、「エンドポイントの検索」を参照してください。また、RDS for Db2 DB インスタンスの作成時に定義したデータベース名、マスターユーザー名、マスターパスワードも把握しておく必要があります。これを確認する方法の詳細については、「DB インスタンスの作成」を参照してください。

DBeaver を使用して RDS for Db2 DB インスタンスに接続するには

1. 起動します。DBeaver
2. ツールバーの [New Connection] アイコンを選択し、[Db2 for LUW] を選択します。



3. [Connect to a database] ウィンドウで、RDS for Db2 DB インスタンスの情報を入力します。

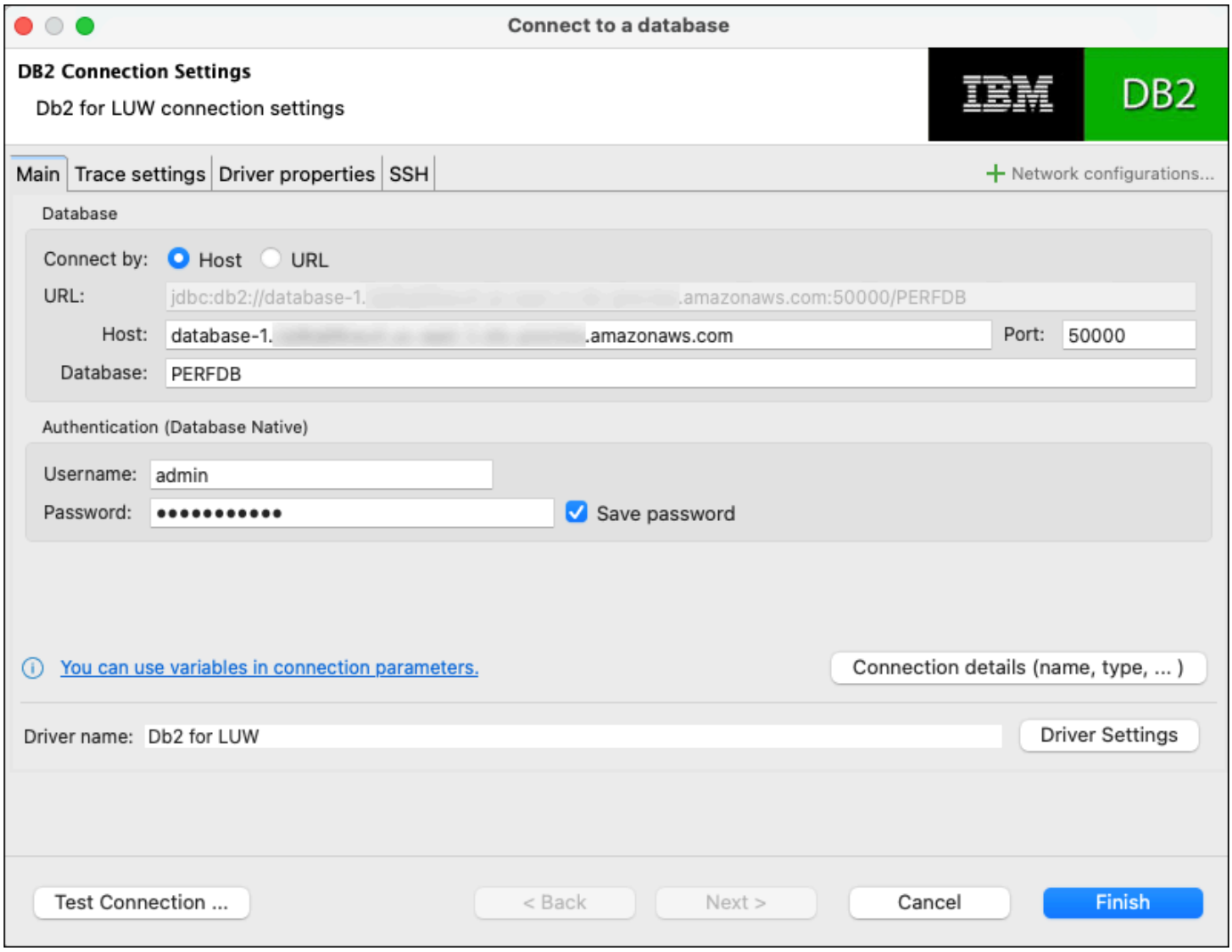
   a. 次の情報を入力します。

   - [Host] に、DB インスタンスの DNS 名を入力します。
   - [Port] に、DB インスタンスのポート番号を入力します。
   - [Database] に、データベースの名前を入力します。
   - [Username] に、DB インスタンスのデータベース管理者の名前を入力します。

- [Password] に、DB インスタンスのデータベース管理者のパスワードを入力します。

b. [Save password] を選択します。

c. [Driver Settings] を選択します。



4. [Edit Driver] ウィンドウで、追加のセキュリティプロパティを指定します。

a. [Driver properties] タブを選択します。

b. 2 つの User Properties を追加します。

i. コンテキスト (右クリック) メニューを開き、[Add new property] を選択します。

ii. [Property Name] で [encryptionAlgorithm] を追加し、[OK] を選択します。

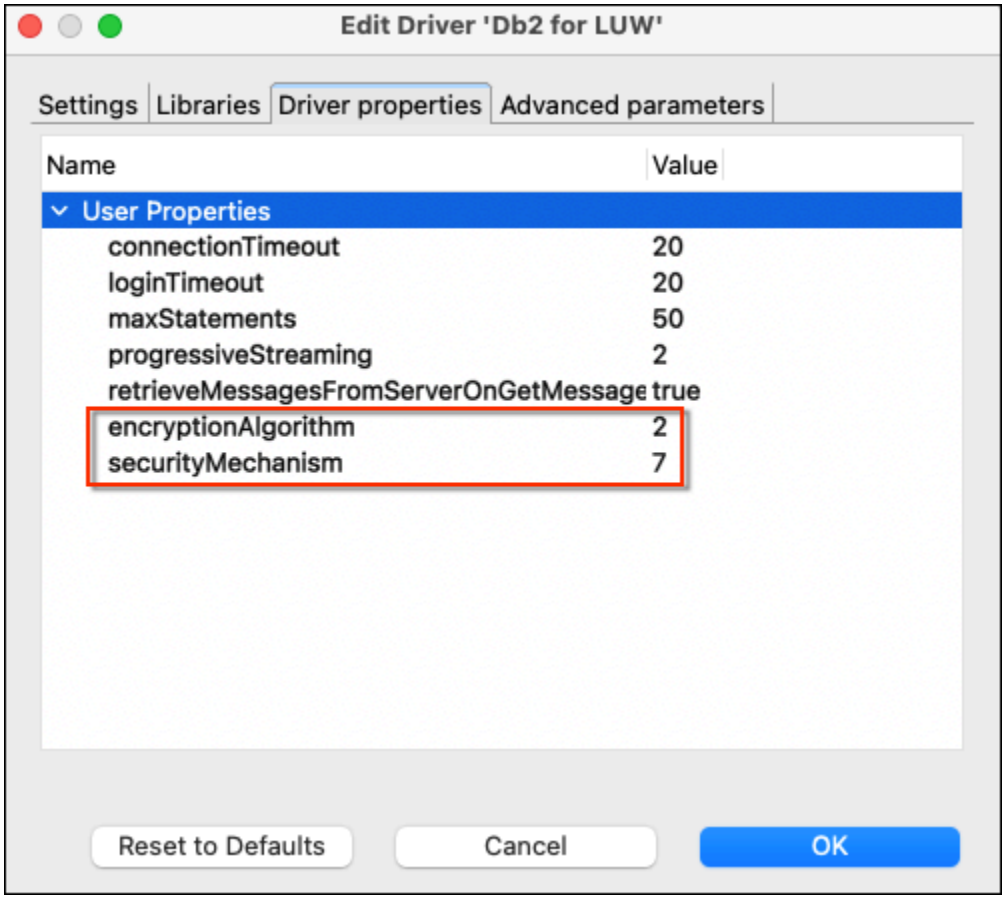
iii. [encryptionAlgorithm] 行を選択し、[Value] 列を選択して [2] を追加します。

iv. コンテキスト (右クリック) メニューを開き、[Add new property] を選択します。

v. [Property Name]で [securityMechanism] を追加し、[OK] を選択します。

vi. [securityMechanism] 行を選択し、[Value] 列を選択して [7] を追加します。

c. [OK] を選択してください。



5. [Connect to a database] ウィンドウで、[Test Connection] を選択します。コンピュータに DB2 JBDC ドライバーがインストールされていない場合、ドライバーは自動的にダウンロードされます。

6. [OK] を選択してください。

7. [Finish] を選択します。

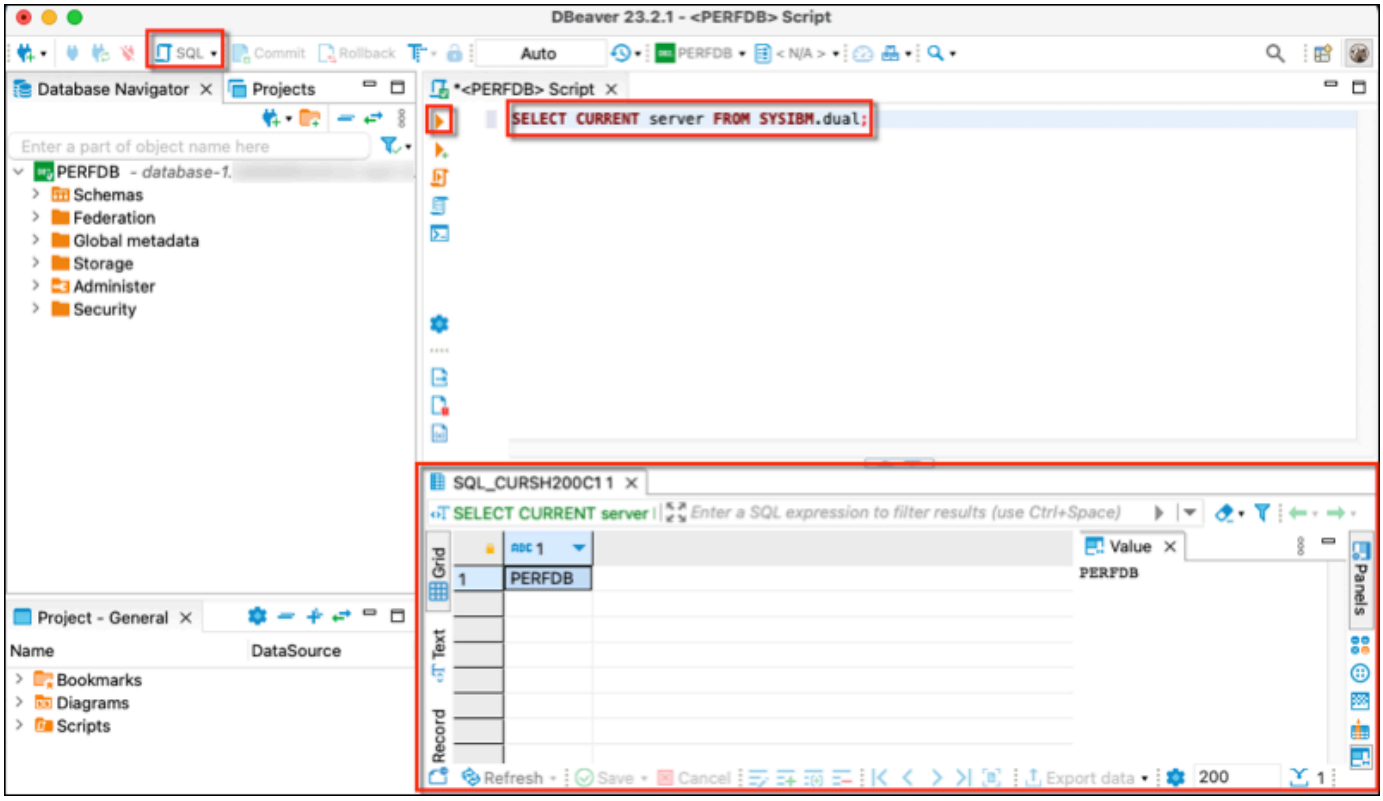
8. [Database Navigation] タブで、データベースの名前を選択します。これでオブジェクトを探索できるようになりました。

これで、SQL コマンドを実行する準備ができました。

SQL コマンドを実行して結果を表示するには

1. 上部メニューで、[SQL] を選択します。SQL スクリプトパネルが開きます。

2. [Script] パネルに、SQL コマンドを入力します。
3. コマンドを実行するには、[Execute SQL query] ボタンを選択します。
4. SQL の結果パネルで、SQL クエリの結果を表示します。



IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続する

IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続できます。IBM Db2 Data Management Console は、複数の RDS for Db2 DB インスタンスを管理およびモニタリングできます。

> **Note**
>
> RDS for Db2 DB インスタンスと同じ VPC およびセキュリティグループにある Amazon EC2 Linux または Windows マシンが必要です。VPC とセキュリティグループは、内部ネットワークを介して DB インスタンスへの接続を制御します。

IBM Db2 Data Management Console では、メタデータとパフォーマンスメトリクスを保存するためにリポジトリ Db2 データベースが必要ですが、RDS for Db2 のリポジトリデータベースを自動的に作成することはできません。代わりに、まずリポジトリデータベースを作成して、1 つ以上の RDS for Db2 DB インスタンスをモニタリングする必要があります。次に、IBM Db2 Data Management Console をインストールし、IBM Db2 Data Management Console を使用して RDS for Db2 DB インスタンスに接続できます。

トピック

- ステップ 1: DB インスタンスをモニタリングするためのリポジトリデータベースを作成する
- ステップ 2: IBM Db2 Data Management Console のインストールとセットアップ
- ステップ 3: リポジトリデータベースを設定し、RDS for Db2 DB インスタンスに接続する
- IBM Db2 Data Management Consoleの使用

ステップ 1: DB インスタンスをモニタリングするためのリポジトリデータベースを作成する

既存の適切なサイズの RDS for Db2 DB インスタンスを IBM Db2 Data Management Console のリポジトリとして使用して、他の RDS for Db2 DB インスタンスをモニタリングできます。ただし、管理者ユーザーにはバッファプールとテーブルスペースを作成する SYSCTRL 権限がないため、IBM Db2 Data Management Console リポジトリの作成を使用してリポジトリデータベースを作成すると失敗します。代わりに、リポジトリデータベースを作成する必要があります。このリポジトリデータベースは、RDS for Db2 DB インスタンスをモニタリングします。

リポジトリデータベースは 2 とおりの方法で作成できます。RDS for Db2 データベースを作成し、バッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを手動で作成できます。または、IBM Db2 Data Management Console リポジトリデータベースをホストする別の Amazon EC2 インスタンスを作成することもできます。

トピック

- バッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを手動で作成する
- IBM Db2 Data Management Console リポジトリをホストする Amazon EC2 インスタンスを作成する

バッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを手動で作成する

バッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを作成するには

1. rdsadmin データベースに接続します。次の例では、*master_username* と *master_password* をユーザー自身の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. IBM Db2 Data Management Console のバッファプールを作成します。次の例では、*database_name* を、RDS for Db2 DB インスタンスをモニタリングするために IBM Db2 Data Management Console 用に作成したリポジトリの名前に置き換えます。

```
db2 "call rdsadmin.create_bufferpool('database_name',
     'BP4CONSOLE', 1000, 'Y', 'Y', 32768)"
```

3. IBM Db2 Data Management Console のユーザーテーブルスペースを作成します。次の例では、*database_name* を、RDS for Db2 DB インスタンスをモニタリングするために IBM Db2 Data Management Console 用に作成したリポジトリの名前に置き換えます。

```
db2 "call rdsadmin.create_tablespace('database_name',
     'TS4CONSOLE', 'BP4CONSOLE', 32768)"
```

4. IBM Db2 Data Management Console のシステム一時テーブルスペースを作成します。次の例では、*database_name* を、RDS for Db2 DB インスタンスをモニタリングするために IBM Db2 Data Management Console 用に作成したリポジトリの名前に置き換えます。

```
db2 "call rdsadmin.create_tablespace('database_name',
    'TS4CONSOLE_TEMP', 'BP4CONSOLE', 32768, 0, 0, 'S')"
```

これで IBM Db2 Data Management Console をインストールする準備ができました。インストールとセットアップの詳細については、「ステップ 2: IBM Db2 Data Management Console のインストールとセットアップ」を参照してください。

IBM Db2 Data Management Console リポジトリをホストする Amazon EC2 インスタンスを作成する

IBM Db2 Data Management Console リポジトリをホストする別の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成できます。Amazon EC2 インスタンスを作成する方法の詳細については、「Amazon EC2 ユーザーガイド」の「チュートリアル: Amazon EC2 Linux インスタンスの開始方法」を参照してください。

ステップ 2: IBM Db2 Data Management Console のインストールとセットアップ

バッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを作成したら、IBM Db2 Data Management Console をインストールしてセットアップする準備が整います。

> **Important**
>
> RDS for Db2 DB インスタンスと同じ VPC およびセキュリティグループにある Amazon EC2 Linux または Windows マシンが必要です。VPC とセキュリティグループは、内部ネットワークを介して DB インスタンスへの接続を制御します。また、IBM Db2 Data Management Console のリポジトリデータベースを既に作成している必要があります。

IBM Db2 Data Management Console をインストールしてセットアップするには

1. IBM Support ウェブサイトの IBM Db2 Data Management Console バージョン 3.1x リリースから IBM Db2 Data Management Console をダウンロードします。
2. IBM Db2 Data Management Console をインストールします。
3. IBM Db2 Data Management Console を開き、Amazon EC2 マシンの IP アドレスと Amazon EC2 インスタンスへの HTTP または HTTPS 接続に使用したポート番号を使用します。例えば、`http://xx.xx.xx.xx:11080` または `https://xx.xx.xx.xx.11081` を使用します。*xx.xx.xx.xx* を Amazon EC2 マシンの IP アドレスに置き換えます。`11080` と `11081` は、HTTP 接続と HTTPS 接続のデフォルトポートです。
4. (オプション) Amazon EC2 インスタンスでポート 80 または 443 を使用する場合は、Apache httpd または Nginx HTTP サーバーを使用して IBM Db2 Data Management Console ポートをポート 80 または 443 にプロキシできます。詳細については、「Apache HTTP Server プロジェクト」および「nginx ウェブサイト」を参照してください。

   IBM Db2 Data Management Console への接続を許可するには、セキュリティグループのインバウンドルールを編集する必要があります。プロキシを使用する場合は、TCP/IP ポート 80 また

は 443 を変更して、IBM Db2 Data Management Console ポートにリダイレクトします。プロキシを使用しない場合は、TCP/IP ポート 80 または 443 をデフォルトのポート 11080 (HTTP) または 11081 (HTTPS) に変更します。

これで、IBM Db2 Data Management Console にログインしてリポジトリデータベースを設定し、RDS for Db2 DB インスタンスに接続する準備が整いました。詳細については、「リポジトリデータベースの設定と DB インスタンスへの接続」を参照してください。

ステップ 3: リポジトリデータベースを設定し、RDS for Db2 DB インスタンスに接続する

リポジトリデータベースに初めて接続すると、IBM Db2 Data Management Console によってリポジトリが自動的に設定されます。リポジトリデータベースを設定したら、IBM Db2 Data Management Console にデータベースを接続できます。

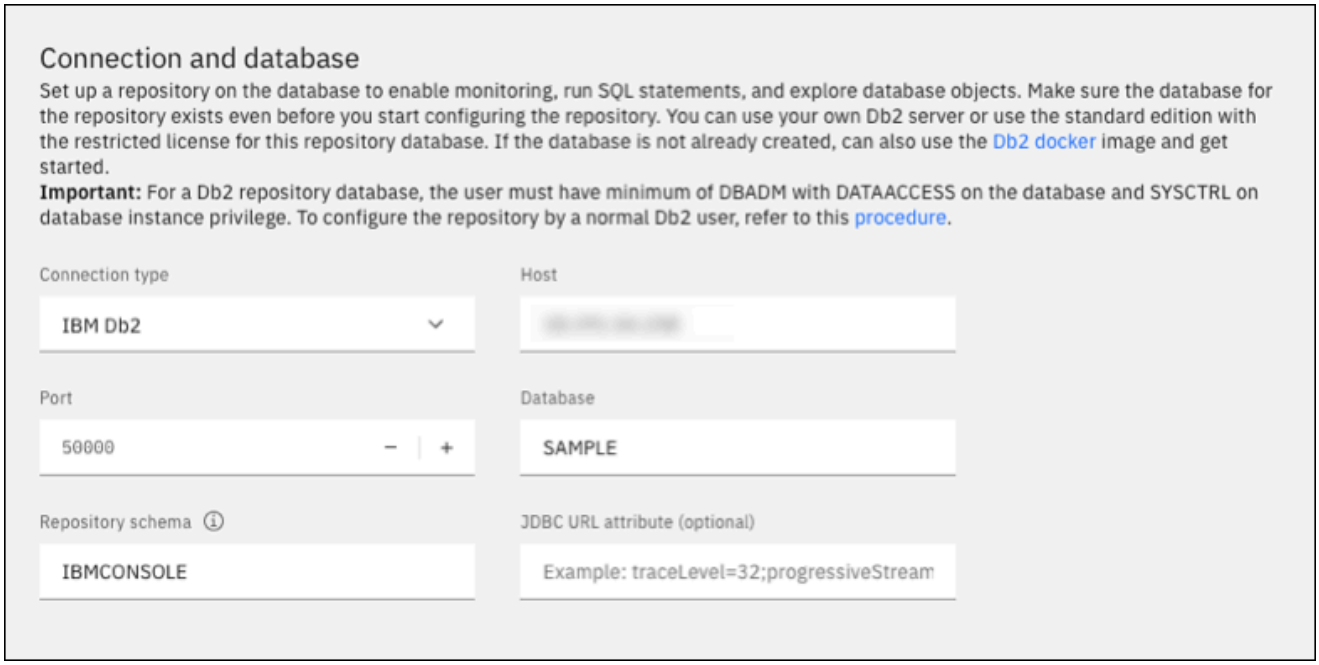
RDS for Db2 DB インスタンスに接続するには、DNS 名とポート番号が必要です。これを確認する方法の詳細については、「エンドポイントの検索」を参照してください。また、RDS for Db2 DB インスタンスの作成時に定義したデータベース名、マスターユーザー名、マスターパスワードも把握しておく必要があります。これを確認する方法の詳細については、「DB インスタンスの作成」を参照してください。インターネット経由で接続する場合は、データベースポートへのトラフィックを許可します。詳細については、「DB インスタンスの作成」を参照してください。

IBM Db2 Data Management Console を使用して RDS for Db2 DB インスタンスに接続するには

1. インストール時に設定した認証情報を使用して IBM Db2 Data Management Console にログインします。
2. リポジトリを設定します。

    a. [Connection and database] セクションで、RDS for Db2 DB インスタンスの次の情報を入力します。

    - [Host] に、DB インスタンスの DNS 名を入力します。
    - [Port] に、DB インスタンスのポート番号を入力します。
    - [Database] に、データベースの名前を入力します。



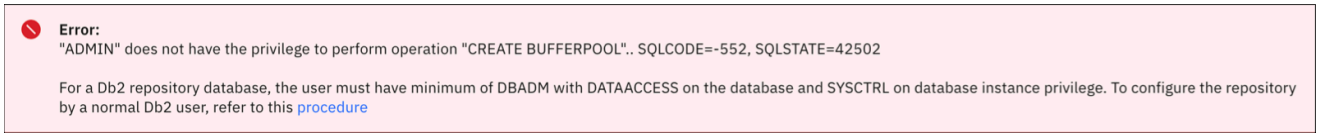
b. [Security and credential] セクションで、RDS for Db2 DB インスタンスの次の情報を入力します。

- [Security type] で、[Encrypted user and password] を選択します。
- [Username] に、DB インスタンスのデータベース管理者の名前を入力します。
- [Password] に、DB インスタンスのデータベース管理者のパスワードを入力します。

c. [Test connection] を選択します。

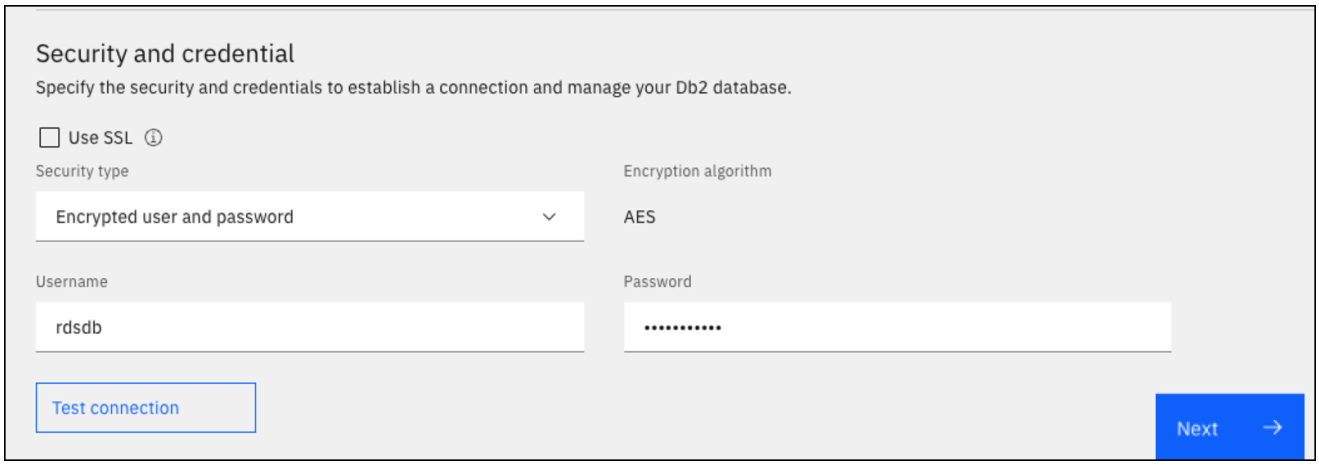
> **Note**
>
> 接続に失敗した場合は、セキュリティグループのインバウンドルールを見てデータベースポートが開いていることを確認します。詳細については、「Amazon RDS for Db2 のセキュリティグループの考慮事項」を参照してください。

RDS for Db2 でバッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを手動で作成しなかった場合、次のエラーメッセージが表示されることがあります。



RDS for Db2 DB インスタンスをモニタリングするための、IBM Db2 Data Management Console リポジトリのバッファテーブル、テーブルスペース、およびオブジェクトを作成したことを確認します。または、Amazon EC2 Db2 DB インスタンスを使用して IBM Db2 Data Management Console リポジトリをホストし、RDS for Db2 DB インスタンスをモニタリングすることもできます。詳細については、「ステップ 1: DB インスタンスをモニタリングするためのリポジトリデータベースを作成する」を参照してください。

d. 接続を正常にテストできたら、[Next] を選択します。

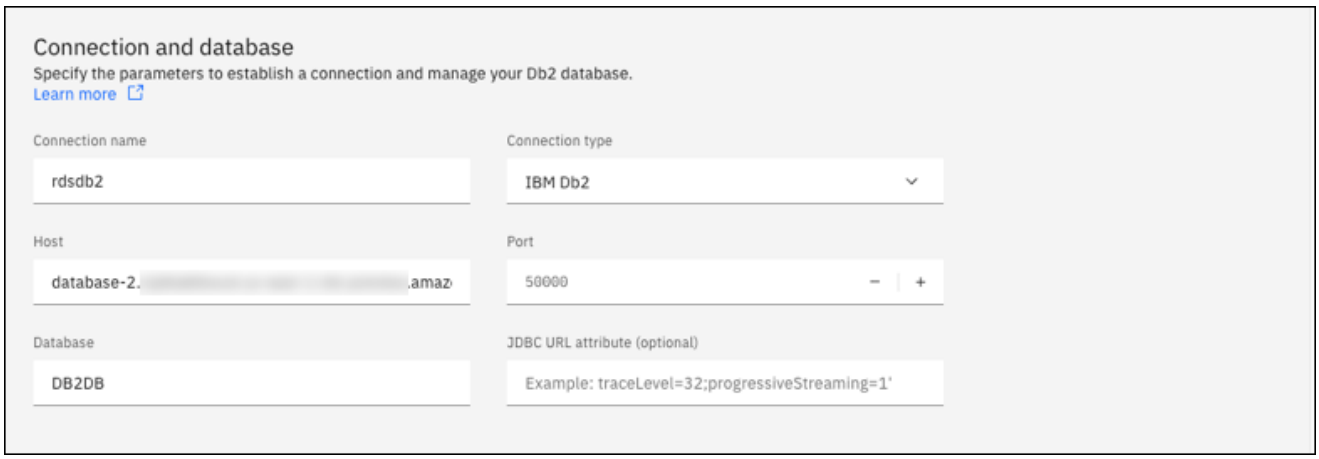


IBM Db2 Data Management Console が RDS for Db2 DB インスタンスでバッファプール、ユーザーテーブルスペース、およびシステム一時テーブルスペースを検出した場合、IBM Db2 Data Management Console はリポジトリデータベースを自動的に設定します。Amazon EC2 インスタンスで Db2 インスタンスをリポジトリデータベースとして使用すると、 IBM Db2 Data Management Console は自動的にバッファプールやその他のオブジェクトを作成します。

3. [Set statistics event monitor opt-in] ウィンドウで、[Next] を選択します。
4. (オプション) 新しい接続を追加します。管理とモニタリングに別の RDS for Db2 DB インスタンスを使用する場合は、非リポジトリ RDS for Db2 DB インスタンスへの接続を追加します。

   a. [Connection and database] セクションに、管理とモニタリングに使用する RDS for Db2 DB インスタンスの次の情報を入力します。

      - [Connection name]、Db2 データベース識別子を入力します。
      - [Host] に、DB インスタンスの DNS 名を入力します。
      - [Port] に、DB インスタンスのポート番号を入力します。
      - [Database] に、データベースの名前を入力します。



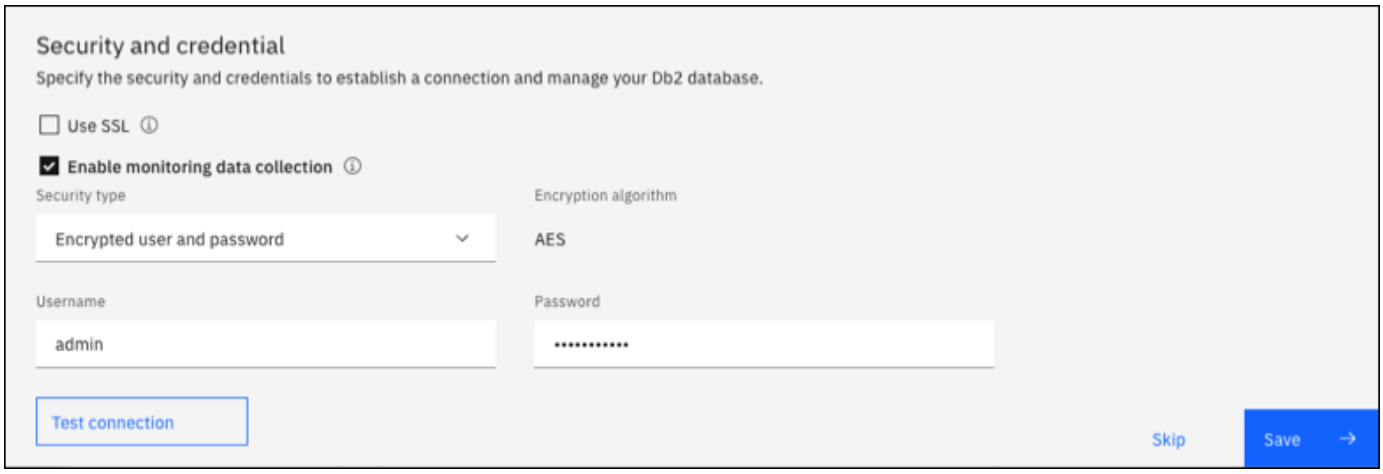
b. [Security and credential] セクションで、[Enable monitoring data collection] を選択します。

c. RDS for Db2 DB インスタンスの次の情報を入力します。

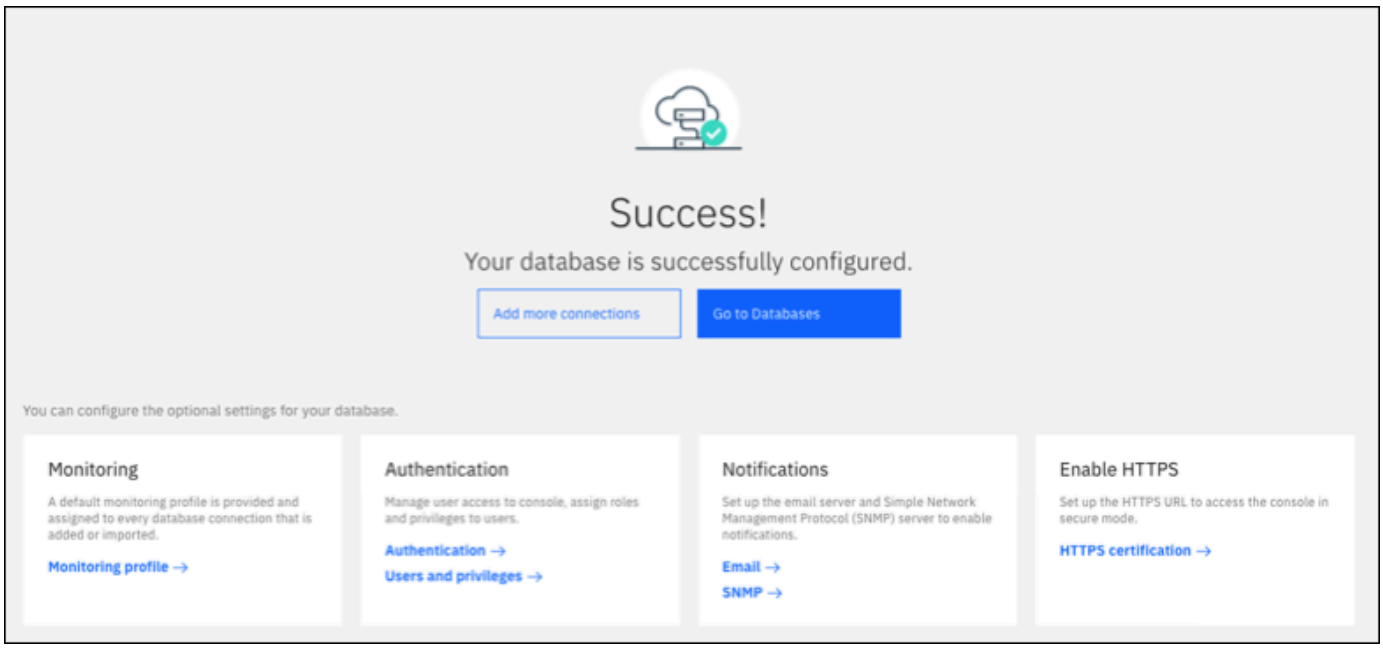
- [Username] に、DB インスタンスのデータベース管理者の名前を入力します。
- [Password] に、DB インスタンスのデータベース管理者のパスワードを入力します。

d. [Test connection] を選択します。

e. 接続が正常にテストされたら、[Save] を選択します。



接続が追加されると、次のようなウィンドウが表示されます。このウィンドウは、データベースが正常に設定されたことを示します。



5. [Go to Databases] を選択します。次のようなデータベースウィンドウが表示されます。このウィンドウは、メトリクス、ステータス、接続を示すダッシュボードです。

これで IBM Db2 Data Management Console の使用を開始できます。

IBM Db2 Data Management Consoleの使用

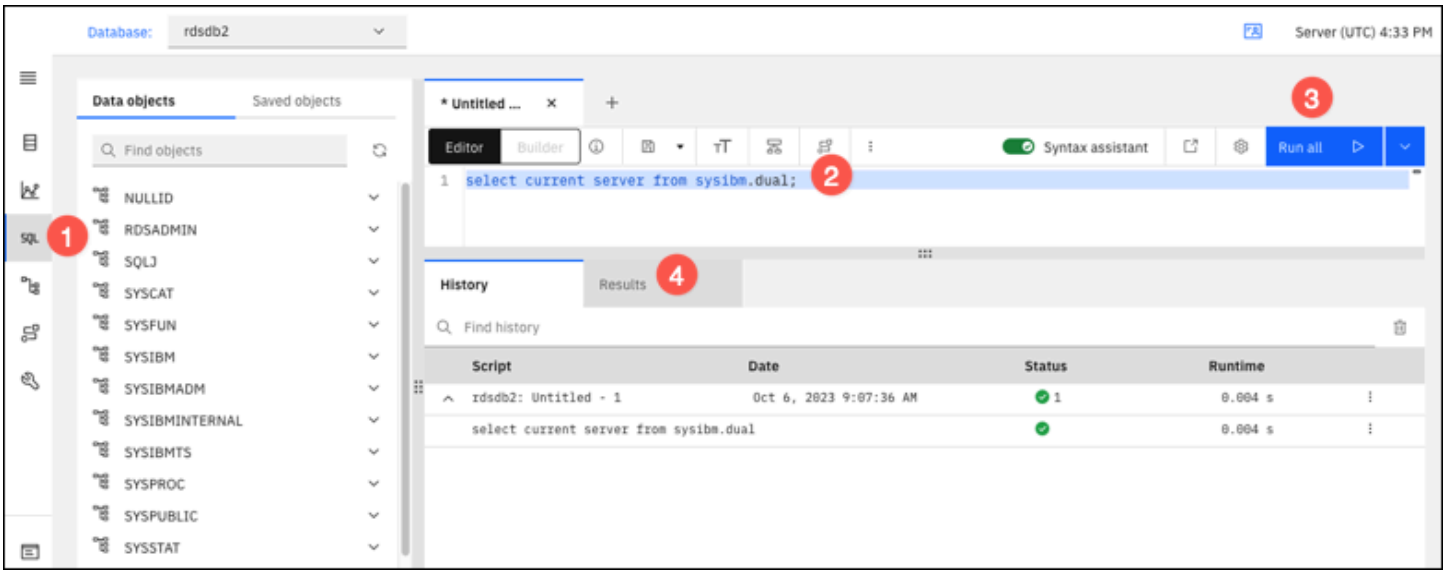
これで、IBM Db2 Data Management Console を使用して次のタイプのタスクを実行できます。

- 複数の RDS for Db2 DB を管理する。
- SQL コマンドを実行する。
- データオブジェクトとデータベースオブジェクトを探索、作成、または変更する。
- SQL で `EXPLAIN PLAN` ステートメントを作成する。

- クエリを調整する。

SQL コマンドを実行して結果を表示するには

1. 左側のナビゲーションバーで [SQL] を選択します。
2. SQL コマンドを入力します。
3. [Run all] を選択します。
4. 結果を表示するには、[Results] タブを選択します。



Amazon RDS for Db2 のセキュリティグループの考慮事項

Amazon RDS for Db2 DB インスタンスに接続するためには、必要な IP アドレスとネットワーク設定を含むセキュリティグループに関連付ける必要があります。RDS for Db2 DB インスタンスは、デフォルトのセキュリティグループを使用することがあります。RDS for Db2 DB インスタンスの作成時に、デフォルトの設定されていないセキュリティグループを割り当てた場合は、ファイアウォールによってインターネット接続が禁止されます。新しいセキュリティグループの作成方法については、セキュリティグループによるアクセス制御 を参照してください。

新しいセキュリティグループを作成したら、そのセキュリティグループと関連付けるように DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

SSL を使用して DB インスタンスへの接続を暗号化することで、セキュリティを高めることができます。詳細については、「Amazon RDS for Db2 DB インスタンスでの SSL/TLS の使用」を参照してください。

Amazon RDS for Db2 DB インスタンス接続の保護

Amazon RDS for Db2 は、RDS for Db2 DB インスタンスのセキュリティを向上させる方法をサポートしています。

トピック

- Amazon RDS for Db2 DB インスタンスでの SSL/TLS の使用
- RDS for Db2 による Active Directory の操作

Amazon RDS for Db2 DB インスタンスでの SSL/TLS の使用

SSL は、クライアントとサーバー間のネットワーク接続を安全に保つための業界標準のプロトコルです。SSL バージョン 3.0 以降は、名前が TLS と変更されていますが、いまだに SSL と呼称されることもよくあります。Amazon RDS は、Amazon RDS for Db2 DB インスタンス向けに SSL での暗号化をサポートしています。SSL/TLS を使用して、アプリケーションクライアントと RDS for Db2 DB インスタンス間の接続を暗号化できます。SSL/TLS サポートは、RDS for Db2 のすべての AWS リージョン で提供されています。

RDS for Db2 DB インスタンスの SSL/TLS 暗号化を有効するには、その DB インスタンスに関連付けられているパラメータグループに、Db2 SSL オプションを追加します。Db2 からの要求があるため、Amazon RDS では SSL/TLS 接続のために 2 番目のポートを使用しています。こうすることで、クリアテキストと SSL 暗号化の両方の通信を、DB インスタンスと Db2 クライアント間で同時に実行できます。例えば、このポートで SSL 暗号化通信を使用して VPC 外部のリソースと通信する一方で、このポートでクリアテキスト通信を使用して VPC 内の他のリソースと通信できます。

トピック

- SSL/TLS 接続の作成
- Db2 データベースサーバーに接続する

SSL/TLS 接続の作成

SSL/TLS 接続を作成するには、認証機関 (CA) を選択し、すべての AWS リージョンの証明書バンドルをダウンロードして、カスタムパラメータグループにパラメータを追加します。

ステップ 1: CA を選択して証明書をダウンロードする

認証機関 (CA) を選択し、すべての AWS リージョンの証明書バンドルをダウンロードします。詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

ステップ 2: カスタムパラメータグループのパラメータを更新する

> **Important**
>
> RDS for Db2 に Bring-Your-Own-License (BYOL) モデルを使用している場合は、IBM Customer ID と IBM Site ID 用に作成したカスタムパラメータグループを変更します。RDS for Db2 に別のライセンスモデルを使用している場合は、手順に従ってカスタムパラメータグループにパラメータを追加します。詳細については、「Amazon RDS for Db2 のライセンスオプション」を参照してください。

RDS for Db2 DB インスタンスのデフォルトのパラメータグループを変更することはできません。したがって、カスタムパラメータグループを作成して変更し、それを RDS for Db2 DB インスタンスにアタッチする必要があります。パラメータグループの詳細については、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。

次の表のパラメータ設定を使用します。

パラメータ	値
DB2COMM	TCPIP,SSL -または-SSL
SSL_SVCENAME	<any port number except the number used for the non-SSL port>

カスタムパラメータグループのパラメータを更新するには

1. create-db-parameter-group コマンドを実行して、カスタムパラメータグループを作成します。

   次の必須パラメータを含めます。

   - `--db-parameter-group-name` - 作成するパラメータグループの名前。

- `--db-parameter-group-family` - Db2 エンジンエディションとメジャーバージョン。有効な値: `db2-se-11-5`、`db2-ae-11.5`。
- `--description` - このパラメータグループの説明。

DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。

2. modify-db-parameter-group コマンドを実行して、作成したカスタムパラメータグループのパラメータを変更します。

   次の必須パラメータを含めます。

   - `--db-parameter-group-name` – 作成したパラメータグループの名前。
   - `--parameters` – パラメータ名、値、およびパラメータ更新用のアプリケーションメソッドの配列。

   パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

3. パラメータグループを RDS for Db2 DB インスタンスに関連付けます。詳細については、「Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける」を参照してください。

Db2 データベースサーバーに接続する

Db2 データベースサーバーへの接続手順は言語によって異なります。

Java

Java を使用して Db2 データベースサーバーに接続するには

1. JDBC ドライバーをダウンロードします。詳細については、IBM サポートドキュメントの「DB2 JDBC Driver Versions and Downloads」を参照してください。
2. 以下のコンテンツを使用して、シェルスクリプトファイルを作成します。このスクリプトによりは、バンドルのすべての証明書が Java KeyStore に追加されます。

> **Important**
>
> スクリプトが keytool を見つけることができるように、スクリプトのパス上に存在することを確認します。Db2 クライアントを使用する場合、 keytool は ~sqlib/java/jdk64/jre/bin にあります。

```
#!/bin/bash
PEM_FILE=$1
PASSWORD=$2
KEYSTORE=$3
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)
for N in $(seq 0 $(($CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cat $PEM_FILE |
    awk "n==$N { print }; /END CERTIFICATE/ { n++ }" |
    keytool -noprompt -import -trustcacerts -alias $ALIAS -keystore $KEYSTORE -
storepass $PASSWORD
done
```

3. シェルスクリプトを実行し、証明書バンドルを含む PEM ファイルを Java KeyStore にインポートするには、次のコマンドを実行します。*shell_file_name.sh* をシェルスクリプトファイルの名前に置き換え、*password* を Java KeyStore のパスワードに置き換えます。

```
./shell_file_name.sh global-bundle.pem password truststore.jks
```

4. Db2 サーバーに接続するには、次のコマンドを実行します。例の次のプレースホルダーを RDS for Db2 DB インスタンス情報に置き換えます。

   - *ip_address* – DB インスタンスエンドポイントの IP アドレス。
   - *port* – SSL 接続のポート番号。これは、SSL 以外のポートに使用される番号を除く任意のポート番号にすることができます。
   - *database_name* – DB インスタンス内のデータベースの名前。
   - *master_username* – DB インスタンスのマスターユーザー名。
   - *master_password* – DB インスタンスのマスターパスワード。

```
export trustStorePassword=MyPassword
java -cp ~/dsdriver/jdbc_sqlj_driver/linuxamd64/db2jcc4.jar \
com.ibm.db2.jcc.DB2Jcc -url \
"jdbc:db2://ip_address:port/database_name:\
sslConnection=true;sslTrustStoreLocation=\
~/truststore.jks;\
sslTrustStorePassword=${trustStorePassword};\
sslVersion=TLSv1.2;\
encryptionAlgorithm=2;\
securityMechanism=7;" \
-user master_username -password master_password
```

Node.js

Node.js を使用して Db2 データベースサーバーに接続するには

1. node-ibm_db ドライバーをインストールします。詳細については、 IBM Db2 ドキュメントの「Installing the node-ibm_db driver on Linux and UNIX systems」を参照してください。
2. 次の内容に基づいて JavaScript ファイルを作成します。例の次のプレースホルダーを RDS for Db2 DB インスタンス情報に置き換えます。

   - *ip_address* – DB インスタンスエンドポイントの IP アドレス。
   - *master_username* – DB インスタンスのマスターユーザー名。
   - *master_password* – DB インスタンスのマスターパスワード。
   - *database_name* – DB インスタンス内のデータベースの名前。
   - *port* – SSL 接続のポート番号。これは、SSL 以外のポートに使用される番号を除く任意のポート番号にすることができます。

```
var ibmdb = require("ibm_db");
const hostname = "ip_address";
const username = "master_username";
const password = "master_password";
const database = "database_name";
const port = "port";
const certPath = "/root/qa-bundle.pem";
```

```
ibmdb.open("DRIVER={DB2};DATABASE=" + database + ";HOSTNAME=" +
 hostname + ";UID=" + username + ";PWD=" + password + ";PORT=" + port +
 ";PROTOCOL=TCPIP;SECURITY=SSL;SSLServerCertificate=" + certPath + ";", function
 (err, conn){
 if (err) return console.log(err);
 conn.close(function () {
 console.log('done');
 });
});
```

3. JavaScript ファイルを実行するには、次のコマンドを実行します。

```
node ssl-test.js
```

Python

Python を使用して Db2 データベースサーバーに接続するには

1. 以下のコンテンツを含む Python ファイルを作成します。例の次のプレースホルダーを RDS for Db2 DB インスタンス情報に置き換えます。

   - *port* – SSL 接続のポート番号。これは、SSL 以外のポートに使用される番号を除く任意のポート番号にすることができます。
   - *master_username* – DB インスタンスのマスターユーザー名。
   - *master_password* – DB インスタンスのマスターパスワード。
   - *database_name* – DB インスタンス内のデータベースの名前。
   - *ip_address* – DB インスタンスエンドポイントの IP アドレス。

```
import click
import ibm_db
import sys

port = port;
master_user_id = "master_username" # Master id used to create your DB instance
master_password = "master_password" # Master password used to create your DB
 instance
db_name = "database_name" # If not given "db-name'
vpc_customer_private_ip = "ip_address" # Hosts end points - Customer private IP
 Addressicert_path = "/root/ssl/global-bundle.pem" # cert path
```

```
@click.command()
@click.option("--path", help="certificate path")
def db2_connect(path):

    try:
        conn =
 ibm_db.connect(f"DATABASE={db_name};HOSTNAME={vpc_customer_private_ip};PORT={port};

 PROTOCOL=TCPIP;UID={master_user_id};PWD={master_password};SECURITY=ssl;SSLServerCertifi
 "", "")
        try:
            ibm_db.exec_immediate(conn, 'create table tablename (a int);')
            print("Query executed successfully")
        except Exception as e:
            print(e)
        finally:
            ibm_db.close(conn)
            sys.exit(1)
    except Exception as ex:
        print("Trying to connect...")

if __name__ == "__main__":
    db2_connect()
```

2. 作成した Python ファイルを実行する次のシェルスクリプトを作成します。*python_file_name.py* を Python スクリプトファイルの名前に置き換えます。

```
#!/bin/bash
PEM_FILE=$1
# number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)

for N in $(seq 0 $(($CERTS - 1))); do
    ALIAS="${PEM_FILE%.*}-$N"
    cert=`cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }"`
    cat $PEM_FILE | awk "n==$N { print }; /END CERTIFICATE/ { n++ }" >
 $ALIAS.pem
    python3 <python_file_name.py> --path $ALIAS.pem
    output=`echo $?`
    if [ $output == 1 ]; then
        break
    fi
```

```
done
```

3. 証明書バンドルを含む PEM をインポートし、シェルスクリプトを実行するには、次のコマンドを実行します。*shell_file_name.sh* をシェルスクリプトファイルの名前に置き換えます。

```
./shell_file_name.sh global-bundle.pem
```

Db2 CLP

Db2 CLP を使用して Db2 データベースサーバーに接続するには

1. Db2 CLP を使用して Db2 インスタンスに接続するには、IBM Fix Central からダウンロードできる GSKit が必要です。Db2 CLP を使用するには、IBM Db2 クライアントも必要です。このクライアントは、IBM サポートの「Download initial Version 11.5 clients and drivers」からダウンロードできます。
2. キーストアを作成します。

```
gsk8capicmd_64 -keydb -create -db "directory/keystore-filename" -pw
 "changeThisPassword" -type pkcs12 -stash
```

3. 証明書のバンドルをキーストアにインポートします。

```
gsk8capicmd_64 -cert -import -file global-bundle.pem -target directory/keystore-
filename> -target_stashed
```

4. Db2 インスタンス設定を更新します。

```
db2 update dbm cfg using SSL_CLNT_KEYDB keystore-filename
 SSL_CLNT_STASH keystore stash file immediate
```

5. ノードとデータベースをカタログ化します。

```
db2 catalog tcpip node ssluse1 REMOTE endpoint SERVER ssl_svcename security ssl

db2 catalog database testdb as ssltest at node ssluse1
```

6. データベースに接続します。

```
db2 connect to sltest user username using password
```

RDS for Db2 による Active Directory の操作

RDS for Db2 DB インスタンスを Microsoft Active Directory (AD) ドメインに参加させることができます。AD ドメインは、AWS 内の AWS マネージド AD で、企業データセンターなど任意の場所のセルフマネージド AD で、AWS EC2 で、またはその他のクラウドプロバイダーでホストすることができます。

セルフマネージド Active Directory および AWS Managed Microsoft AD では、Kerberos 認証を使用してドメインユーザーを認証できます。

以下のセクションでは、Amazon RDS での Db2 について、セルフマネージド Active Directory と AWS マネージド Active Directory の使用方法を説明します。

トピック

- RDS for Db2 DB インスタンスによるセルフマネージド Active Directory の操作
- RDS for Db2 による AWS Managed Active Directory の操作

RDS for Db2 DB インスタンスによるセルフマネージド Active Directory の操作

RDS for Db2 は、セルフマネージド Active Directory (AD) ドメインと統合されます。AD は、データセンター、Amazon EC2、または他のクラウドプロバイダーでホストできます。この統合により、Kerberos プロトコルを介した直接ユーザー認証が可能になり、複雑な中間ドメインやフォレストの信頼性が不要になります。RDS for Db2 DB インスタンスに接続すると、RDS for Db2 は指定された AD ドメインに認証リクエストを安全に転送します。これにより、Amazon RDS マネージドデータベース機能を使用しながら、既存の ID 管理構造が維持されます。

利用可能なリージョンとバージョン

Amazon RDS は、すべての商用 AWS リージョンおよび AWS GovCloud (US) Regions で Kerberos を使用するセルフマネージド AD for Db2 をサポートしています。

RDS for Db2 を使用したセルフマネージド Active Directory の設定と管理については、以下のトピックを参照してください。

トピック

- 要件
- セルフマネージド Active Directory の設定
- セルフマネージド Active Directory ドメイン内の DB インスタンスの管理

要件

RDS for Db2 DB インスタンスをセルフマネージド AD ドメインに参加させる前に、次の要件を満たしていることを確認してください。

トピック

- オンプレミス AD の設定
- ネットワーク接続の設定
- AD ドメインサービスアカウントの設定

オンプレミス AD の設定

RDS for Db2 インスタンスを参加させることができるオンプレミスまたはその他のセルフマネージド Microsoft AD があることを確認してください。オンプレミス AD には以下の設定が必要です。

- AD サイトが定義されている場合、RDS for Db2 DB インスタンスに関連付けられている VPC 内のサブネットが AD サイトで定義されていることを確認してください。VPC 内のサブネットと他の AD サイトのサブネットの間に競合がないことを確認します。
- AD ドメインコントローラーは、Windows Server 2008 R2 以降のドメイン機能レベルを持ちます。
- AD の完全修飾ドメイン名 (FQDN) は 47 文字以内で指定します。

ネットワーク接続の設定

以下のネットワーク要件を満たしていることを確認してください。

- RDS for Db2 DB インスタンスを作成する Amazon VPC とセルフマネージド AD の間の接続を設定します。接続は、AWS 直接接続、AWS VPN、VPC ピアリング、または AWS トランジットゲートウェイを使用して設定できます。
- VPC セキュリティグループについては、デフォルトの VPC のデフォルトのセキュリティグループが、コンソールの RDS for Db2 DB インスタンスに既に追加されています。RDS for Db2 DB インスタンスを作成するサブネットのセキュリティグループと VPC ネットワーク ACL が、次の表に示すポートでトラフィックを許可していることを確認します。

  セルフマネージド AD に必要なポート

プロトコル	ポート	ロール
TCP / UDP	53	ドメインネームシステム (DNS)
TCP / UDP	88	Kerberos 認証
TCP / UDP	464	パスワードを変更 / 設定する
TCP / UDP	389	Lightweight Directory Access Protocol (LDAP)

- 通常、ドメイン DNS サーバーは AD ドメインコントローラーにあります。この機能を使用するために VPC DHCP オプションセットを設定する必要はありません。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。

> **Important**
>
> VPC ネットワーク ACL を使用している場合は、RDS for Db2 DB インスタンスからのダイナミックポート (49152-65535) でのアウトバウンドトラフィックも許可する必要があります。これらのトラフィックルールが、AD ドメインコントローラー、DNS サーバー、RDS for Db2 DB インスタンスのそれぞれに適用されるファイアウォールにもミラーリングされていることを確認します。
> VPC セキュリティグループでは、ネットワークトラフィックが発生する方向でのみポートを開く必要があります。ただし、ほとんどの Windows ファイアウォールと VPC ネットワーク ACL では、両方向にポートを開く必要があります。

AD ドメインサービスアカウントの設定

AD ドメインサービスアカウントが次の要件を満たしていることを確認してください。

- セルフマネージド AD ドメインに、専用の組織単位 (OU) でユーザーオブジェクトを作成および管理するための委任されたアクセス許可を持つドメインサービスアカウントがあることを確認します。ドメインサービスアカウントは、特定のタスクを実行するアクセス許可を委任されたセルフマネージド AD のユーザーアカウントです。
- ドメインサービスアカウントには、RDS for Db2 DB インスタンスを参加させる OU の以下のアクセス許可を委任する必要があります。
  - ユーザーオブジェクトを作成および削除する
  - パスワードのリセット
  - `msDS-SupportedEncryptionTypes` の読み取りと書き込み
  - `servicePrincipalName` の読み取りと書き込み

  これらのアクセス許可を設定するステップバイステップの手順については、「ステップ 3: AD ドメインサービスアカウントに制御を委任する」を参照してください。

> **Important**
>
> DB インスタンスの作成後に RDS for Db2 が組織単位に作成したユーザーオブジェクトを移動しないでください。関連付けられたオブジェクトを移動すると、RDS for Db2 DB インスタンスの設定が正しくなくなります。Amazon RDS によって作成されたユーザーオブジェ

クトを移動する必要がある場合は、ModifyDBInstance API オペレーションを使用して、ユーザーオブジェクトの目的の場所にドメインパラメータを変更します。

セルフマネージド Active Directory の設定

セルフマネージド AD を設定するには、次の手順を実行してください。

トピック

- ステップ 1: AD に組織単位を作成する
- ステップ 2: AD に AD ドメインサービスアカウントを作成する
- ステップ 3: AD ドメインサービスアカウントに制御を委任する
- ステップ 4: AWS KMS キーを作成する
- ステップ 5: AWS シークレットを作成する
- ステップ 6: Db2 DB インスタンスを作成または変更する

ステップ 1: AD に組織単位を作成する

> **Important**
>
> セルフマネージド AD ドメインに参加した RDS for Db2 DB インスタンスを所有する AWS アカウントに、専用の OU とその OU を対象とするサービス認証情報を作成することをお勧めします。OU とサービス認証情報を専用にすることで、アクセス許可の競合を回避し、最小特権の原則に従うことができます。

AD に OU を作成するには

1. ドメイン管理者として AD ドメインに接続します。
2. [Active Directory ユーザーとコンピューター] を開き、OU を作成するドメインを選択します。
3. ドメインのコンテキストメニューを開き、[新規] を選択し、[組織単位] を選択します。
4. OU の名前を入力します。
5. [コンテナを誤って削除しないように保護する] ボックスはオンのままにしてください。
6. [OK] を選択してください。新しい OU がドメインの下に表示されます。

ステップ 2: AD に AD ドメインサービスアカウントを作成する

ドメインサービスアカウントの認証情報は AWS Secrets Manager のシークレットに使用されます。

AD に AD ドメインサービスアカウントを作成するには

1. [Active Directory ユーザーとコンピューター] を開き、ユーザーを作成するドメインと OU を選択します。
2. [ユーザー] のコンテキストメニューを開き、[新規]、[ユーザー] の順に選択します。
3. ユーザーの名、姓、およびログオン名を入力します。[次へ] を選択します。
4. ユーザーのパスワードを入力します。[ユーザーは次回のログイン時にパスワードを変更する必要がある] を選択しないでください。[アカウントは無効です] を選択しないでください。[次へ] を選択します。
5. [OK] を選択してください。新しいユーザーがドメインの下に表示されます。

ステップ 3: AD ドメインサービスアカウントに制御を委任する

パート A: コントロールの委任ウィザードを使用してアクセス許可を委任する

ドメイン内の AD ドメインサービスアカウントに制御を委任するには

1. [Active Directory ユーザーとコンピュータ] Microsoft 管理コンソール (MMC) スナップインを開き、ユーザーを作成するドメインを選択します。
2. 前に作成した OU のコンテキストメニューを開き、[コントロールの委任] を選択します。
3. [コントロールの委任ウィザード] で、[次へ] を選択します。
4. [ユーザーまたはグループ] セクションで、[追加] を選択します。
5. [ユーザー、コンピュータ、またはグループの選択] セクションで、作成した AD ドメインサービスアカウントを入力し、[名前を確認] を選択します。AD ドメインサービスアカウントのチェックが成功したら、[OK] を選択します。
6. [ユーザーまたはグループ] セクションで、AD ドメインサービスアカウントが追加されたことを確認し、[次へ] を選択します。
7. [委任するタスク] セクションで、[委任するカスタムタスクを作成] を選択し、[次へ] を選択します。
8. [Active Directory オブジェクトタイプ] セクションで:

    a. [フォルダ内の次のオブジェクトのみ] を選択します。

b. [ユーザーオブジェクト] を選択します。
c. [このフォルダに選択したオブジェクトを作成] を選択します。
d. [このフォルダ内の選択したオブジェクトを削除] を選択し、[次へ] を選択します。

9. [アクセス許可] セクションで:

a. [全般] を選択したままにします。
b. [プロパティ固有] を選択します。
c. [特定の子オブジェクトの作成/削除] を選択します。
d. [パスワードのリセット] を選択します。
e. [msDS-SupportedEncryptionTypes の読み取り] を選択します。
f. [msDS-SupportedEncryptionTypes の書き込み] を選択します。

10. [コントロールの委任ウィザードの完了] で設定を確認し、[完了] を選択します。

パート B: PowerShell を使用して servicePrincipalName アクセス許可を委任する

コントロールの委任ウィザードは、ユーザーオブジェクトの `servicePrincipalName` プロパティを公開しません。ドメインコントローラーで次の PowerShell スクリプトを実行し、AD ドメインサービスアカウントに OU のユーザーオブジェクトの `servicePrincipalName` に対する読み取りおよび書き込みアクセスを許可します。

`service-account` をステップ 2 で作成した AD ドメインサービスアカウントの名前に置き換え、ステップ 1 で作成した OU と一致するように OU 識別名を更新します。

```
$ou = "OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain"
$user = Get-ADUser "service-account"
$acl = Get-Acl "AD:\$ou"

$schemaPath = (Get-ADRootDSE).schemaNamingContext
$spnSchema = Get-ADObject -SearchBase $schemaPath -Filter {lDAPDisplayName -eq
 "servicePrincipalName"} -Properties schemaIDGUID
$spnGuid = New-Object Guid (,$spnSchema.schemaIDGUID)
$userGuid = New-Object Guid "bf967aba-0de6-11d0-a285-00aa003049e2"

$ace = New-Object System.DirectoryServices.ActiveDirectoryAccessRule(
   $user.SID,
   "ReadProperty,WriteProperty",
   "Allow",
   $spnGuid,
```

```
    "Descendents",
    $userGuid
)

$acl.AddAccessRule($ace)
Set-Acl "AD:\$ou" $acl
```

アクセス許可が正しく適用されていることを確認するには、次の PowerShell スクリプトを実行します。

```
$acl = Get-Acl "AD:\$ou"
$acl.Access | Where-Object { $_.IdentityReference -like "*service-account*" } | Format-
List
```

ステップ 4: AWS KMS キーを作成する

KMS キーは、AWS シークレットを暗号化します。

> **Note**
>
> [暗号化キー] として、AWS デフォルトの KMS キーを使用しないでください。AWS KMS キーは、セルフマネージド AD に参加させる RDS for Db2 DB インスタンスを含んでいるのと同じ AWS アカウントに作成してください。

AWS KMS キーを作成するには

1. https://console.aws.amazon.com/kms/ で AWS KMS コンソールを開きます。[Create key] (キーの作成) を選択します。
2. [キーの種類] として、[対称] を選択します。
3. [キーの使用方法] として、[暗号化と復号化] を選択します。
4. [Advanced options (詳細オプション)] の場合:
   a. [キーマテリアルのオリジン] として、[KMS] を選択します。
   b. [リージョナリティ] では、[単一リージョンキー] を選択し、[次へ] を選択します。
5. [エイリアス] に、KMS キーの名前を指定します。
6. (オプション) [説明] に、KMS キーの説明を入力します。

7. (オプション) タグで、KMS キーのタグを指定し、次へを選択します。
8. [キー管理者] として、IAM ユーザーの名前を入力して選択します。
9. [キーの削除] で、[キー管理者にこのキーの削除を許可する] のボックスをオンのままにして、[次へ] を選択します。
10. [キーユーザー] として、前のステップと同じ IAM ユーザーを指定して選択します。[次へ] を選択します。
11. 設定を確認します。
12. [キーポリシー] で、以下をポリシー [ステートメント] に含めます。

```
{
    "Sid": "Allow use of the KMS key on behalf of RDS",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "rds.amazonaws.com"
        ]
    },
    "Action": "kms:Decrypt",
    "Resource": "*"
}
```

13. [Finish] を選択してください。

ステップ 5: AWS シークレットを作成する

> **Note**
>
> シークレットは、セルフマネージド AD に参加させる RDS for Db2 DB インスタンスを含んでいるのと同じ AWS アカウントに作成してください。

シークレットを作成する

1. https://console.aws.amazon.com/secretsmanager/ から AWS Secrets Manager コンソールを開きます。新しいシークレットを保存 を選択します。
2. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
3. [キーと値のペア] として、次の 2 つのキーを追加します。

a. 最初のキーには、SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME と入力します。
b. 最初のキーの値には、AD ユーザーのユーザー名 (ドメインプレフィックスなし) のみを入力します。ドメイン名を含めないでください。含めると、インスタンスの作成が失敗します。
c. 2 番目のキーとして、SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD と入力します。
d. 2 番目のキーの値には、ドメインの AD ユーザー用に作成したパスワードを入力します。

4. [暗号化キー] で、前のステップで作成した KMS キーを入力し、[次へ] を選択します。
5. [シークレット名] として、後でシークレットを見つけやすい、わかりやすい名前を入力します。
6. (オプション) [説明] として、シークレット名の説明を入力します。
7. リソース権限については、編集を選択します。
8. 以下のポリシーをアクセス許可ポリシーに追加します。

> **Note**
>
> Confused Deputy Problem (混乱した代理の問題) を回避するために、ポリシーの aws:sourceAccount および aws:sourceArn 条件を使用することをお勧めします。aws:sourceAccount の AWS アカウントと aws:sourceArn の RDS for Db2 DB インスタンス ARN を使用します。詳細については、「サービス間での混乱した代理問題の防止」を参照してください。

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Principal":
            {
                "Service": "rds.amazonaws.com"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*",
            "Condition":
            {
                "StringEquals":
                {
                    "aws:sourceAccount": "123456789012"
```

```
                },
                "ArnLike":
                {
                    "aws:sourceArn": "arn:aws:rds:us-west-2:123456789012:db:*"
                }
            }
        }
    ]
}
```

9. [保存]、[次へ] の順に選択します。
10. [ローテーションの設定] は、デフォルト値のままにして、[次へ] を選択します。
11. シークレットの設定を確認し、[保存] を選択します。
12. 作成したシークレットを選択し、[シークレット ARN] の値をコピーします。この値を次のステップで使用して、セルフマネージド Active Directory をセットアップします。

ステップ 6: Db2 DB インスタンスを作成または変更する

AWS CLI を使用して、RDS for Db2 DB インスタンスをセルフマネージド AD ドメインに関連付けることができます。これは、次のいずれかの方法で行うことができます。

- create-db-instance CLI コマンドを使用して、新しい Db2 DB インスタンスを作成します。手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- modify-db-instance CLI コマンドを使用して、既存の Db2 DB インスタンスを変更します。手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI を使用する場合は、DB インスタンスが、作成したセルフマネージド AD ドメインを使用できるように、以下のパラメータが必要です。

- `--domain-fqdn` パラメータには、セルフマネージド AD の完全修飾ドメイン名 (FQDN) を使用します。
- `--domain-ou` パラメータには、セルフマネージド AD で作成した OU を使用します。
- `--domain-auth-secret-arn` パラメータには、前のステップで作成したシークレット ARN の値を使用します。
- `--domain-dns-ips` パラメータには、セルフマネージド AD の DNS サーバーのプライマリ IPv4 アドレスとセカンダリ IPv4 アドレスを使用します。セカンダリ DNS サーバーの IP アドレスがない場合は、プライマリ IP アドレスを 2 回入力します。

次の CLI コマンドは、新しい RDS for Db2 DB インスタンスを作成し、それをセルフマネージド AD ドメインに参加させます。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier my-DB-instance \
    --db-instance-class db.m6i.large \
    --allocated-storage 50 \
    --engine db2-se \
    --engine-version 11.5 \
    --master-username my-master-username \
    --master-user-password my-master-password \
    --domain-fqdn my_AD_domain.my_AD.my_domain \
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier my-DB-instance ^
    --db-instance-class db.m6i.large ^
    --allocated-storage 50 ^
    --engine db2-se ^
    --engine-version 11.5 ^
    --master-username my-master-username ^
    --master-user-password my-master-password ^
    --domain-fqdn my_AD_domain.my_AD.my_domain ^
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" ^
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

セルフマネージド Active Directory ドメイン内の DB インスタンスの管理

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、DB インスタンスおよびセルフマネージド AD ドメインとの関係を管理できます。例えば、DB インスタンスをドメイン内、ドメイン外、またはドメイン間で移動させることができます。

Amazon RDS API を使用して次を実行できます。

- 失敗したメンバーシップのセルフマネージドドメインへの参加を再試行するには、ModifyDBInstance API オペレーションを使用して、同じパラメータセットを指定します。
  - `--domain-fqdn`
  - `--domain-dns-ips`
  - `--domain-ou`
  - `--domain-auth-secret-arn`
- セルフマネージドドメインから DB インスタンスを削除するには、`ModifyDBInstance` API オペレーションを使用し、ドメインパラメータとして `--disable-domain` を指定します。
- DB インスタンスを別のセルフマネージドドメインに移動するには、`ModifyDBInstance` API オペレーションを使用し、新しいドメインのドメインパラメータを指定します。
  - `--domain-fqdn`
  - `--domain-dns-ips`
  - `--domain-ou`
  - `--domain-auth-secret-arn`
- 各 DB インスタンスのセルフマネージド AD ドメインメンバーシップを一覧表示するには、DescribeDBInstances API オペレーションを使用します。

セルフマネージド Active Directory ドメインメンバーシップについて

AD の詳細を指定しながら DB インスタンスを作成または変更した後、そのインスタンスはセルフマネージド AD ドメインのメンバーになります。AWS コンソールは、DB インスタンスについて、セルフマネージド Active Directory ドメインメンバーシップのステータスを示します。DB インスタンスのステータスは、以下のいずれかです。

- joined – インスタンスは AD ドメインのメンバーです。
- Joining – インスタンスは、AD ドメインのメンバーになる途中です。
- pending-join (参加保留中) – インスタンスのメンバーシップは保留中です。
- pending-maintenance-join – AWS は、次に予定されているメンテナンスウィンドウ中に、インスタンスを AD ドメインのメンバーにできるよう試みます。
- pending-removal – AD ドメインからのインスタンスの削除は保留中です。
- pending-maintenance-removal – AWS は、次に予定されているメンテナンスウィンドウ中に、AD ドメインからのインスタンスの削除を試みます。

- failed – 設定の問題により、インスタンスは AD ドメインに参加できませんでした。インスタンスの変更コマンドを再発行する前に、設定を確認して修正してください。
- removing – インスタンスをセルフマネージド AD ドメインから削除しています。

> **Important**
>
> セルフマネージド AD ドメインのメンバーになるリクエストは、ネットワーク接続の問題が原因で失敗する場合があります。例えば、DB インスタンスを作成したか、既存のインスタンスを変更したが、DB インスタンスをセルフマネージド AD ドメインのメンバーにする試みが失敗することがあります。この場合、コマンドを再発行して DB インスタンスを作成または変更するか、新しく作成されたインスタンスを変更して、セルフマネージド AD ドメインに参加させます。

RDS for Db2 による AWS Managed Active Directory の操作

ユーザーが Amazon RDS for Db2 DB インスタンスに接続する際、Kerberos 認証を使用してユーザーを認証できます。この設定では、DB インスタンスは AWS Directory Service for Microsoft Active Directory と連携します (AWS Managed Microsoft AD とも言う)。次に、AWS Managed Microsoft AD ディレクトリのドメインおよびその他の情報を RDS for Db2 DB インスタンスに追加します。ユーザーが、信頼性の高いドメインに接続された RDS for Db2 DB インスタンスを使用して認証すると、認証リクエストは Directory Service で作成した AWS Managed Microsoft AD ディレクトリに転送されます。

同じディレクトリにすべての認証情報を保持することで時間と労力を節約できます。この方法により、複数の DB インスタンスの認証情報を一元的に保存および管理できます。また、ディレクトリを使用することで、セキュリティプロファイル全体を向上できます。

また、独自のオンプレミスの Microsoft Active Directory から認証情報にアクセスできます。そのためには、信頼するドメイン関係を作成して、AWS Managed Microsoft AD ディレクトリがオンプレミスの Microsoft Active Directory を信頼するようにします。これにより、ユーザーは、オンプレミスネットワークのワークロードにアクセスするときと同じ Windows シングルサインオン (SSO) を使用して、RDS for Db2 DB インスタンスにアクセスできます。

詳細については、「AWS Directory Service 管理ガイド」の「Directory Service とは」を参照してください。

Kerberos 認証の詳細については、以下のトピックを参照してください。

トピック

- Amazon RDS for Db2 DB インスタンスの Kerberos 認証を設定する
- Kerberos 認証を使用して Amazon RDS for Db2 に接続する

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。Kerberos 認証を使用した RDS for Db2 のバージョンとリージョンの可用性の詳細については、「Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン」を参照してください。

> **Note**
>
> Kerberos 認証は、RDS for Db2 DB インスタンスで廃止された DB インスタンスクラスではサポートされていません。詳細については、「Amazon RDS for Db2 インスタンスクラス」を参照してください。

RDS for Db2 DB インスタンスの Kerberos 認証の概要

RDS for Db2 DB インスタンスに Kerberos 認証を設定するには、以下の一般的なステップを完了します。詳細は後で説明します。

1. AWS Managed Microsoft AD を使用して AWS Managed Microsoft AD ディレクトリを作成します。AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI)、または Directory Service を使用して、ディレクトリを作成できます。詳細については、「AWS Directory Service 管理ガイド」の「AWS Managed Microsoft AD ディレクトリの作成」を参照してください。
2. マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する AWS Identity and Access Management (IAM) ロールの作成 この IAM ロールにより Amazon RDS はディレクトリを呼び出すことができます。

   IAM ロールによるアクセスを許可するには、AWS Security Token Service (AWS STS) エンドポイントを AWS アカウント の正しい AWS リージョン でアクティベートする必要があります。AWS STS エンドポイントはすべての AWS リージョン でデフォルトでアクティブになっているため、他のアクションを実行せずに、エンドポイントを使用することができます。詳細については、「IAM ユーザーガイド」の「AWS リージョンでの AWS STS のアクティブ化と非アクティブ化」を参照してください。
3. 以下のいずれかの方法で、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、RDS for Db2 DB インスタンスを作成または変更します。
   - コンソール、create-db-instance コマンド、または CreateDBInstance API オペレーションを使用して新しい RDS for Db2 DB インスタンスを作成します。手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - コンソール、modify-db-instance コマンド、または ModifyDBInstance API オペレーションを使用して既存の RDS for Db2 DB インスタンスを変更します。手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
   - コンソール、restore-db-instance-from-db-snapshot コマンド、または RestoreDBInstanceFromDBSnapshot API オペレーションを使用して、DB スナップショットか

ら RDS for Db2 DB インスタンスを復元します。手順については、「DB インスタンスへの復元」を参照してください。

- コンソール、restore-db-instance-to-point-in-time コマンド、または RestoreDBInstanceToPointInTime API オペレーションを使用して、RDS for Db2 DB インスタンスを特定時点に復元します。手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

DB インスタンスは、ディレクトリと同じ Amazon Virtual Private Cloud (VPC) か、別の AWS アカウント または VPC にあります。RDS for Db2 DB インスタンスの作成または変更時に、次の手順を行います。

- ディレクトリの作成時に、生成されたドメイン識別子 (d-* 識別子) を指定します。
- 作成した IAM ロール名を指定します。
- DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからインバウンドトラフィックを受信できることを確認します。

4. Db2 クライアントを設定し、次のポートでクライアントホストと Directory Service の間でトラフィックが流れることを確認します。
   - TCP/UDP ポート 53 – DNS
   - TCP 88 – Kerberos 認証
   - TCP 389 – LDAP
   - TCP 464 – Kerberos 認証

ドメインの DB インスタンスの管理

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB インスタンスおよび Microsoft Active Directory との関係を管理できます。例えば、Kerberos 認証を有効化するために、Active Directory を関連付けることができます。また、Active Directory の関連付けを解除して、Kerberos 認証を無効化することもできます。さらに、DB インスタンスを外部認証する Microsoft Active Directory を別の Active Directory に変更することもできます。

例えば、modify-db-instance CLI コマンドを実行して、次のアクションを実行できます。

- `--domain` オプションに現在のメンバーシップのディレクトリ ID を指定して、失敗したメンバーシップに対する Kerberos 認証の有効化を再試行します。
- `--domain` オプションに `none` を指定して、DB インスタンスの Kerberos 認証を無効にします。
- `--domain` オプションに新しいドメインのドメイン識別子を指定して、DB インスタンスを 1 つのドメインから別のドメインに移動します。

ドメインのメンバーシップを理解する

DB インスタンスを作成または変更すると、そのインスタンスはドメインのメンバーになります。ドメインメンバーシップのステータスは、コンソールで、または describe-db-instances コマンドを実行して表示できます。DB インスタンスのステータスは、以下のいずれかです。

- `kerberos-enabled` - DB インスタンスで Kerberos 認証が有効化されました。
- `enabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を有効化中です。
- `pending-enable-kerberos` - この DB インスタンスでの Kerberos 認証の有効化が保留中です。
- `pending-maintenance-enable-kerberos` - AWS は、次の予定メンテナンスウィンドウ中に、DB インスタンスでの Kerberos 認証の有効化を試みます。
- `pending-disable-kerberos` - この DB インスタンスでの Kerberos 認証の無効化が保留中です。
- `pending-maintenance-disable-kerberos` - AWS は、次の予定メンテナンスウィンドウ中に、DB インスタンスでの Kerberos 認証の無効化を試みます。
- `enable-kerberos-failed` - 設定の問題により、AWS は DB インスタンスで Kerberos 認証を有効化できませんでした。DB インスタンスを変更するコマンドを再発行する前に、設定の問題を修正します。
- `disabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を無効化中です。

ネットワーク接続の問題や正しくない IAM ロールのために、Kerberos 認証の有効化リクエストは失敗する可能性があります。場合によっては、DB インスタンスを作成または変更するときに、Kerberos 認証を有効にしようとすると失敗する可能性があります。その場合、正しい IAM ロールを使用していることを確認してから、DB インスタンスを変更し、ドメインに接続します。

Amazon RDS for Db2 DB インスタンスの Kerberos 認証を設定する

AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) を使用して、RDS for Db2 DB インスタンスの Kerberos 認証を設定します。Kerberos 認証を設定するには、次の手順に従います。

トピック

- ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する
- ステップ 2: 信頼関係を作成する

- ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する
- ステップ 4: ユーザーを作成して設定する
- ステップ 5: AWS Managed Microsoft AD で RDS for Db2 管理者グループを作成する
- ステップ 6: DB パラメータを変更する
- ステップ 7: RDS for Db2 DB インスタンスを作成または変更する
- ステップ 8: PowerShell で Active Directory グループ SID を取得する
- ステップ 9: RDS for Db2 DB インスタンスに SID と GroupName のマッピングを追加する
- ステップ 10: Db2 クライアントを設定する

ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する

Directory Service は、Active Directory でフルマネージド AWS クラウド を作成します。AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service が 2 つのドメインコントローラーと DNS サーバーを作成します。ディレクトリサーバーは、VPC 内の異なるサブネットで作成されます。この冗長性により、障害が発生してもディレクトリがアクセスできるようになります。

AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって自動的に以下のタスクを実行します。

- VPC 内で Active Directory を設定します。
- ユーザー名 `Admin` と指定されたパスワードで、ディレクトリ管理者アカウントを作成します。このアカウントを使用してディレクトリを管理します。

> **Important**
>
> このパスワードは必ず保管してください。Directory Service にはこのパスワードは保存されず、復元やリセットもできません。

- ディレクトリコントローラー用セキュリティグループを作成します。セキュリティグループは、RDS for Db2 DB インスタンスとの通信を許可する必要があります。

AWS Directory Service for Microsoft Active Directory を起動すると、AWS は組織単位 (OU) を作成します。OU にはディレクトリのオブジェクトがすべて含まれています。この OU はドメインルートにあります。OU にはディレクトリを作成する際に入力した NetBIOS 名があります。ドメインルートは AWS が所有し、管理します。

Admin ディレクトリに作成された AWS Managed Microsoft AD アカウントには、OU に対して頻繁に実行される管理行為の権限が含まれています。

- ユーザーを作成、更新、削除する。
- ファイルやプリントサーバーなどのドメインにリソースを追加して、追加したリソースへのアクセス許可を OU のユーザーとグループに割り当てる。
- 追加の OU やコンテナを作成する。
- 権限を委譲する。
- 削除したオブジェクトを Active Directory のごみ箱から元に戻す。
- Directory Service で Windows PowerShell 向けの Active Directory およびドメインネームサービス (DNS) モジュールを実行する。

Admin アカウントには、ドメイン全体に関係するアクティビティを実行する権限もあります。

- DNS 設定 (レコード、ゾーン、フォワーダーの追加、削除、または更新) を管理する。
- DNS イベントログを参照する。
- セキュリティイベントログを参照する。

AWS Managed Microsoft AD でディレクトリを作成するには

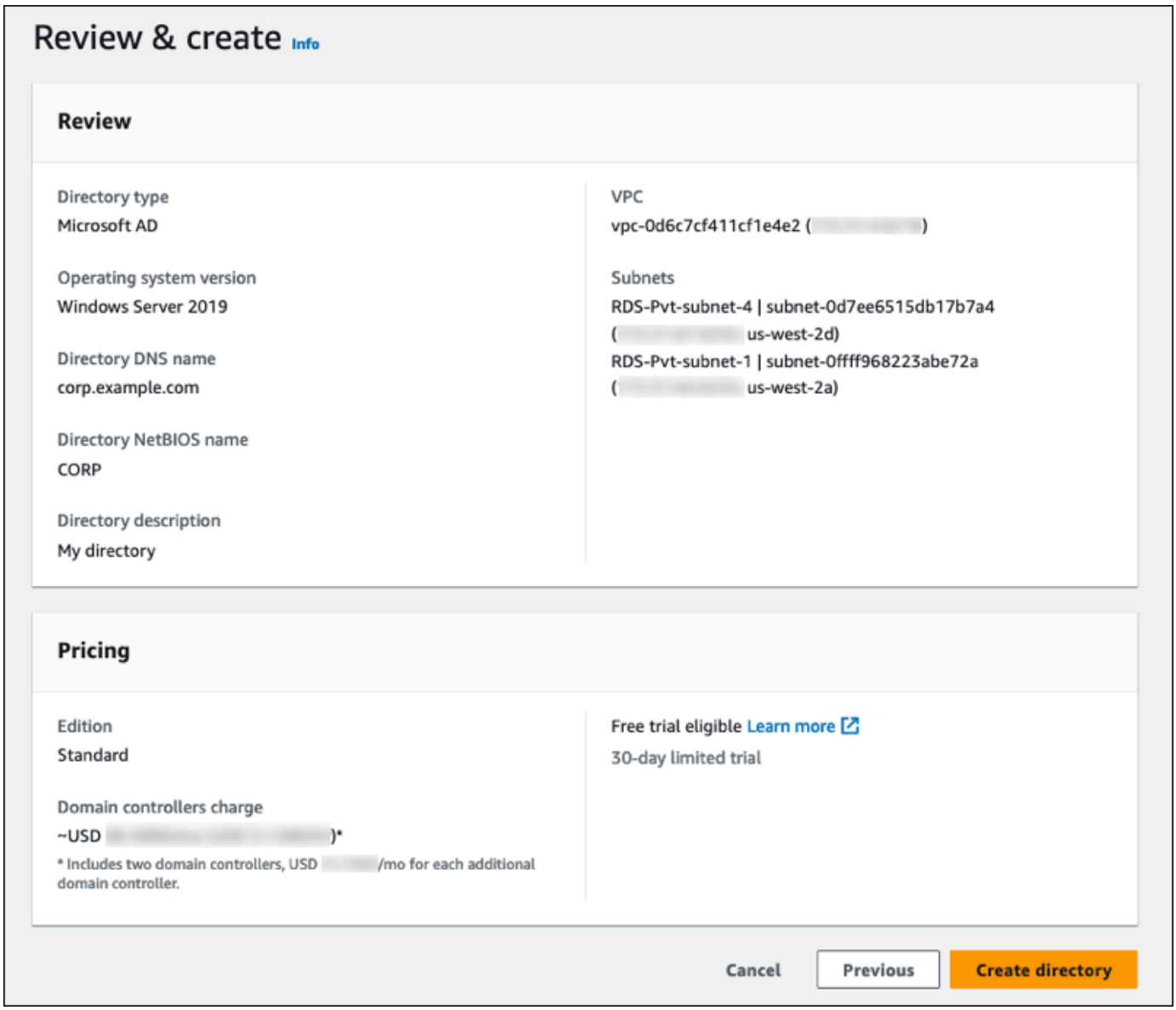
1. AWS マネジメントコンソール にサインインし、Directory Service コンソール (https://console.aws.amazon.com/directoryservicev2/)を開きます。
2. [ディレクトリの設定] を選択します。
3. [AWS Managed Microsoft AD] を選択します。現在、Amazon RDS での使用がサポートされているのは AWS Managed Microsoft AD のオプションのみです。
4. [次へ] を選択します。
5. [ディレクトリ情報の入力] ページに、以下の情報を指定します。
   - エディション - 目的の要件を満たすエディションを選択します。
   - ディレクトリ DNS 名 - ディレクトリの完全修飾名 (例: corp.example.com)。
   - ディレクトリ NetBIOS 名 - (オプション) ディレクトリの短縮名 (例: CORP)。
   - ディレクトリの説明 - (オプション) ディレクトリの説明。
   - 管理者パスワード – ディレクトリ管理者のパスワード。ディレクトリの作成プロセスでは、ユーザー名 Admin とこのパスワードを使用して管理者アカウントが作成されます。

ディレクトリ管理者のパスワードには、「admin」の単語を含めることはできません。パスワードは大文字と小文字を区別し、8-64 文字にします。また、以下の 4 つのカテゴリうち 3 つから少なくとも 1 文字を含める必要があります。

- 小文字 (a ~ z)
- 大文字 (A ~ Z)
- 数字 (0 ~ 9)
- 英数字以外の文字 (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)
- パスワードの確認 - 管理者パスワードを再入力します。

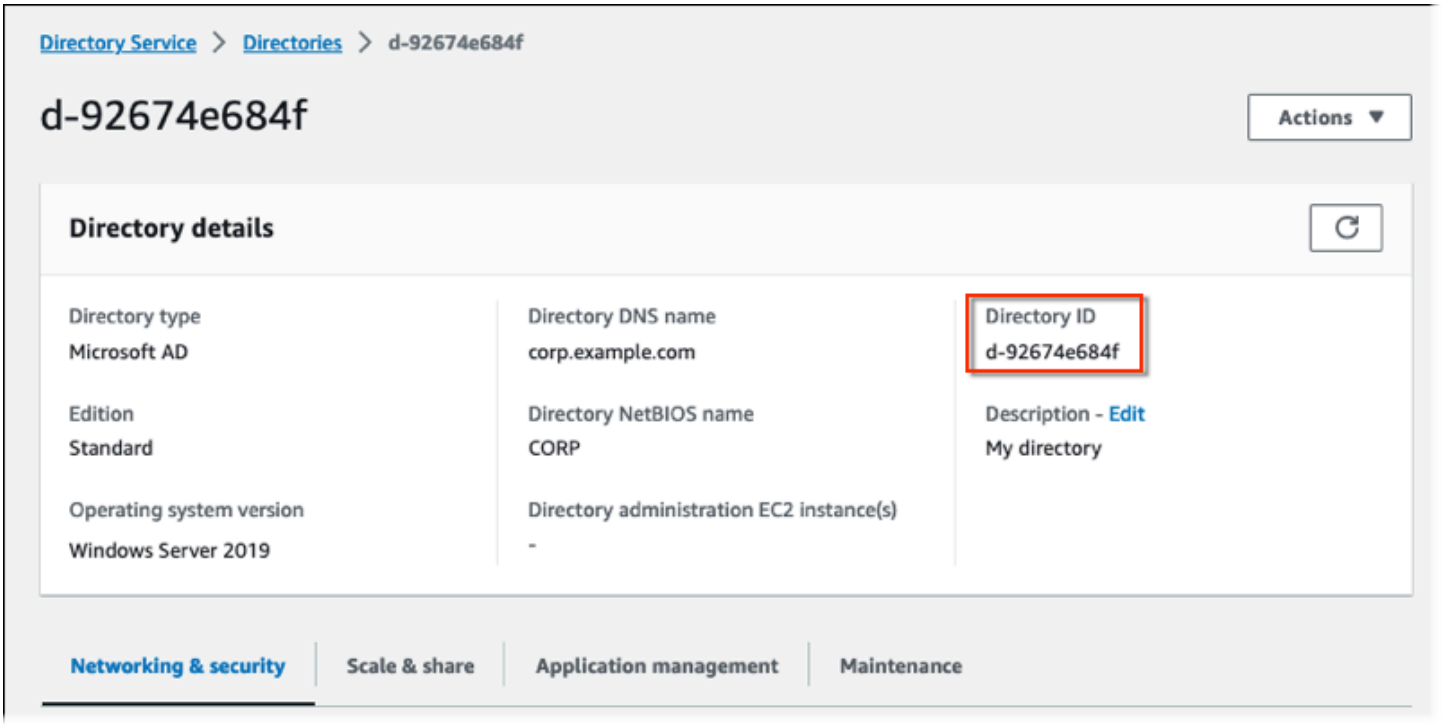
> **Important**
>
> このパスワードは必ず保管してください。Directory Service にはこのパスワードは保存されず、復元やリセットもできません。

6. [Next (次へ)] を選択します。
7. [VPC とサブネットの選択] ページで、以下の情報を指定します。
   - VPC - ディレクトリの VPC を選択します。RDS for Db2 DB インスタンスは、この同じ VPC または別の VPC で作成できます。
   - サブネット - ディレクトリサーバーのサブネットを選択します。2 つのサブネットは、異なるアベイラビリティーゾーンに存在している必要があります。
8. [Next (次へ)] を選択します。
9. ディレクトリの情報を確認します。変更が必要な場合は、[戻る] を選択し、変更を行います。情報が正しい場合は、[Create directory (ディレクトリの作成)] を選択します。



ディレクトリが作成されるまで、数分かかります。正常に作成されると、[Status] 値が [Active] に変わります。

ディレクトリに関する情報を表示するには、[ディレクトリ ID] で、ディレクトリ ID を選択します。ディレクトリ ID 値を書き留めます。RDS for Db2 DB インスタンスを作成または変更する際に、この値が必要です。



ステップ 2: 信頼関係を作成する

AWS Managed Microsoft AD のみを使用する予定の場合は、「ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する」に進みます。

セルフマネージド Active Directory を使用して Kerberos 認証を有効にするには、セルフマネージド Active Directory との間にフォレストの信頼関係を作成する必要があります。フォレストの信頼とは、Microsoft AD およびセルフマネージド Active Directory と、前のステップで作成した AWS Managed Microsoft AD との間の信頼関係です。信頼は、両方の Active Directory が相互に信頼する双方向にすることもできます。Directory Service を使用してフォレストの信頼を設定する方法の詳細については、「AWS Directory Service Administration Guide」の「When to create a trust relationship」を参照してください。

ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する

Amazon RDS が Directory Service を呼び出すには、AWS アカウント にマネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する IAM ロールが必要です。このロールにより、Amazon RDS は Directory Service への呼び出しを行うことができます。

AWS マネジメントコンソール を使用して DB インスタンスを作成し、コンソールユーザーが `iam:CreateRole` アクセス許可を持っている場合、コンソールは必要な IAM ロールを自動的に作成します。この場合、ロール名は `rds-directoryservice-kerberos-access-`

role です。それ以外の場合は、IAM ロールを手動で作成する必要があります。IAM ロールを作成する場合、[Directory Service] を選択し、それに AWS マネージドポリシー AmazonRDSDirectoryServiceAccess をアタッチします。

サービス用の IAM ロールを作成する方法の詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Note**
>
> RDS for Microsoft SQL Server の Windows 認証に使用される IAM ロールは、RDS for Db2 では使用できません。

AmazonRDSDirectoryServiceAccess マネージドポリシーを使用する代わりに、必要なアクセス許可を使用してポリシーを作成することもできます。これを行うには、IAM ロールに次の IAM 信頼ポリシーが必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

また、ロールには、以下の IAM ロールポリシーも必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

ステップ 4: ユーザーを作成して設定する

Active Directory Users and Computers ツールを使用してユーザーを作成できます。これは、Active Directory Domain Services および Active Directory Lightweight Directory Services ツールのいずれかです。詳細については、Microsoft ドキュメントの「Active Directory ドメインにユーザーとコンピューターを追加する」を参照してください。この場合、ユーザーは個人またはその他のエンティティです。例えば、ドメインの一部であり、その ID がディレクトリで管理されているコンピュータなどです。

Directory Service ディレクトリにユーザーを作成するには、Directory Service ディレクトリのメンバーである Windows ベースの Amazon EC2 インスタンスに接続している必要があります。同時に、ユーザーを作成する権限を持つユーザーとしてサインインしていなければなりません。詳細については、AWS Directory Service 管理ガイドの「ユーザーの作成」を参照してください。

ステップ 5: AWS Managed Microsoft AD で RDS for Db2 管理者グループを作成する

RDS for Db2 は、マスターユーザー、または 2 つの Amazon RDS 予約ユーザー `rdsdb` と `rdsadmin` の Kerberos 認証をサポートしていません。代わりに、AWS Managed Microsoft AD で `masterdba` という名前の新しいグループを作成する必要があります。詳細については、Microsoft ドキュメントの「Create a Group Account in Active Directory」を参照してください。このグループに追加するユーザーには、マスターユーザー権限があります。

Kerberos 認証を有効にすると、マスターユーザーは masterdba ロールを失います。その結果、Kerberos 認証を無効にしない限り、マスターユーザーはインスタンスのローカルユーザーグループのメンバーシップにアクセスできなくなります。パスワードログインでマスターユーザーを引き続き使用するには、マスターユーザーと同じ名前のユーザーを AWS Managed Microsoft AD に作成します。続いて、ユーザーをグループ masterdba に追加します。

ステップ 6: DB パラメータを変更する

AWS Managed Microsoft AD のみを使用する予定の場合は、「ステップ 7: RDS for Db2 DB インスタンスを作成または変更する」に進みます。

セルフマネージド Active Directory を使用して Kerberos 認証を有効にするには、パラメータグループのパラメータ rds.active_directory_configuration を AWS_MANAGED_AD_WITH_TRUST に設定する必要があります。デフォルトでは、このパラメータは AWS Managed Microsoft AD を使用する場合にのみ AWS_MANAGED_AD に設定されます。

DB パラメータの変更については、「パラメータグループのパラメータの変更」を参照してください。

ステップ 7: RDS for Db2 DB インスタンスを作成または変更する

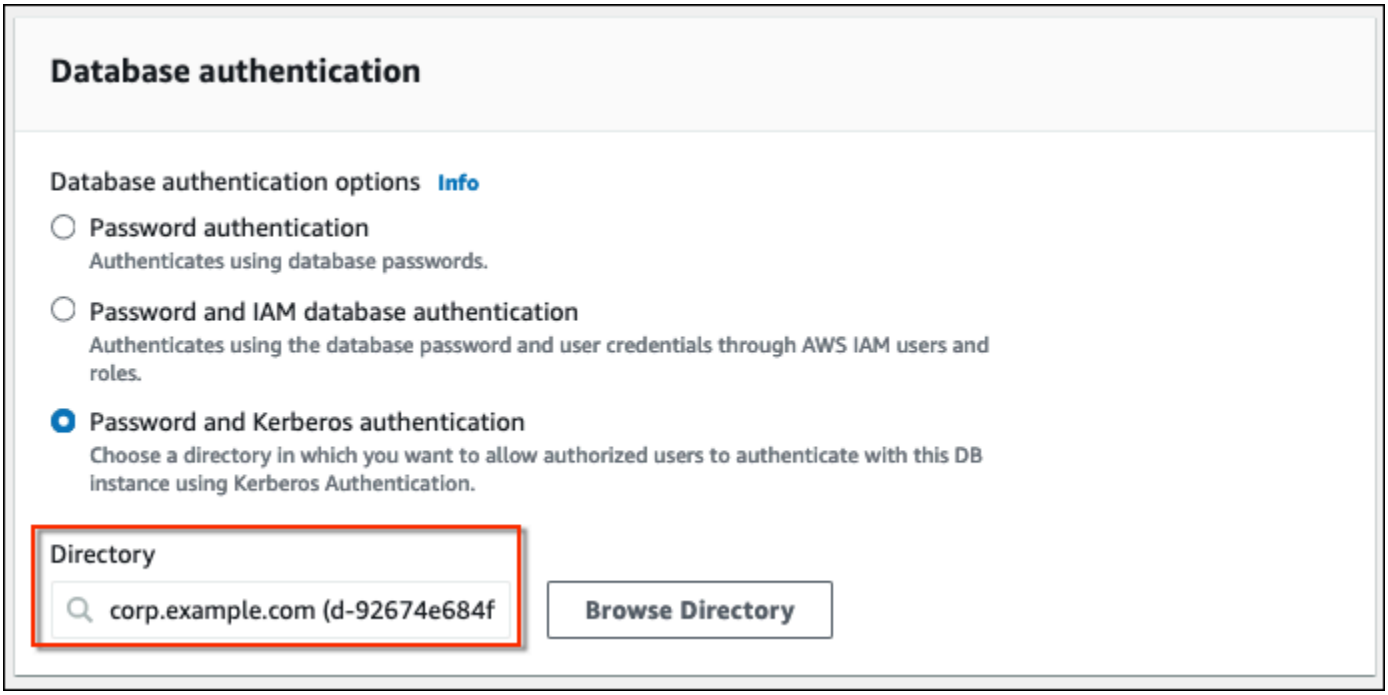
ディレクトリで使用する RDS for Db2 DB インスタンスを作成または変更します。AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これは、次のいずれかの方法で行うことができます。

- コンソール、create-db-instance コマンド、または CreateDBInstance API オペレーションを使用して新しい RDS for Db2 DB インスタンスを作成します。手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance コマンド、または ModifyDBInstance API オペレーションを使用して、既存の RDS for Db2 DB インスタンスを変更します。手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、restore-db-instance-from-db-snapshot コマンド、または RestoreDBInstanceFromDBSnapshot API オペレーションを使用して、DB スナップショットから RDS for Db2 DB インスタンスを復元します。手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time コマンド、または RestoreDBInstanceToPointInTime API オペレーションを使用して、RDS for Db2 DB インスタンスを特定時点に復元します。手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

Kerberos 認証は、VPC 内の RDS for Db2 DB インスタンスでのみサポートされています。DB インスタンスは、ディレクトリと同じ VPC または異なる VPC 内にあります。DB インスタンスがディレクトリと通信できるように、ディレクトリの VPC 内の出力を許可するセキュリティグループを使用する必要があります。

コンソール

DB インスタンスを作成、変更、復元するためにコンソールを使用する場合は、[データベース認証] セクションの [パスワードと Kerberos 認証] を選択します。次に、[ディレクトリのブラウジング] を選択します。Directory Service を使用するには、ディレクトリを選択するか、[ディレクトリを作成] を選択します。



AWS CLI

AWS CLI を使用する場合は、DB インスタンスが、作成したディレクトリを使用できるように、以下のパラメータが必要です。

- `--domain` パラメータには、ディレクトリの作成時に生成されたドメイン識別子 ("d-*" 識別子) を使用します。
- `--domain-iam-role-name` パラメータには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する作成済みのロールを使用します。

以下の例では、ディレクトリを使用するように DB インスタンスを変更します。例の、以下のプレースホルダをユーザー自身の値に置き換えます。

- *db_instance_name* - RDS for Db2 DB インスタンスの名前。
- *directory_id* - 作成した AWS Directory Service for Microsoft Active Directory ディレクトリの ID。
- *role_name* - 作成した IAM ロールの名前。

```
aws rds modify-db-instance --db-instance-identifier db_instance_name --domain
 d-directory_id --domain-iam-role-name role_name
```

> **Important**
>
> Kerberos 認証を有効化するために DB インスタンスを変更する場合は、変更後に DB インスタンスを再起動します。

ステップ 8: PowerShell で Active Directory グループ SID を取得する

セキュリティ ID (SID) は、セキュリティプリンシパルまたはセキュリティグループを一意に識別します。セキュリティグループまたはアカウントを Active Directory で作成すると、Active Directory は SID をグループに割り当てます。Active Directory から AD セキュリティグループの SID を取得するには、Active Directory ドメインの一部である Windows クライアントマシンで `Get-ADGroup` コマンドレットを使用します。`Identity` パラメータは、SID を取得する対象である Active Directory グループの名前を指定します。

次の例では、Active Directory グループ `adgroup1` の SID を返します。

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

             SID
-----------------------------------------------
S-1-5-21-3168537779-1985441202-1799118680-1612
```

データベースに関連するすべてのグループに対して、このマッピングを生成する必要があります。

ステップ 9: RDS for Db2 DB インスタンスに SID と GroupName のマッピングを追加する

前のステップで作成した SID と GroupName のマッピングを RDS for Db2 DB インスタンスに追加する必要があります。マッピングごとに、次のストアドプロシージャを呼び出します。*SID* と *group_name* を独自の情報に置き換えます。

```
db2 connect to rdsadmin
db2 "call rdsadmin.set_sid_group_mapping(?, 'SID','group_name')"
```

詳細については、「rdsadmin.set_sid_group_mapping」を参照してください。

タスクのステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

ステップ 10: Db2 クライアントを設定する

Db2 クライアントを設定するには

1. ドメインを指す /etc/krb5.conf ファイル (または同等) を作成します。

   > **Note**
   >
   > Windows オペレーティングシステムの場合は、C:\windows\krb5.ini ファイルを作成します。

2. クライアントホストと Directory Service 間でトラフィックが流れることを確認します。以下のタスクには、Netcat などのネットワークユーティリティを使用します。

   a. ポート 53 の DNS 経由のトラフィックを確認します。

   b. ポート 53 および Kerberos の TCP/UDP 上のトラフィックを確認します。これには、Directory Service の場合ポート 88 および 464 が含まれます。

3. データベースポートを介してクライアントホストと DB インスタンス間でトラフィックが流れることを確認します。コマンド db2 を使用して、データベースに接続してアクセスできます。

次の例は、AWS Managed Microsoft AD の /etc/krb5.conf ファイルの内容です。

```
[libdefaults]
default_realm = EXAMPLE.COM
[realms]
```

```
EXAMPLE.COM = {
kdc = example.com
admin_server = example.com
}
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

Kerberos 認証を使用して Amazon RDS for Db2 に接続する

Kerberos 認証を使用して Amazon RDS for Db2 DB インスタンスに接続するには、次の手順に従います。

Kerberos 認証を使用して RDS for Db2 に接続するには

1. コマンドプロンプトで、次のコマンドを実行します。次の例では、*username* をユーザーの Microsoft Active Directory ユーザー名に置き換えます。

   ```
   kinit username
   ```

2. RDS for Db2 DB インスタンスがパブリックにアクセス可能な VPC を使用している場合、Amazon EC2 クライアントの /etc/hosts ファイルに DB インスタンスエンドポイントのプライベート IP アドレスを追加します。次の例では IP アドレスを取得し、それを /etc/hosts ファイルに追加します。

   ```
   % dig +short Db2-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo "34.210.197.118  Db2-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/hosts
   ```

3. 次のコマンドを使用して、Active Directory に関連付けられている RDS for Db2 DB インスタンスにログインします。*database_name* を RDS for Db2 データベースの名前に置き換えます。

   ```
   db2 connect to database_name
   ```

Amazon RDS for Db2 DB インスタンスの管理

このトピックでは、Amazon RDS for Db2 DB インスタンスで実行する一般的な管理タスクついて説明します。一部のタスクは、すべての Amazon RDS DB インスタンスで同じです。その他のタスクは、RDS for Db2 に固有です。

以下のタスクは、すべての RDS データベースに共通です。標準の SQL クライアントを使用した RDS for Db2 データベースへの接続など、RDS for Db2 に固有のタスクもあります。

タスク領域	関連資料
インスタンスクラス、ストレージ、PIOPS 本稼働インスタンスを作成する場合は、Amazon RDS でインスタンスクラス、ストレージタイプ、プロビジョンド IOPS がどのように機能するかを学習します。	DB インスタンスクラス Amazon RDS ストレージタイプ
マルチ AZ 配置 本稼働 DB インスタンスは、マルチ AZ 配置を使用する必要があります。マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。	Amazon RDS でのマルチ AZ 配置の設定と管理
Amazon VPC AWS アカウント にデフォルトの仮想プライベートクラウド (VPC) がある場合、DB インスタンスはデフォルトの VPC 内に自動的に作成されます。アカウントにデフォルト VPC がなく、DB インスタンスを VPC に作成する必要がある場合は、DB インスタンスを作成する前に VPC とサブネットグループを作成する場合があります。	VPC 内の DB インスタンスの使用
セキュリティグループ: デフォルトでは、DB インスタンスはアクセスを防止するファイアウォールを使用します。DB インスタンスにアクセスするために、正しい IP アドレスとネットワーク構成を備えたセキュリティグループを作成する必要があります。	セキュリティグループによるアクセス制御

タスク領域	関連資料
パラメータグループ RDS for Db2 DB インスタンスでは、`rds.ibm_customer_id` および `rds.ibm_site_id` パラメータを追加する必要があるため、DB インスタンスを作成する前にパラメータグループを作成します。DB インスタンスで特定の他のデータベースパラメータが必要な場合も、DB インスタンスを作成する前にこのパラメータグループに追加します。	RDS for Db2 DB インスタンスのパラメータグループに IBM ID を追加する Amazon RDS のパラメータグループ
オプショングループ DB インスタンスに特定のデータベースオプションが必要になる場合は、DB インスタンスを作成する前にオプショングループを作成します。	RDS for Db2 DB インスタンスのオプション
DB インスタンスへの接続 セキュリティグループを作成し、DB インスタンスに関連付けると、IBM Db2 CLP などの標準的な SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。	Db2 DB インスタンスへの接続
バックアップと復元 ストレージバックアップが自動的に作成されるように DB インスタンスを設定するか、ストレージスナップショットを手動で作成しておくと、そのバックアップまたはスナップショットからインスタンスを復元できます。	データのバックアップ、復元、エクスポート

タスク領域	関連資料
モニタリング IBM Db2 Data Management Console を使用して RDS for Db2 DB インスタンスをモニタリングできます。 CloudWatch Amazon RDS メトリクス、イベント、および拡張モニタリングを使用することで、RDS for Db2 DB インスタンスをモニタリングすることもできます。	IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続する Amazon RDS コンソールでのメトリクスの表示 Amazon RDS イベントの表示 拡張モニタリングを使用した OS メトリクスのモニタリング
ログファイル RDS for Db2 DB インスタンスのログファイルにアクセスできます。	Amazon RDS ログファイルのモニタリング

トピック

- Amazon RDS for Db2 DB インスタンスの一般的なシステムタスクの実行
- Amazon RDS for Db2 DB インスタンスの一般的なデータベースタスクの実行

Amazon RDS for Db2 DB インスタンスの一般的なシステムタスクの実行

次に、Db2 を実行している Amazon RDS DB インスタンスで、システムに関連する特定の一般的なデータベース管理者タスクを実行する方法を示します。マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスは提供していません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。

RDS for Db2 の権限の付与と取り消し、およびリモートデータベースへのアタッチの詳細については、以下のトピックを参照してください。

トピック

- RDS for Db2 の権限の付与と取り消し

- リモート RDS for Db2 DB インスタンスへのアタッチ

カスタムデータベースエンドポイントの作成

Amazon RDS for Db2 に移行する場合、カスタムデータベースエンドポイント URL を使用すると、アプリケーションの変更を最小限に抑えることができます。例えば、`db2.example.com` を現在の DNS レコードとして使用すると、Amazon Route 53 に追加できます。Route 53 では、プライベートホストゾーンを使用して、現在の DNS データベースエンドポイントを RDS for Db2 データベースエンドポイントにマッピングできます。Amazon RDS データベースエンドポイントのカスタム `A` または `CNAME` レコードを追加するには、「Amazon Route 53 デベロッパーガイド」の「Amazon Route 53 を使用したドメインの登録と管理」を参照してください。

> **Note**
>
> ドメインを Route 53 に移管できない場合は、DNS プロバイダーを使用して RDS for Db2 データベースエンドポイント URL の `CNAME` レコードを作成します。DNS プロバイダーのドキュメントを参照してください。

RDS for Db2 の権限の付与と取り消し

ユーザーは、データベースにアタッチされているグループのメンバーシップを通じてデータベースへのアクセス権を取得します。

データベースへのアクセスを制御する権限を付与および取り消すには、次の手順に従います。

これらの手順では、ローカルマシンで実行されている IBM Db2 CLP を使用して RDS for Db2 DB インスタンスに接続します。ローカルマシンで実行されている RDS for Db2 DB インスタンスに接続するには、TCPIP ノードとデータベースを必ず分類してください。詳細については、「IBM Db2 CLP を使用して Amazon RDS for Db2 DB インスタンスに接続する」を参照してください。

トピック

- データベースへのユーザーアクセスの付与
- ユーザーのパスワードの変更
- ユーザーへのグループの追加
- ユーザーからのグループの削除
- ユーザーの削除

- ユーザーの一覧表示
- ロールの作成
- ロールの付与
- ロールの取り消し
- ロールの削除
- データベース認可の付与
- データベース認可の取り消し

データベースへのユーザーアクセスの付与

データベースへのユーザーアクセスを付与するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

このコマンドでは、次の例のような出力が生成されます。

```
Database Connection Information

Database server        = DB2/LINUXX8664 11.5.8.0
SQL authorization ID   = ADMIN
Local database alias   = RDSADMIN
```

2. rdsadmin.add_user を呼び出して、承認リストにユーザーを追加します。詳細については、「rdsadmin.add_user」を参照してください。

```
db2 "call rdsadmin.add_user(
    'username',
    'password',
    'group_name,group_name')"
```

3. (オプション) rdsadmin.add_groups を呼び出して、ユーザーにその他のグループを追加します。詳細については、「rdsadmin.add_groups」を参照してください。

```
db2 "call rdsadmin.add_groups(
```

```
    'username',
    'group_name,group_name')"
```

4. ユーザーが利用できる権限を確認します。次の例で、*rds_database_alias*、*master_user*、*master_password* を自分の情報に置き換えます。また、*username* をユーザーのユーザー名に置き換えます。

```
db2 terminate
db2 connect to rds_database_alias user master_user using master_password
db2 "SELECT SUBSTR(AUTHORITY,1,20) AUTHORITY, D_USER, D_GROUP, D_PUBLIC
       FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('username', 'U') ) AS
 T
       ORDER BY AUTHORITY"
```

このコマンドでは、次の例のような出力が生成されます。

```
AUTHORITY            D_USER D_GROUP D_PUBLIC
-------------------- ------ ------- --------
ACCESSCTRL           N      N       N
BINDADD              N      N       N
CONNECT              N      N       N
CREATETAB            N      N       N
CREATE_EXTERNAL_ROUT N      N       N
CREATE_NOT_FENCED_RO N      N       N
CREATE_SECURE_OBJECT N      N       N
DATAACCESS           N      N       N
DBADM                N      N       N
EXPLAIN              N      N       N
IMPLICIT_SCHEMA      N      N       N
LOAD                 N      N       N
QUIESCE_CONNECT      N      N       N
SECADM               N      N       N
SQLADM               N      N       N
SYSADM               *      N       *
SYSCTRL              *      N       *
SYSMAINT             *      N       *
SYSMON               *      N       *
WLMADM               N      N       N
```

5. ユーザーを追加したグループに RDS for Db2 ロール ROLE_NULLID_PACKAGES、ROLE_TABLESPACES、および ROLE_PROCEDURESを付与します。詳細については、「Amazon RDS for Db2 のデフォルトロール」を参照してください。

> **Note**
>
> RDS for Db2 DB インスタンスは RESTRICTIVE モードで作成されています。このため、RDS for Db2 ロール ROLE_NULLID_PACKAGES、ROLE_TABLESPACES、および ROLE_PROCEDURES によって、IBM Db2 CLP および Dynamic SQL の NULLID パッケージに対する実行権限が付与されます。これらのロールによって、テーブルスペースに対するユーザー権限も付与されます。

a. Db2 データベースに接続します。次の例で、*database_name*、*master_user*、*master_password* を自分の情報に置き換えます。

```
db2 connect to database_name user master_user using master_password
```

b. グループにロール ROLE_NULLED_PACKAGES を付与します。次の例では、*group_name* を、ロールを追加するグループの名前に置き換えます。

```
db2 "grant role ROLE_NULLID_PACKAGES to group group_name"
```

c. 同じグループにロール ROLE_TABLESPACES を付与します。次の例では、*group_name* を、ロールを追加するグループの名前に置き換えます。

```
db2 "grant role ROLE_TABLESPACES to group group_name"
```

d. 同じグループにロール ROLE_PROCEDURES を付与します。次の例では、*group_name* を、ロールを追加するグループの名前に置き換えます。

```
db2 "grant role ROLE_PROCEDURES to group group_name"
```

6. ユーザーを追加したグループに connect、bindadd、createtab、および IMPLICIT_SCHEMA 権限を付与します。次の例では、*group_name* を、ユーザーを追加した 2 番目のグループの名前に置き換えます。

```
db2 "grant usage on workload SYSDEFAULTUSERWORKLOAD to public"
db2 "grant connect, bindadd, createtab, implicit_schema on database to
 group group_name"
```

7. ユーザーを追加する追加のグループごとに、ステップ 4～6 を繰り返します。

8. ユーザーとして接続して、テーブルを作成し、テーブルに値を挿入して、テーブルからデータを返すことで、ユーザーのアクセスをテストします。次の例では、*rds_database_alias*、*username* 、*password* をデータベースの名前およびユーザーのユーザー名とパスワードに置き換えます。

```
db2 connect to rds_database_alias user username using password
db2 "create table t1(c1 int not null)"
db2 "insert into t1 values (1),(2),(3),(4)"
db2 "select * from t1"
```

ユーザーのパスワードの変更

ユーザーのパスワードを変更するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.change_password を呼び出して、パスワードを変更します。詳細については、「rdsadmin.change_password」を参照してください。

```
db2 "call rdsadmin.change_password(
    'username',
    'new_password')"
```

ユーザーへのグループの追加

グループをユーザーに追加するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.add_groups を呼び出して、グループをユーザーに追加します。詳細については、「rdsadmin.add_groups」を参照してください。

```
db2 "call rdsadmin.add_groups(
    'username',
    'group_name,group_name')"
```

ユーザーからのグループの削除

ユーザーからグループを削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.remove_groups を呼び出して、グループを削除します。詳細については、「rdsadmin.remove_groups」を参照してください。

```
db2 "call rdsadmin.remove_groups(
    'username',
    'group_name,group_name')"
```

ユーザーの削除

承認リストからユーザーを削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.remove_user を呼び出して、承認リストからユーザーを削除します。詳細については、「rdsadmin.remove_user」を参照してください。

```
db2 "call rdsadmin.remove_user('username')"
```

ユーザーの一覧表示

承認リストにユーザーを一覧表示するには、 rdsadmin.list_users ストアドプロシージャを呼び出します。詳細については、「rdsadmin.list_users」を参照してください。

```
db2 "call rdsadmin.list_users()"
```

ロールの作成

rdsadmin.create_role ストアドプロシージャを使用すると、ロールを作成できます。

ロールを作成するには

1. rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. コンテンツを出力するように Db2 を設定します。

   ```
   db2 set serveroutput on
   ```

3. ロールを作成します。詳細については、「the section called “rdsadmin.create_role”」を参照してください。

   ```
   db2 "call rdsadmin.create_role(
       'database_name',
       'role_name')"
   ```

4. コンテンツを出力しないように Db2 を設定します。

   ```
   db2 set serveroutput off
   ```

ロールの付与

rdsadmin.grant_role ストアドプロシージャを使用すると、ロール、ユーザー、またはグループにロールを割り当てることができます。

ロールを割り当てるには

1. rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. コンテンツを出力するように Db2 を設定します。

```
db2 set serveroutput on
```

3. ロールを割り当てます。詳細については、「the section called “rdsadmin.grant_role”」を参照してください。

```
db2 "call rdsadmin.grant_role(
    'database_name',
    'role_name',
    'grantee',
    'admin_option')"
```

4. コンテンツを出力しないように Db2 を設定します。

```
db2 set serveroutput off
```

ロールの取り消し

rdsadmin.revoke_role ストアドプロシージャを使用すると、ロール、ユーザー、またはグループのロールを取り消すことができます。

ロールを取り消すには

1. rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. ロールを取り消します。詳細については、「the section called “rdsadmin.revoke_role”」を参照してください。

```
db2 "call rdsadmin.revoke_role(
```

```
    ?,
    'database_name',
    'role_name',
    'grantee')"
```

ロールの削除

ロールを削除するには、rdsadmin.drop_role ストアドプロシージャを使用します。

ロールを削除するには

1. rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. ロールを削除します。詳細については、「the section called “rdsadmin.drop_role”」を参照してください。

```
db2 "call rdsadmin.drop_role(
    ?,
    'database_name',
    'role_name')"
```

データベース認可の付与

DBADM 権限を持つマスターユーザーは、ロール、ユーザー、またはグループに DBADM、ACCESSCTRL、または DATAACCESS 権限を付与できます。

データベース認可を付与するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.dbadm_grant を呼び出して、ユーザーアクセスを付与します。詳細については、「rdsadmin.dbadm_grant」を参照してください。

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

ユースケースの例

次の手順では、ロールの作成、ロールへの DBADM 承認の付与、ユーザーへのロールの割り当て、およびグループへのロールの付与について説明します。

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

2. TESTDB というデータベースの PROD_ROLE というロールを作成します。詳細については、「rdsadmin.create_role」を参照してください。

   ```
   db2 "call rdsadmin.create_role(
       'TESTDB',
       'PROD_ROLE')"
   ```

3. PROD_USER というユーザーにロールを割り当てます。PROD_USER には、ロールを割り当てる管理者権限が付与されます。詳細については、「rdsadmin.grant_role」を参照してください。

   ```
   db2 "call rdsadmin.grant_role(
       ?,
       'TESTDB',
       'PROD_ROLE',
       'USER PROD_USER',
       'Y')"
   ```

4. (オプション) 追加の権限または特権を指定します。次の例では、FUNDPROD というデータベースの PROD_ROLE という名前のロールに DBADM 権限を付与します。詳細については、「rdsadmin.dbadm_grant」を参照してください。

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'FUNDPROD',
    'DBADM',
    'ROLE PROD_ROLE')"
```

5. セッションを終了します。

```
db2 terminate
```

6. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、TESTDB データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to TESTDB user master_username using master_password
```

7. ロールにさらに権限を追加します。

```
db2 "grant connect, implicit_schema on database to role PROD_ROLE"
```

8. グループにロール PROD_ROLE を付与します。

```
db2 "grant role PROD_ROLE to group PRODGRP"
```

グループ PRODGRP に属するユーザーは、TESTDB データベースへの接続、テーブルの作成、スキーマの作成などのアクションを実行できるようになりました。

データベース認可の取り消し

DBADM 権限を持つマスターユーザーは、ロール、ユーザー、またはグループの DBADM、ACCESSCTRL、または DATAACCESS 権限を取り消すことができます。

データベース認可の取り消すには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.dbadm_revoke を呼び出して、ユーザーアクセスを取り消します。詳細については、「rdsadmin.dbadm_revoke」を参照してください。

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

リモート RDS for Db2 DB インスタンスへのアタッチ

リモート RDS for Db2 DB インスタンスにアタッチし、get snapshot オペレーションを実行するには、次の手順を実行します。

リモート RDS for Db2 DB インスタンスにアタッチするには

1. クライアント側の IBM Db2 CLP セッションを実行します。RDS for Db2 DB インスタンスとデータベースのカタログ化については、「IBM Db2 CLP を使用して Amazon RDS for Db2 DB インスタンスに接続する」を参照してください。RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを書き留めます。
2. RDS for Db2 DB インスタンスにアタッチします。次の例では、*node_name*、*master_username*、*master_password* を、カタログ化した TCPIP ノード名と、RDS for Db2 DB インスタンスのマスターユーザー名およびマスターパスワードに置き換えます。

```
db2 attach to node_name user master_username using master_password
```

リモート RDS for Db2 DB インスタンスにアタッチすると、次のコマンドやその他の get snapshot コマンドを実行できるようになります。詳細については、IBM Db2 ドキュメントの「GET SNAPSHOT command」をご参照ください。

```
db2 list applications
db2 get snapshot for all databases
db2 get snapshot for database manager
db2 get snapshot for all applications
```

Amazon RDS for Db2 DB インスタンスの一般的なデータベースタスクの実行

Amazon RDS for Db2 DB インスタンスのデータベースに関連する特定の一般的な DBA タスクを実行することができます。マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、マスターユーザーは、SYSADM、SYSMAINT、または SYSCTRL 権限を必要とするコマンドやユーティリティを実行できません。

バッファプール、データベース、テーブルスペースの一般的なタスクについては、以下のトピックを参照してください。

トピック

- バッファプールの一般的なタスク
- 一般的なデータベースタスク
- テーブルスペースの一般的なタスク

バッファプールの一般的なタスク

RDS for Db2 データベースのバッファプールを作成、変更、または削除できます。バッファプールを作成、変更、または削除するには、マスターユーザーが使用できない上位レベルの SYSADM または SYSCTRL 権限が必要です。代わりに、Amazon RDS ストアドプロシージャを使用します。

バッファプールは、フラッシュすることもできます。

トピック

- バッファプールの作成
- バッファプールの変更
- バッファプールの削除
- バッファプールのフラッシュ

バッファプールの作成

RDS for Db2 データベースのバッファプールを作成するには、 `rdsadmin.create_bufferpool` ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「CREATE BUFFERPOOL statement」を参照してください。

バッファプールを作成するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.create_bufferpool を呼び出してバッファプールを作成します。詳細については、「rdsadmin.create_bufferpool」を参照してください。

```
db2 "call rdsadmin.create_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    page_size,
    number_block_pages,
    block_size)"
```

バッファプールの変更

RDS for Db2 データベースのバッファプールを変更するには、 rdsadmin.alter_bufferpool ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「ALTER BUFFERPOOL statement」を参照してください。

バッファプールを変更するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.alter_bufferpool を呼び出してバッファプールを変更します。詳細については、「rdsadmin.alter_bufferpool」を参照してください。

```
db2 "call rdsadmin.alter_bufferpool(
    'database_name',
```

```
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    change_number_blocks,
    number_block_pages,
    block_size)"
```

バッファプールの削除

RDS for Db2 データベースのバッファプールを削除するには、 `rdsadmin.drop_bufferpool` ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「Dropping buffer pools」を参照してください。

> **Important**
>
> 削除するバッファプールにテーブルスペースが割り当てられていないことを確認します。

バッファプールを削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、`rdsadmin` データベースに接続します。次の例で、*`master_username`* と *`master_password`* を自分の情報に置き換えます。

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.drop_bufferpool` を呼び出してバッファプールを削除します。詳細については、「rdsadmin.drop_bufferpool」を参照してください。

   ```
   db2 "call rdsadmin.drop_bufferpool(
       'database_name',
       'buffer_pool_name')"
   ```

バッファプールのフラッシュ

RDS for Db2 がメモリからストレージにページを書き込むように、バッファプールをフラッシュしてチェックポイントを強制できます。

> **Note**
>
> バッファプールは、フラッシュする必要はありません。Db2 はトランザクションをコミットする前にログを同期的に書き込みます。ダーティページがまだバッファプールに存在している能性がありますが、Db2 はそれらを非同期的にストレージに書き込みます。システムが予期せずシャットダウンした場合でも、データベースを再起動すると、Db2 は自動的にクラッシュリカバリを実行します。クラッシュリカバリ中、Db2 はコミットされた変更をデータベースに書き込むか、コミットされていないトランザクションの変更をロールバックします。

バッファプールをフラッシュするには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、Db2 データベースに接続します。次の例で、*rds_database_alias*、*master_username*、*master_password* を自分の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

2. バッファプールをフラッシュします。

```
db2 flush bufferpools all
```

一般的なデータベースタスク

RDS for Db2 DB インスタンスでデータベースを作成、削除、または復元できます。データベースを作成、削除、または復元するには、マスターユーザーが使用できない上位レベルの SYSADM 権限が必要です。代わりに、Amazon RDS ストアドプロシージャを使用します。

また、モニタリング、メンテナンス、データベースに関する情報の収集など、一般的な管理タスクを実行することもできます。

トピック

- データベースを作成する
- データベース設定の構成
- データベースのパラメータの修正

- ログの保持を設定
- ログ情報の一覧表示
- きめ細かなアクセス制御 (FGAC) の使用
- データベースの非アクティブ化
- データベースのアクティブ化
- データベースの再アクティブ化
- データベースの削除
- データベースのバックアップ
- Amazon S3 へのアーカイブログのコピー
- データベースの復元
- データベースのリスト化
- データベースに関する情報の収集
- データベースからのアプリケーションの強制削除
- パフォーマンスレポートの生成
- db2exfmt レポートの生成

データベースを作成する

RDS for Db2 DB インスタンスでデータベースを作成するには、`rdsadmin.create_database` ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「CREATE DATABASE コマンド」を参照してください。

> **Note**
>
> `db2_compatibility_vector` パラメータを変更する場合は、データベースを作成する前に変更します。詳細については、「db2_compatibility_vector パラメータの設定」を参照してください。

データベースを作成する

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、`rdsadmin` データベースに接続します。次の例で、*`master_username`* と *`master_password`* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.create_database を呼び出してデータベースを作成します。詳細については、「rdsadmin.create_database」を参照してください。

```
db2 "call rdsadmin.create_database(
    'database_name',
    'database_page_size',
    'database_code_set',
    'database_territory',
    'database_collation',
    'database_autoconfigure_str',
    'database_non-restrictive')"
```

3. (オプション) 作成する各データベースを、rdsadmin.create_database を呼び出して作成します。各 Db2 DB インスタンスには、最大 50 個のデータベースを含めることができます。詳細については、「rdsadmin.create_database」を参照してください。

```
db2 "call rdsadmin.create_database('database_name')"
```

4. (オプション) 次のいずれかの方法を使用して、データベースが作成されたことを確認します。

   - rdsadmin.list_databases を呼び出します。詳細については、「rdsadmin.list_databases」を参照してください。
   - 次の SQL コマンドを実行します。

```
db2 "select varchar(r.task_type,25) as task_type, r.database_name,
    varchar(r.lifecycle,15) as lifecycle, r.created_at, r.database_name,
    varchar(bson_to_json(task_input_params),256) as input_params,
    varchar(r.task_output,1024) as task_output
    from table(rdsadmin.get_task_status(null,null,'create_database'))
    as r order by created_at desc"
```

データベース設定の構成

RDS for Db2 DB インスタンスのデータベースの設定を構成するには、rdsadmin.set_configuration ストアドプロシージャを呼び出します。例えば、復元オペレーション中に作成するバッファまたはバッファマニピュレータの数を設定できます。

データベース設定の構成方法

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (オプション) rdsadmin.show_configuration を呼び出して、現在の構成設定を確認します。詳細については、「the section called “rdsadmin.show_configuration”」を参照してください。

```
db2 "call rdsadmin.show_configuration('name')"
```

3. rdsadmin.set_configuration を呼び出して、データベースの設定を構成します。詳細については、「the section called “rdsadmin.set_configuration”」を参照してください。

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

データベースのパラメータの修正

Amazon RDS for Db2 は、データベースマネージャー設定パラメータ、レジストリ変数、データベース設定パラメータの 3 種類のパラメータを使用します。最初の 2 つのタイプはパラメータグループで管理でき、最後のタイプは rdsadmin.update_db_param ストアドプロシージャで更新できます。

> **Note**
>
> 既存のパラメータの値のみ変更できます。RDS for Db2 がサポートしていない新しいパラメータを追加することはできません。

これらのパラメータとその値を変更する方法の詳細については、「the section called “Db2 パラメータ”」を参照してください。

ログの保持を設定

Amazon RDS が RDS for Db2 データベースのログファイルを保持する期間を設定するには、rdsadmin.set_archive_log_retention ストアドプロシージャを呼び出します。

データベースのログの保持の設定方法

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (オプション) rdsadmin.show_archive_log_retention を呼び出して、現在の設定でログの保持を確認します。詳細については、「the section called “rdsadmin.show_archive_log_retention”」を参照してください。

```
db2 "call rdsadmin.show_archive_log_retention(
    ?,
    'database_name')"
```

3. rdsadmin.set_archive_log_retention を呼び出して、データベースのログ保持を設定します。詳細については、「the section called “rdsadmin.set_archive_log_retention”」を参照してください。

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'database_name',
    'archive_log_retention_hours')"
```

ログ情報の一覧表示

使用された合計ストレージサイズなど、アーカイブログファイルの詳細を一覧表示するには、rdsadmin.list_archive_log_information ストアドプロシージャを呼び出します。

データベースのログ情報を一覧表示するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.list_archive_log_information を呼び出して、ログファイル情報のリストを返します。詳細については、「the section called “rdsadmin.list_archive_log_information”」を参照してください。

```
db2 "call rdsadmin.list_archive_log_information(
    ?,
    'database_name')"
```

きめ細かなアクセス制御 (FGAC) の使用

きめ細かなアクセス制御コマンドを使用して RDS for Db2 DB インスタンスのデータベース内のテーブルデータへのアクセスを制御するには、rdsadmin.fgac_command ストアドプロシージャを呼び出します。FGAC を使用して、ユーザーロールまたはデータ属性に基づいてデータへのアクセスを制限できます。例えば、データの種類や特定の医療プロバイダーに基づいて、患者の医療データへのアクセスを制限できます。

きめ細かなアクセス制御を使用してデータベース内のテーブルデータへのアクセスを制御するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.fgac_command を呼び出して、さまざまなきめ細かなアクセス制御コマンドを実行します。詳細については、「rdsadmin.fgac_command」を参照してください。

```
db2 "call rdsadmin.fgac_command(
    ?,
    'database_name',
    'fgac_command')"
```

データベースの非アクティブ化

RDS for Db2 DB インスタンスのデータベースを非アクティブ化するには、rdsadmin.deactivate_database ストアドプロシージャを呼び出します。

デフォルトでは、Amazon RDS は RDS for Db2 DB インスタンスでデータベースを作成するときに、データベースをアクティブ化します。使用頻度の低いデータベースを非アクティブ化して、メモリリソースを節約できます。

データベースを非アクティブ化するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.deactivate_database を呼び出してデータベースを非アクティブ化します。詳細については、「rdsadmin.deactivate_database」を参照してください。

```
db2 "call rdsadmin.deactivate_database(
    ?,
    'database_name')"
```

データベースのアクティブ化

スタンドアロン RDS for Db2 DB インスタンスのデータベースをアクティブ化するには、rdsadmin.activate_database ストアドプロシージャを呼び出します。

デフォルトでは、Amazon RDS は RDS for Db2 DB インスタンスでデータベースを作成するときに、データベースをアクティブ化します。使用頻度の低いデータベースを非アクティブ化してメモリリソースを節約したり、後でそのデータベースをアクティブ化したりできます。

データベースをアクティブ化するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.activate_database を呼び出してデータベースをアクティブ化します。詳細については、「rdsadmin.activate_database」を参照してください。

```
db2 "call rdsadmin.activate_database(
```

```
    ?,
    'database_name')"
```

データベースの再アクティブ化

レプリカソースの RDS for Db2 DB インスタンスのデータベースを再アクティブ化するには、rdsadmin.reactivate_database ストアドプロシージャを呼び出します。データベース設定を変更した後、RDS for Db2 DB インスタンスのデータベースを再アクティブ化する必要が生じる場合があります。データベースを再アクティブ化する必要があるかどうかを判断するには、データベースに接続して db2 get db cfg show detail を実行します。

また、このストアドプロシージャを呼び出して、データベース設定を変更した後にスタンドアロン RDS for Db2 DB インスタンスのデータベースを再アクティブ化することもできます。または、まず rdsadmin.deactivate_database ストアドプロシージャを呼び出し、次に rdsadmin.activate_database ストアドプロシージャを呼び出して、スタンドアロン RDS for Db2 DB インスタンスのデータベースを再アクティブ化することもできます。詳細については、「データベースの非アクティブ化」および「データベースのアクティブ化」を参照してください。

データベースを再アクティブ化するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. rdsadmin.reactivate_database を呼び出してデータベースを再アクティブ化します。詳細については、「rdsadmin.reactivate_database」を参照してください。

   ```
   db2 "call rdsadmin.reactivate_database(
       ?,
       'database_name')"
   ```

データベースの削除

RDS for Db2 DB インスタンスからデータベースを削除するには、rdsadmin.drop_database ストアドプロシージャを呼び出します。詳細については、「IBM Db2 ドキュメント」の「データベースの削除」を参照してください。

> **Note**
>
> 特定の条件が満たされた場合にのみ、ストアドプロシージャを呼び出してデータベースを削除できます。詳細については、「rdsadmin.drop_database の the section called “使用に関する注意事項”」を参照してください。

データベースを削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. rdsadmin.drop_database を呼び出してデータベースを削除します。詳細については、「rdsadmin.drop_database」を参照してください。

   ```
   db2 "call rdsadmin.drop_database('database_name')"
   ```

データベースのバックアップ

RDS for Db2 DB インスタンスのデータベースを Amazon S3 にバックアップするには、rdsadmin.backup_database ストアドプロシージャを呼び出します。詳細については、「IBM Db2 ドキュメント」の「BACKUP DATABASE command」を参照してください。

> **Note**
>
> このストアドプロシージャでは、Amazon S3 との統合を使用します。続行する前に、統合が設定されていることを確認してください。詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。

データベースをバックアップするには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. S3 の VPC ゲートウェイエンドポイントを作成します。S3 用の VPC ゲートウェイエンドポイントが既にある場合は、ステップ 4 に進みます。

   RDS for Db2 DB インスタンスから Amazon S3 と対話できるようにするには、プライベートサブネットが使用する VPC と Amazon S3 ゲートウェイエンドポイントが必要です。詳細については、「ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する」を参照してください。

3. S3 の VPC ゲートウェイエンドポイントを確認します。詳細については、「ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する」を参照してください。

4. `rdsadmin.backup_database` を呼び出してデータベースをバックアップします。詳細については、「rdsadmin.backup_database」を参照してください。

```
db2 "call rdsadmin.backup_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    'backup_type',
    'compression_option',
    'util_impact_priority',
    'num_files',
    'parallelism',
    'num_buffers')"
```

5. 接続を終了します。

```
terminate
```

6. (オプション) バックアップファイルが *s3_prefix/dbi_resource_id/db_name* で Amazon S3 バケットにアップロードされたことを確認します。ファイルが *s3_prefix/dbi_resource_id/db_name* に表示されない場合は、データベースのバックアップのステータスを確認して問題を特定します。詳細については、「rdsadmin.get_task_status」を参照してください。特定された問題を解決できない場合は、AWS サポートにお問い合わせください。

7. (オプション) Amazon S3 へのバックアップが完了したら、バックアップを RDS for Db2 DB インスタンスまたはローカルサーバーなどの別の場所に復元できます。RDS for Db2 DB インスタンスに復元する方法については、「データベースの復元」を参照してください。

Amazon S3 へのアーカイブログのコピー

Db2 アーカイブログを RDS for Db2 DB インスタンスから Amazon S3 にコピーできるようになりました。rdsadmin.backup_database を使用して作成されたネイティブバックアップと組み合わせたアーカイブログは、別の RDS for Db2 インスタンスまたは EC2 データベースの特定の時点にデータベースを復元およびロールフォワードするために使用できます。

この機能を設定する前に、ストアドプロシージャ rdsadmin.backup_database を使用して RDS for Db2 データベースを設定します。

この機能は RDS for Db2 DB インスタンスレベルで動作しますが、アーカイブログのコピーはデータベースごとに有効または無効にできます。

Amazon S3 へのアーカイブログのコピーを設定するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. the section called “rdsadmin.set_configuration” を呼び出してアーカイブログの S3 へのバックアップを設定します。

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

例:

```
db2 "call rdsadmin.set_configuration('ARCHIVE_LOG_COPY_TARGET_S3_ARN',
 'arn:aws:s3:::my_rds_db2_backups/archive-log-copy/')"
```

3. rdsadmin.enable_archive_log_copy を呼び出して、データベースのアーカイブログのコピーを有効にします。*database_name* をデータベースの名前に置き換えます。

```
db2 "call rdsadmin.enable_archive_log_copy(?, 'database_name')"
```

4. 同様に、データベースのアーカイブログのコピーを無効にするには、rdsadmin.disable_archive_log_copy を呼び出します。

```
db2 "call rdsadmin.disable_archive_log_copy(?, 'database_name')"
```

5. rdsadmin.list_databases を呼び出して、アーカイブログのコピーステータスを確認します。

```
db2 "select * from table(rdsadmin.list_databases())"
```

サンプル出力。

```
DATABASE_NAME   CREATE_TIME                 DATABASE_UNIQUE_ID
                         ARCHIVE_LOG_RETENTION_HOURS ARCHIVE_LOG_COPY
 ARCHIVE_LOG_LAST_UPLOAD_FILE ARCHIVE_LOG_LAST_UPLOAD_FILE_TIME
 ARCHIVE_LOG_COPY_STATUS
--------------- -------------------------
 ------------------------------------------------- --------------------------
 --------------- --------------------------- ---------------------------------
 ------------------------------
RDSADMIN        2026-01-06-02.03.42.569069 RDSADMIN
                                    0 DISABLED         -
  -                                 -
FOO             2026-01-06-02.13.42.885650 F0D81C7E-7213-4565-B376-4F33FCF420E3
                                    7 ENABLED          S0006536.LOG
  2026-01-28-19.15.10.000000        UPLOADING
CODEP           2026-01-14-19.42.42.508476 106EEF95-6E30-4FFF-85AE-B044352DF095
                                    0 DISABLED         -
  -                                 -
...
```

データベースの復元

Amazon S3 バケットのデータベースを RDS for Db2 DB インスタンスに移動するには、rdsadmin.restore_database ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「RESTORE DATABASE コマンド」を参照してください。

データベースを復元するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. (オプション) rdsadmin.show_configuration を呼び出して復元オペレーションを最適化するには、現在の設定を確認します。詳細については、「the section called "rdsadmin.show_configuration"」を参照してください。

```
db2 "call rdsadmin.show_configuration('name')"
```

3. rdsadmin.set_configuration を呼び出して、復元オペレーションを最適化するように設定します。これらの値を明示的に設定すると、大量のデータを含むデータベースを復元する際のパフォーマンスを向上させることができます。詳細については、「the section called "rdsadmin.set_configuration"」を参照してください。

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

4. rdsadmin.restore_database を呼び出してデータベースを復元します。詳細については、「the section called "rdsadmin.restore_database"」を参照してください。

```
db2 "call rdsadmin.restore_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    restore_timestamp,
    'backup_type')"
```

5. (オプション) データベースが復元されたことを、rdsadmin.list_databases を呼び出して確認し、そのデータベースがリストにあるかをチェックします。詳細については、「rdsadmin.list_databases」を参照してください。

6. rdsadmin.rollforward_database を呼び出して、データベースをオンラインにし、追加のトランザクションログを適用します。詳細については、「the section called "rdsadmin.rollforward_database"」を参照してください。

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'database_name',
    's3_bucket_name',
    s3_prefix,
    'rollforward_to_option',
    'complete_rollforward')"
```

7. (オプション) the section called “rdsadmin.rollforward_status” ストアドプロシージャを呼び出して、rdsadmin.rollforward_database ストアドプロシージャのステータスを確認します。
8. 前のステップで complete_rollforward を FALSE に設定した場合は、rdsadmin.complete_rollforward を呼び出してデータベースをオンラインに戻す必要があります。詳細については、「the section called “rdsadmin.complete_rollforward”」を参照してください。

```
db2 "call rdsadmin.complete_rollforward(
    ?,
    'database_name')"
```

9. (オプション) the section called “rdsadmin.rollforward_status” ストアドプロシージャを呼び出して、rdsadmin.complete_rollforward ストアドプロシージャのステータスを確認します。

データベースのリスト化

Amazon RDS for Db2 で実行中のすべてのデータベースを一覧表示するには、rdsadmin.list_databases ユーザー定義関数を呼び出します。

データベースを一覧表示するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.list_databases を呼び出してデータベースを一覧表示します。詳細については、「rdsadmin.list_databases」を参照してください。

```
db2 "select * from table(rdsadmin.list_databases())"
```

データベースに関する情報の収集

RDS for Db2 DB インスタンスのデータベースに関する情報を収集するには、rdsadmin.db2pd_command ストアドプロシージャを呼び出します。この情報は、データベースのモニタリングや問題のトラブルシューティングに役立ちます。

データベースに関する情報を収集するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.db2pd_command を呼び出して、データベースに関する情報を収集します。詳細については、「rdsadmin.db2pd_command」を参照してください。

```
db2 "call rdsadmin.db2pd_command('db2pd_cmd')"
```

データベースからのアプリケーションの強制削除

アプリケーションを RDS for Db2 DB インスタンスのデータベースから強制的にオフにするには、rdsadmin.force_application ストアドプロシージャを呼び出します。データベースのメンテナンスを実行する前に、アプリケーションをデータベースから強制的にオフにします。

データベースからアプリケーションを強制的に削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.force_application を呼び出して、アプリケーションをデータベースから強制的に削除します。詳細については、「rdsadmin.force_application」を参照してください。

```
db2 "call rdsadmin.force_application(
    ?,
    'applications')"
```

パフォーマンスレポートの生成

プロシージャまたはスクリプトを使用して、パフォーマンスレポートを生成できます。手順の使用方法については、 IBM Db2 ドキュメントの「DBSUMMARY procedure - Generate a summary report of system and application performance metrics」を参照してください。

Db2 では ~sqllib/sample/perf ディレクトリに db2mon.sh ファイルを含めます。スクリプトを実行すると、低コストで広範な SQL メトリクスレポートが生成されます。db2mon.sh ファイルおよび関連するスクリプトファイルをダウンロードするには、IBM db2-samples GitHubリポジトリの perf ディレクトリを参照してください。

スクリプトを使用してパフォーマンスレポートを生成するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、Db2 データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. rdsadmin.create_bufferpool を呼び出して、ページサイズが 4096 の db2monbp という名前のバッファプールを作成します。詳細については、「rdsadmin.create_bufferpool」を参照してください。

```
db2 "call rdsadmin.create_bufferpool('database_name','db2monbp',4096)"
```

3. rdsadmin.create_tablespace を呼び出して、db2monbp バッファプールを使用する db2montmptbsp という名前の一時テーブルスペースを作成します。詳細については、「rdsadmin.create_tablespace」を参照してください。

```
db2 "call rdsadmin.create_tablespace('database_name',\
    'db2montmptbsp','db2monbp',4096,1000,100,'T')"
```

4. db2mon.sh スクリプトを開き、データベースへの接続に関する行を変更します。

   a. 次の行を削除します。

```
db2 -v connect to $dbName
```

b. 前のステップの行を次の行に置き換えます。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードに置き換えます。

```
db2 -v connect to $dbName user master_username using master_password
```

c. 次の行を削除します。

```
db2 -v create bufferpool db2monbp

db2 -v create user temporary tablespace db2montmptbsp bufferpool db2monbp

db2 -v drop tablespace db2montmptbsp

db2 -v drop bufferpool db2monbp
```

5. db2mon.sh スクリプトを実行して、指定した間隔でレポートを出力します。次の例では、*absolute_path* をスクリプトファイルへの完全なパスに、*rds_database_alias* をデータベースの名前に、*seconds* をレポート生成間の秒数 (0 ~ 3600) に置き換えます。

```
absolute_path/db2mon.sh rds_database_alias seconds | tee -a db2mon.out
```

例

次の例は、スクリプトファイルが perf ディレクトリの下の home ディレクトリにあることを示しています。

```
/home/db2inst1/sqllib/samples/perf/db2mon.sh rds_database_alias seconds | tee -a
 db2mon.out
```

6. db2mon.sh ファイル用に作成されたバッファプールとテーブルスペースを削除します。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードに置き換えます。*database_name* をデータベースの名前に置き換えます。詳細については、「rdsadmin.drop_tablespace」および「rdsadmin.drop_bufferpool」を参照してください。

```
db2 connect to rdsadmin user master_username using master_password

db2 "call rdsadmin.drop_tablespace('database_name','db2montmptbsp')"
```

```
db2 "call rdsadmin.drop_bufferpool('database_name','db2monbp')"
```

db2exfmt レポートの生成

db2exfmt は、RDS for Db2 データベースのクエリパフォーマンスの問題をデバッグするために使用できる、クエリパフォーマンス分析ツールです。

前提条件

db2exfmt を使用する前に、以下の要件を満たしていることを確認してください。

データベースアクセス

- db2exfmt は、ユーザーデータベースに対してのみ実行します。
- RDSADMIN に対して db2exfmt を実行することはできません。認可エラー (SQL0551N) が生成されます。

クライアントのセットアップ

- ターゲット RDS for Db2 データベースは、カタログ化されており、db2exfmt コマンドを実行するクライアントマシンからアクセス可能である必要があります。
- IBM クライアントのインストールには、db2exfmt バイナリは含まれていません。Db2 Community Edition など、Db2 の完全なインストールが必要です。

db2exfmt バイナリがクライアントマシンで使用可能であることを確認します。

```
ls -l DB2_INSTALL_PATH/bin/db2exfmt
```

db2exfmt を実行するには

1. db2exfmt ツールを実行するためのシェルスクリプト (例: db2exfmt.sh) を作成します。

```
#!/bin/bash
MYUSER=master_username        # Master user name
MYPASS=master_password        # Master user password

DBNAME=database_name          # Database name
QRY=q1.sql                    # Filename that contains the SQL statement
OUTPUT_FL=my_db2exfmt.out     # Filename where db2exfmt output will be saved

echo "select * from MASTERUSER.NAMES ;" > $QRY
```

```
db2 connect to $DBNAME user $MYUSER using $MYPASS
db2 set current query optimization = 5
db2 set current explain mode explain
db2 -tvf $QRY
db2 set current explain mode no
db2exfmt -d $DBNAME -g TIC -w -1 -n % -s % -# 0 -u $MYUSER $MYPASS -o ${OUTPUT_FL}
```

このスクリプトは以下の処理を実行します:

- RDS for Db2 データベースに接続します
- クエリ最適化レベルを設定します
- Explain モードを有効にしてクエリ実行プランをキャプチャします
- SQL クエリを実行します
- db2exfmt を実行してフォーマットされた分析レポートを生成します

db2exfmt コマンドパラメータの完全なリストと説明については、IBM Db2 ドキュメントの「db2exfmt - Explain テーブル形式コマンド」を参照してください。

2. スクリプトを実行します。

```
$ ./db2exfmt.sh
```

サンプル出力:

```
   Database Connection Information

 Database server        = DB2/LINUXX8664 11.5.9.0
 SQL authorization ID   = MASTERUS...
 Local database alias   = UHPROD01

DB20000I  The SQL command completed successfully.
DB20000I  The SQL command completed successfully.
select * from MASTERUSER.NAMES
SQL0217W  The statement was not executed as only Explain information requests
are being processed.  SQLSTATE=01604

DB20000I  The SQL command completed successfully.
```

```
DB2 Universal Database Version 11.5, 5622-044 (c) Copyright IBM Corp. 1991, 2019
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

Connecting to the Database as user masteruser.
Connect to Database Successful.
Using SYSTOOLS schema for Explain tables.
Output is in my_db2exfmt.out.
Executing Connect Reset -- Connect Reset was Successful.
```

ターゲットデータベースに SYSTOOLS.EXPLAIN* テーブルが見つからない場合は、次のコマンドを実行して Explain テーブルを作成します。

```
db2 -v CONNECT TO database_name user master_username using master_password
db2 -v "CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C', CAST (NULL AS VARCHAR(128)),
 CAST (NULL AS VARCHAR(128)))"
```

ストレージの管理

Db2 は、自動ストレージを使用して、テーブル、インデックス、一時ファイルなどのデータベースオブジェクトの物理ストレージを管理します。自動ストレージを使用すると、ストレージ領域を手動で割り当て、使用されているストレージパスを追跡する代わりに、Db2 システムによって必要に応じてストレージパスが作成および管理されます。これにより、Db2 データベースの管理を簡素化し、人為的なミスによるエラーの可能性を減らすことができます。詳細については、IBM Db2 ドキュメントの「Automatic storage」(ストレージクラス) を参照してください。

RDS for Db2 では、論理ボリュームとファイルシステムの自動拡張により、ストレージサイズを動的に増やすことができます。詳細については、「Amazon RDS DB インスタンスのストレージを使用する」を参照してください。

テーブルスペースの一般的なタスク

RDS for Db2 データベースのテーブルスペースを作成、変更、名前の変更、または削除できます。テーブルスペースを作成、変更、名前の変更、または削除するには、マスターユーザーが使用できない上位レベルの SYSADM 権限が必要です。代わりに、Amazon RDS ストアドプロシージャを使用します。

トピック

- テーブルスペースの作成
- テーブルスペースの変更

- テーブルスペースの名前変更
- テーブルスペースの削除
- テーブルスペースのステータスの確認
- テーブルスペースに関する詳細情報を返す
- テーブルスペースの状態とストレージグループの一覧表示
- テーブルのテーブルスペースの一覧表示
- テーブルスペースのコンテナの一覧表示

テーブルスペースの作成

RDS for Db2 データベースのテーブルスペースを作成するには、 `rdsadmin.create_tablespace` ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「CREATE TABLESPACE statement」を参照してください。

> **Important**
>
> テーブルスペースを作成するには、テーブルスペースを関連付ける同じページサイズのバッファプールが既に存在している必要があります。詳細については、「バッファプールの一般的なタスク」を参照してください。

テーブルスペースを作成するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、`rdsadmin` データベースに接続します。次の例で、*`master_username`* と *`master_password`* を自分の情報に置き換えます。

   ```
   db2 "connect to rdsadmin user master_username using master_password"
   ```

2. `rdsadmin.create_tablespace` を呼び出してテーブルスペースを作成します。詳細については、「rdsadmin.create_tablespace」を参照してください。

   ```
   db2 "call rdsadmin.create_tablespace(
       'database_name',
       'tablespace_name',
       'buffer_pool_name',
       tablespace_initial_size,
       tablespace_increase_size,
   ```

```
    'tablespace_type')"
```

テーブルスペースの変更

RDS for Db2 データベースのテーブルスペースを変更するには、 rdsadmin.alter_tablespace ストアドプロシージャを呼び出します。このストアドプロシージャを使用すると、テーブルスペースのバッファプールを変更したり、ハイウォーターマークを下げたり、テーブルスペースをオンラインにしたりできます。詳細については、IBM Db2 ドキュメントの「ALTER TABLESPACE statement」を参照してください。

テーブルスペースを変更するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.alter_tablespace を呼び出してテーブルスペースを変更します。詳細については、「rdsadmin.alter_tablespace」を参照してください。

```
db2 "call rdsadmin.alter_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    buffer_pool_size,
    tablespace_increase_size,
    'max_size', 'reduce_max',
    'reduce_stop',
    'reduce_value',
    'lower_high_water',
    'lower_high_water_stop',
    'switch_online')"
```

テーブルスペースの名前変更

RDS for Db2 データベースのテーブルスペース名を変更するには、rdsadmin.rename_tablespace ストアドプロシージャを呼び出します。詳細については、IBM Db2 ドキュメントの「RENAME TABLESPACE statement」を参照してください。

テーブルスペース名を変更するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.rename_tablespace を呼び出してテーブルスペース名を変更します。テーブルスペース名に関する制限などの詳細については、「rdsadmin.rename_tablespace」を参照してください。

```
db2 "call rdsadmin.rename_tablespace(
    'database_name',
    'source_tablespace_name',
    'target_tablespace_name')"
```

テーブルスペースの削除

RDS for Db2 データベースのテーブルスペースを削除するには、 rdsadmin.drop_tablespace ストアドプロシージャを呼び出します。テーブルスペースを削除する前に、まずテーブル、インデックス、ラージオブジェクト (LOB) などのテーブルスペース内のオブジェクトを削除します。詳細については、IBM Db2 ドキュメントの「Dropping table spaces」を参照してください。

テーブルスペースを削除するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 "connect to rdsadmin user master_username using master_password"
```

2. rdsadmin.drop_tablespace を呼び出してテーブルスペースを削除します。詳細については、「rdsadmin.drop_tablespace」を参照してください。

```
db2 "call rdsadmin.drop_tablespace(
    'database_name',
    'tablespace_name')"
```

テーブルスペースのステータスの確認

cast 関数を使用して、テーブルスペースのステータスを確認できます。

テーブルスペースのステータスを確認するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、Db2 データベースに接続します。次の例で、*rds_database_alias*、*master_username*、*master_password* を自分の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

2. サマリー出力を返します。

   概要出力の場合:

```
db2 "select cast(tbsp_id as smallint) as tbsp_id,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(tbsp_type as varchar(3)) as tbsp_type,
cast(tbsp_state as varchar(10)) as state,
cast(tbsp_content_type as varchar(8)) as contents from
 table(mon_get_tablespace(null,-1)) order by tbsp_id"
```

テーブルスペースに関する詳細情報を返す

cast 関数を使用して、1 人またはすべてのメンバーのテーブルスペースに関する情報を返すことができます。

テーブルスペースに関する詳細情報を返すには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、Db2 データベースに接続します。次の例で、*rds_database_alias*、*master_username*、*master_password* を自分の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

2. 1 人のメンバーまたはすべてのメンバーについて、データベース内のすべてのテーブルスペースに関する詳細を返します。

1 人のメンバーの場合:

```
db2 "select cast(member as smallint) as member,
cast(tbsp_id as smallint) as tbsp_id,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(tbsp_type as varchar(3)) as tbsp_type,
cast(tbsp_state as varchar(10)) as state,
cast(tbsp_content_type as varchar(8)) as contents,
cast(tbsp_total_pages as integer) as total_pages,
cast(tbsp_used_pages as integer) as used_pages,
cast(tbsp_free_pages as integer) as free_pages,
cast(tbsp_page_top as integer) as page_hwm,
cast(tbsp_page_size as integer) as page_sz,
cast(tbsp_extent_size as smallint) as extent_sz,
cast(tbsp_prefetch_size as smallint) as prefetch_sz,
cast(tbsp_initial_size as integer) as initial_size,
cast(tbsp_increase_size_percent as smallint) as increase_pct,
cast(storage_group_name as varchar(12)) as stogroup from
 table(mon_get_tablespace(null,-1)) order by member, tbsp_id "
```

すべてのメンバーの場合:

```
db2 "select cast(member as smallint) as member
cast(tbsp_id as smallint) as tbsp_id,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(tbsp_type as varchar(3)) as tbsp_type,
cast(tbsp_state as varchar(10)) as state,
cast(tbsp_content_type as varchar(8)) as contents,
cast(tbsp_total_pages as integer) as total_pages,
cast(tbsp_used_pages as integer) as used_pages,
cast(tbsp_free_pages as integer) as free_pages,
cast(tbsp_page_top as integer) as page_hwm,
cast(tbsp_page_size as integer) as page_sz,
cast(tbsp_extent_size as smallint) as extent_sz,
cast(tbsp_prefetch_size as smallint) as prefetch_sz,
cast(tbsp_initial_size as integer) as initial_size,
cast(tbsp_increase_size_percent as smallint) as increase_pct,
cast(storage_group_name as varchar(12)) as stogroup from
 table(mon_get_tablespace(null,-2)) order by member, tbsp_id "
```

テーブルスペースの状態とストレージグループの一覧表示

SQL ステートメントを実行することで、テーブルスペースの状態とストレージグループを一覧表示できます。

テーブルスペースの状態とストレージグループを一覧表示するには、次の SQL ステートメントを実行します。

```
db2 "SELECT varchar(tbsp_name, 30) as tbsp_name,
                  varchar(TBSP_STATE, 30) state,
                  tbsp_type,
                  varchar(storage_group_name,30) storage_group
FROM TABLE(MON_GET_TABLESPACE('',-2)) AS t"
```

テーブルのテーブルスペースの一覧表示

SQL ステートメントを実行して、テーブルのテーブルスペースを一覧表示できます。

テーブルのテーブルスペースを一覧表示するには、次の SQL ステートメントを実行します。次の例では、*SCHEMA_NAME* と *TABLE_NAME* をスキーマとテーブルの名前に置き換えます。

```
db2 "SELECT
   VARCHAR(SD.TBSPACE,30) AS DATA_SPACE,
   VARCHAR(SL.TBSPACE,30) AS LONG_SPACE,
   VARCHAR(SI.TBSPACE,30) AS INDEX_SPACE
 FROM
   SYSCAT.DATAPARTITIONS P
   JOIN SYSCAT.TABLESPACES SD ON SD.TBSPACEID = P.TBSPACEID
   LEFT JOIN SYSCAT.TABLESPACES SL ON SL.TBSPACEID = P.LONG_TBSPACEID
   LEFT JOIN SYSCAT.TABLESPACES SI ON SI.TBSPACEID = P.INDEX_TBSPACEID
 WHERE
    TABSCHEMA = 'SCHEMA_NAME'
 AND TABNAME   = 'TABLE_NAME'"
```

テーブルスペースのコンテナの一覧表示

cast コマンドを使用して、すべてのテーブルスペースコンテナまたは特定のテーブルスペースコンテナを一覧表示できます。

テーブルスペースのテーブルスペースコンテナを一覧表示するには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、Db2 データベースに接続します。次の例では、*rds_database_alias*、*master_username*、*master_password* をユーザー自身の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

2. データベース内のすべてのテーブルスペースコンテナまたは特定のテーブルスペースコンテナのリストを返します。

   すべてのテーブルスペースコンテナの場合:

```
db2 "select cast(member as smallint) as member,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(container_id as smallint) as id,
cast(container_name as varchar(60)) as container_path, container_type as type from
 table(mon_get_container(null,-2)) order by member,tbsp_id,container_id"
```

   特定のテーブルスペースコンテナの場合:

```
db2 "select cast(member as smallint) as member,
cast(tbsp_name as varchar(35)) as tbsp_name,
cast(container_id as smallint) as id,
cast(container_name as varchar(60)) as container_path, container_type as type from
 table(mon_get_container('TBSP_1',-2)) order by member, tbsp_id,container_id"
```

Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合

Amazon RDS ストアドプロシージャを使用して、Amazon RDS for Db2 DB インスタンスと Amazon Simple Storage Service (Amazon S3) バケット間でファイルを転送できます。詳細については、「Amazon RDS for Db2 ストアドプロシージャリファレンス」を参照してください。

> **Note**
>
> DB インスタンスと Amazon S3 バケットは同じ AWS リージョン に存在する必要があります。

RDS for Db2 が Amazon S3 と統合するには、RDS for Db2 がある Amazon S3 バケットに DB インスタンスがアクセスできる必要があります。この時点で S3 バケットがない場合、バケット を作成します。

トピック

- ステップ 1: IAM ポリシーを作成する
- ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする
- ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する

ステップ 1: IAM ポリシーを作成する

このステップでは、Amazon S3 バケットからお客様の RDS DB インスタンスにファイルを転送するために必要なアクセス許可を持つ AWS Identity and Access Management (IAM) ポリシーを作成します。このステップは、S3 バケットが既に作成されていることを前提としています。詳細については、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

ポリシーを作成する前に、次の情報を書き留めます。

- バケットの Amazon リソースネーム (ARN)
- ユーザーの AWS Key Management Service (AWS KMS) キーの ARN (バケットが SSE-KMS または SSE-S3 暗号化を使用している場合)

作成する IAM ポリシーには、次の情報が含まれている必要があります。*{amzn-s3-demo-bucket}* は、実際の S3 バケット名に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::${amzn-s3-demo-bucket}/*",
                "arn:aws:s3:::${amzn-s3-demo-bucket}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

IAM ポリシーは、AWS マネジメントコンソール または AWS Command Line Interface (AWS CLI) を使用して作成できます。

コンソール

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ポリシーを作成するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール https://console.aws.amazon.com/iam/ を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーを作成] を選択し、[JSON] を選択します。
4. サービスごとにアクションを追加します。Amazon S3 バケットから Amazon RDS にファイルを転送するには、バケットのアクセス許可とオブジェクトのアクセス許可を選択する必要があります。
5. [リソース] を展開します。バケットとオブジェクトのリソースを指定する必要があります。
6. [次へ] を選択します。
7. [ポリシー名] にこのポリシーの名前を入力します。
8. (オプション) [Description (説明)] に、ポリシーの説明を入力します。
9. [Create policy] (ポリシーの作成) を選択します。

AWS CLI

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ポリシーを作成するには

1. 次の JSON ポリシードキュメントを含む JSON ファイルを作成します。*{amzn-s3-demo-bucket}* は、実際の S3 バケット名に置き換えます。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "s3:PutObject",
                "s3:GetObject",
```

```
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::${amzn-s3-demo-bucket}/*",
                "arn:aws:s3:::${amzn-s3-demo-bucket}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

2. create-policy コマンドを実行します。次の例では、*iam_policy_name* と *iam_policy_file_name* を IAM ポリシーの名前と、ステップ 1 で作成した JSON ファイルの名前に置き換えます。

   Linux、macOS、Unix の場合:

```
aws iam create-policy \
    --policy-name iam_policy_name \
    --policy-document '{
      "Version": "2012-10-17"                    ,
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt",
            "s3:PutObject",
            "s3:GetObject",
```

```
            "s3:AbortMultipartUpload",
            "s3:ListBucket",
            "s3:DeleteObject",
            "s3:GetObjectVersion",
            "s3:ListMultipartUploadParts"
          ],
          "Resource": [
            "arn:aws:s3:::s3_bucket_name/*",
            "arn:aws:s3:::s3_bucket_name"
          ]
        }
      ]
    }'
```

Windows の場合:

```
aws iam create-policy ^
    --policy-name iam_policy_name ^
    --policy-document '{
      "Version": "2012-10-17"                    ,
      "Statement": [
        {
          "Effect": "Allow",
            "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:AbortMultipartUpload",
              "s3:ListBucket",
              "s3:DeleteObject",
              "s3:GetObjectVersion",
              "s3:ListMultipartUploadParts"
            ],
            "Resource": [
              "arn:aws:s3:::s3_bucket_name/*",
              "arn:aws:s3:::s3_bucket_name"
            ]
        }
      ]
    }'
```

3. ポリシーが作成されたら、ポリシーの ARN を書き留めます。[ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする] の ARN が必要です。

IAM ポリシーの作成については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする

このステップは、IAM ポリシーを ステップ 1: IAM ポリシーを作成する で作成したことを前提としています。このステップでは、RDS for Db2 DB インスタンスの IAM ロールを作成し、ロールに IAM ポリシーをアタッチします。

AWS マネジメントコンソール または AWS CLI を使用して、DB インスタンスの IAM ロールを作成できます。

コンソール

IAM ロールを作成して IAM ポリシーをアタッチするには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。
3. [ロールの作成] を選択します。
4. [信頼されたエンティティタイプ] で、[AWS のサービス] を選択します。
5. [サービスまたはユースケース] で、[RDS] を選択し、[RDS - データベースにロールを追加] を選択します。
6. [次へ] を選択します。
7. [アクセス許可ポリシー] で、作成した IAM ポリシーの名前を検索して選択します。
8. [次へ] を選択します。
9. [ロール名] に、ロールの名前を入力します。
10. (オプション) [説明] には、新しいロールの説明を入力します。
11. [ロールの作成] を選択します。

AWS CLI

IAM ロールを作成して IAM ポリシーをアタッチするには

1. 次の JSON ポリシードキュメントを含む JSON ファイルを作成します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. create-role コマンドを実行します。次の例では、*iam_role_name* と *iam_assume_role_policy_file_name* を IAM ロールの名前と、ステップ 1 で作成した JSON ファイルの名前に置き換えます。

   Linux、macOS、Unix の場合:

```
aws iam create-role \
    --role-name iam_role_name \
    --assume-role-policy-document '{
      "Version": "2012-10-17"                    ,
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

   Windows の場合:

```
aws iam create-role ^
    --role-name iam_role_name ^
    --assume-role-policy-document '{
      "Version": "2012-10-17"                    ,
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

3. ロールが作成されたら、このロールの ARN を書き留めます。ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する の ARN が必要です。
4. attach-role-policy コマンドを実行します。次の例では、*iam_policy_arn* を ステップ 1: IAM ポリシーを作成する で作成した IAM ポリシーの ARN に置き換えます。*iam_role_name* は、先ほど作成した IAM ロールの名前に置き換えます。

   Linux、macOS、Unix の場合:

   ```
   aws iam attach-role-policy \
      --policy-arn iam_policy_arn \
      --role-name iam_role_name
   ```

   Windows の場合:

   ```
   aws iam attach-role-policy ^
      --policy-arn iam_policy_arn ^
      --role-name iam_role_name
   ```

詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する

このステップでは、IAM ロールを RDS for Db2 DB インスタンスに追加します。次の要件に注意してください。

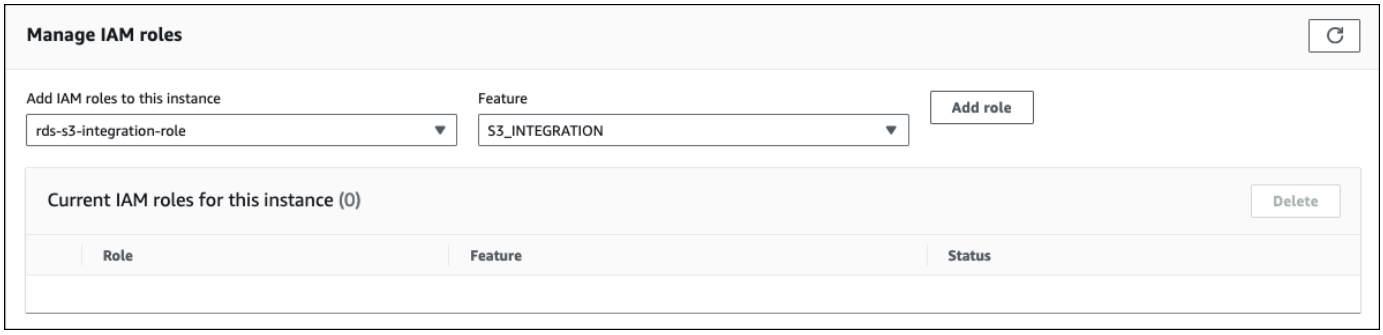
- 必須の Amazon S3 アクセス許可ポリシーがアタッチされた IAM ロールへのアクセスが許可されている必要があります。
- RDS for Db2 DB インスタンスには、一度に 1 つの IAM ロールのみを関連付けることができます。
- RDS for Db2 DB インスタンスは、[使用可能] の状態である必要があります。

IAM ロールを DB インスタンスに追加するには、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

RDS for Db2 DB インスタンスに IAM ロールを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. RDS for Db2 DB インスタンス名を選択します。
4. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、次のいずれかを実行します。
5. [このインスタンスに IAM ロールを追加] で、ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする で作成したロールを選択します。
6. [機能] で、[S3_INTEGRATION] を選択します。
7. [Add role] を選択します。



AWS CLI

IAM ロールを RDS for Db2 DB インスタンスに追加するには、add-role-to-db-instance コマンドを実行します。次の例では、*region*、*db_instance_name*、および *iam_role_arn* を、DB インスタンスが存在する AWS リージョン の名前、DB インスタンスの名前、および「ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする」で作成した IAM ロールの ARN に置き換えます。

Linux、macOS、Unix の場合:

```
aws rds add-role-to-db-instance \
    --region $region \
    --db-instance-identifier $db_instance_name \
    --feature-name S3_INTEGRATION \
    --role-arn $iam_role_arn \
```

Windows の場合:

```
aws rds add-role-to-db-instance ^
    --region $region \
    --db-instance-identifier db_instance_name ^
    --feature-name S3_INTEGRATION ^
    --role-arn iam_role_arn ^
```

ロールが RDS for Db2 DB インスタンスに正常に追加されたことを確認するには、describe-db-instances コマンドを実行します。次の例では、*db_instance_name* を DB インスタンスの名前に置き換えます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
    --filters "Name=db-instance-id,Values=db_instance_name" \
    --query 'DBInstances[].AssociatedRoles'
```

Windows の場合:

```
aws rds describe-db-instances ^
    --filters "Name=db-instance-id,Values=db_instance_name" ^
    --query 'DBInstances[].AssociatedRoles'
```

このコマンドでは、次の例のような出力が生成されます。

```
[
    [
        {
            "RoleArn": "arn:aws:iam::0123456789012:role/rds-db2-s3-role",
            "FeatureName": "S3_INTEGRATION",
            "Status": "ACTIVE"
        }
    ]
]
```

Amazon RDS for Db2 へのデータの移行

AWS またはネイティブ Db2 ツールのいずれかを使用して、セルフマネージド Db2 データベースを Amazon RDS for Db2 に移行できます。

AWS のサービスを使用して Db2 データベースから Amazon RDS for Db2 に移行する方法については、「AWS のサービスを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。

ネイティブ Db2 ツールを使用して Db2 データベースから Amazon RDS for Db2 に移行する方法については、「ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。

AWS のサービスを使用して Db2 から Amazon RDS for Db2 にデータを移行する

Amazon RDS では、Db2 データベースから Amazon RDS for Db2 にデータを移行するには、いくつかの方法があります。Linux、AIX、または Windows 環境から Amazon RDS for Db2 への Db2 データベースの 1 回限りの移行を実行できます。ダウンタイムを最小限に抑えるため、ほぼゼロのダウンタイムでの移行を実行できます。データを Amazon S3 に保存し、Db2 データベースに一度に 1 つのテーブルをロードすることで、データを移行できます。レプリケーションや AWS Database Migration Service を使用して同期移行を実行することもできます。

Linux ベースの Db2 データベースの 1 回限りの移行の場合、Amazon RDS はオフラインバックアップとオンラインバックアップのみをサポートします。Amazon RDS は、増分バックアップおよび Delta バックアップをサポートしていません。Linux ベースの Db2 データベースをほぼゼロのダウンタイムで移行する場合、Amazon RDS にはオンラインバックアップが必要です。ほぼゼロのダウンタイムでの移行にはオンラインバックアップを使用し、ダウンタイムに対処できる移行にはオフラインバックアップを使用することをお勧めします。

トピック

- Amazon RDS for Db2 の Linux から Linux への移行
- Amazon RDS for Db2 のダウンタイムがほぼゼロの Linux から Linux への移行
- Amazon RDS for Db2 の Linux から Linux への同期移行
- Amazon RDS for Db2 の AIX または Windows から Linux への移行
- Amazon S3 から Amazon RDS for Db2 への Db2 データの移行

- AWS Database Migration Service (AWS DMS) を使用した Amazon RDS for Db2 への移行

Amazon RDS for Db2 の Linux から Linux への移行

この移行アプローチでは、セルフマネージド Db2 データベースを Amazon S3 バケットにバックアップします。次に、Amazon RDS ストアドプロシージャを使用して、Db2 データベースを Amazon RDS for Db2 DB インスタンスに復元します。Amazon S3 の使用の詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。

RDS for Db2 のバックアップと復元は、IBM Db2 によってサポートされているアップグレードパスと制限に従います。詳細については、 IBM Db2 ドキュメントの「Db2 サーバーについてサポートされているアップグレードパス」と「Db2 サーバーのアップグレード制限」を参照してください。

トピック

- ネイティブ復元の使用に関する制限と推奨事項
- データベースを Amazon S3 にバックアップする
- デフォルトの自動ストレージグループを作成する
- Db2 データベースの復元

ネイティブ復元の使用に関する制限と推奨事項

ネイティブ復元を使用する場合は、次の制限と推奨事項が適用されます。

- Amazon RDS は、サポートされている RDS for Db2 バージョンと一致するオンプレミスバージョンの Db2 の移行のみをサポートします。サポートされているバージョンの詳細については、「Amazon RDS Db2 インスタンスのアップグレード管理」を参照してください。
- Amazon RDS では、ネイティブ復元にオフラインおよびオンラインのバックアップのみがサポートされています。Amazon RDS は、増分バックアップまたは Delta バックアップをサポートしていません。
- RDS for Db2 DB インスタンスがあるリージョンとは異なる AWS リージョンのAmazon S3 バケットから復元することはできません。
- Amazon S3 では、Amazon S3 バケットにアップロードするファイルのサイズが 5 TB に制限されます。データベースバックアップファイルが 5 TB を超える場合は、バックアップファイルを小さいファイルに分割します。
- Amazon RDS は、非 fenced 外部ルーチン、増分復元、または Delta 復元をサポートしていません。

- 暗号化したソースデータベースから復元することはできませんが、暗号化された Amazon RDS DB インスタンスには復元できます。

復元プロセスは、設定によって異なります。

USE_STREAMING_RESTORE を TRUE に設定すると、Amazon RDS は復元中に S3 バケットから直接バックアップをストリーミングします。ストリーミングはストレージ要件を大幅に削減します。バックアップのサイズまたは元のデータベースのサイズのうち、いずれか大きい方のサイズ以上のストレージスペースをプロビジョニングするだけで済みます。

USE_STREAMING_RESTORE を FALSE に設定すると、Amazon RDS はまず RDS for Db2 DB インスタンスにバックアップをダウンロードし、次にバックアップを抽出します。抽出には追加のストレージスペースが必要です。バックアップのサイズと元のデータベースのサイズの合計以上のストレージスペースをプロビジョニングする必要があります。

復元されたデータベースの最大サイズは、サポートされているデータベースの最大サイズから、復元プロセス中の一時ストレージに必要なスペースを引いた値に等しくなります。

データベースを Amazon S3 にバックアップする

Amazon S3 でデータベースをバックアップするには、次の AWS コンポーネントが必要です。

- バックアップファイルを保存する Amazon S3 バケット: Amazon RDS に移行するバックアップファイルをアップロードします。ダウンタイムに対処できる移行には、オフラインバックアップを使用することをお勧めします。S3 バケットが既にある場合はそのバケットを使用できます。S3 バケットがない場合は、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

  > **Note**
  >
  > データベースが大きく、S3 バケットへの転送に時間がかかる場合は、 AWS Snow Family デバイスを注文して、AWS にバックアップの実行を依頼できます。ファイルをデバイスにコピーして Snow ファミリーチームに返すと、チームはバックアップしたイメージを S3 バケットに転送します。詳細については、「AWS Snow Family ドキュメント」を参照してください。

- S3 バケットにアクセスするための IAM ロール: IAM ロールが既にある場合は、そのロールを使用できます。ロールがない場合は、「ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする」を参照してください。

- IAM ロールにアタッチされた信頼関係とアクセス許可を持つ IAM ポリシー: 詳細については、「ステップ 1: IAM ポリシーを作成する」を参照してください。
- RDS for Db2 DB インスタンスに追加された IAM ロール: 詳細については、「ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する」を参照してください。

デフォルトの自動ストレージグループを作成する

ソースデータベースにはデフォルトの自動ストレージグループが必要です。データベースにデフォルトの自動ストレージグループがない場合は、作成する必要があります。

デフォルトの自動ストレージグループを作成するには

1. ソースデータベースに接続します。次の例では、*source_database* をデータベースの名前に置き換えます。

```
db2 connect to source_database
```

2. 自動ストレージグループを作成し、デフォルトとして設定します。次の例では、*storage_path* をストレージグループが配置されている場所への絶対パスに置き換えます。

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. バックエンドプロセスを終了します。

```
db2 terminate
```

4. データベースを非アクティブ化し、すべてのデータベースサービスを停止します。次の例では、*source_database* を、ストレージグループを作成したデータベースの名前に置き換えます。

```
db2 deactivate db source_database
```

5. データベースをバックアップします。次の例では、*source_database* を、ストレージグループを作成したデータベースの名前に置き換えます。*file_system_path* を、データベースをバックアップする絶対パスに置き換えます。

```
db2 backup database source_database to file_system_path
```

Db2 データベースの復元

Amazon S3 でデータベースをバックアップし、自動ストレージグループを作成したら、Db2 データベースを RDS for Db2 DB インスタンスに復元できます。

Db2 データベースを Amazon S3 バケットから RDS for Db2 DB インスタンスに復元するには

1. RDS for Db2 DB インスタンスに接続します。詳細については、「Db2 DB インスタンスへの接続」を参照してください。
2. (オプション) データベースが最適な設定で構成されていることを確認するには、the section called “rdsadmin.show_configuration” を呼び出して次のパラメータの値を確認します。

   - `RESTORE_DATABASE_NUM_BUFFERS`
   - `RESTORE_DATABASE_PARALLELISM`
   - `RESTORE_DATABASE_NUM_MULTI_PATHS`
   - `USE_STREAMING_RESTORE`

   必要に応じて、the section called “rdsadmin.set_configuration” を使用してこれらの値を変更します。これらのパラメータを適切に設定することで、大量のデータを含むデータベースを復元する際のパフォーマンスが大幅に向上します。ほとんどの移行シナリオでは、ストレージ要件が軽減され、復元速度が向上するため、`USE_STREAMING_RESTORE` を `TRUE` に設定することをお勧めします。
3. `rdsadmin.restore_database` を呼び出してデータベースを復元します。詳細については、「rdsadmin.restore_database」を参照してください。

Amazon RDS for Db2 のダウンタイムがほぼゼロの Linux から Linux への移行

この移行アプローチでは、 Linux ベース Db2 データベースを単一のセルフマネージド Db2 データベース (ソース) から Amazon RDS for Db2 に移行します。このアプローチにより、アプリケーションまたはユーザーの停止またはダウンタイムが最小限に抑えられるか、まったく発生しません。このアプローチでは、データベースをバックアップし、ログの再生で復元します。これにより、進行中のオペレーションの中断を防ぎ、データベースの高可用性を実現できます。

RDS for Db2 は、ほぼゼロのダウンタイムでの移行を達成できるように、ログの再生による復元を実装しています。このアプローチでは、セルフマネージド Linux ベース Db2 データベースのバックアップを作成し、RDS for Db2 サーバーに復元します。Amazon RDS ストアドプロシージャでは、後続のトランザクションログを適用してデータベースを最新の状態にします。

トピック

- ほぼゼロのダウンタイムでの移行の制限と推奨事項
- データベースを Amazon S3 にバックアップする
- デフォルトの自動ストレージグループを作成する
- Db2 データベースの移行

ほぼゼロのダウンタイムでの移行の制限と推奨事項

ほぼゼロのダウンタイムでの移行の使用には、次の制限と推奨事項が適用されます。

- Amazon RDS では、移行のダウンタイムをほぼゼロにするために、オンラインバックアップが必要です。これは、アーカイブされたトランザクションログをアップロードすると、Amazon RDS によってデータベースがロールフォワード保留状態のままになるためです。詳細については、「the section called “Db2 データベースの移行”」を参照してください。
- RDS for Db2 DB インスタンスがあるリージョンとは異なる AWS リージョンのAmazon S3 バケットから復元することはできません。
- Amazon S3 では、S3 バケットにアップロードするファイルのサイズが 5 TB に制限されます。データベースバックアップファイルが 5 TB を超える場合は、バックアップファイルを小さいファイルに分割します。
- Amazon RDS は、非 fenced 外部ルーチン、増分復元、または Delta 復元をサポートしていません。
- 暗号化したソースデータベースから復元することはできませんが、暗号化された Amazon RDS DB インスタンスには復元できます。

復元プロセスは、設定によって異なります。

USE_STREAMING_RESTORE を TRUE に設定すると、Amazon RDS は復元中に S3 バケットから直接バックアップをストリーミングします。ストリーミングはストレージ要件を大幅に削減します。バックアップのサイズまたは元のデータベースのサイズのうち、いずれか大きい方のサイズ以上のストレージスペースをプロビジョニングするだけで済みます。

USE_STREAMING_RESTORE を FALSE に設定すると、Amazon RDS はまず RDS for Db2 DB インスタンスにバックアップをダウンロードし、次にバックアップを抽出します。抽出には追加のストレージスペースが必要です。バックアップのサイズと元のデータベースのサイズの合計以上のストレージスペースをプロビジョニングする必要があります。

復元されたデータベースの最大サイズは、サポートされているデータベースの最大サイズから、復元プロセス中の一時ストレージに必要なスペースを引いた値に等しくなります。

データベースを Amazon S3 にバックアップする

Amazon S3 でデータベースをバックアップするには、次の AWS コンポーネントが必要です。

- バックアップファイルを保存する Amazon S3 バケット: Amazon RDS に移行するバックアップファイルをアップロードします。Amazon RDS では、移行のダウンタイムをほぼゼロにするために、オンラインバックアップが必要です。S3 バケットが既にある場合はそのバケットを使用できます。S3 バケットがない場合は、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

  > **Note**
  >
  > データベースが大きく、S3 バケットへの転送に時間がかかる場合は、 AWS Snow Family デバイスを注文して、AWS にバックアップの実行を依頼できます。ファイルをデバイスにコピーして Snow ファミリーチームに返すと、チームはバックアップしたイメージを S3 バケットに転送します。詳細については、「AWS Snow Family ドキュメント」を参照してください。

- S3 バケットにアクセスするための IAM ロール: AWS Identity and Access Management (IAM) ロールが既にある場合は、そのロールを使用できます。ロールがない場合は、「ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする」を参照してください。
- IAM ロールにアタッチされた信頼関係とアクセス許可を持つ IAM ポリシー: 詳細については、「ステップ 1: IAM ポリシーを作成する」を参照してください。
- RDS for Db2 DB インスタンスに追加された IAM ロール: 詳細については、「ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する」を参照してください。

デフォルトの自動ストレージグループを作成する

ソースデータベースにはデフォルトの自動ストレージグループが必要です。データベースにデフォルトの自動ストレージグループがない場合は、作成する必要があります。

デフォルトの自動ストレージグループを作成するには

1. ソースデータベースに接続します。次の例では、*source_database* をデータベースの名前に置き換えます。

```
db2 connect to source_database
```

2. 自動ストレージグループを作成し、デフォルトとして設定します。次の例では、*storage_path* をストレージグループが配置されている場所への絶対パスに置き換えます。

```
db2 "create stogroup IBMSTOGROUP ON storage_path set as default"
```

3. バックエンドプロセスを終了します。

```
db2 terminate
```

Db2 データベースの移行

ほぼゼロのダウンタイムでの移行をセットアップしたら、Db2 データベースを Amazon S3 バケットから RDS for Db2 DB インスタンスに移行できます。

Amazon S3 バケットから RDS for Db2 DB インスタンスへのバックアップファイルのダウンタイムをほぼゼロにするには

1. ソースデータベースのオンラインバックアップを実行します。詳細については、IBM Db2 ドキュメントの「BACKUP DATABASE command」をご参照ください。
2. データベースのバックアップを Amazon S3 バケットにコピーします。Amazon S3 の使用の詳細については、「Amazon Simple Storage Service ユーザーガイド」を参照してください。
3. RDS for Db2 DB インスタンスの *master_username* と *master_password* を使用して rdsadmin サーバーに接続します。

```
db2 connect to rdsadmin user master_username using master_password
```

4. (オプション) データベースが最適な設定で構成されていることを確認するには、the section called “rdsadmin.show_configuration” を呼び出して次のパラメータの値を確認します。

   - RESTORE_DATABASE_NUM_BUFFERS
   - RESTORE_DATABASE_PARALLELISM
   - RESTORE_DATABASE_NUM_MULTI_PATHS
   - USE_STREAMING_RESTORE

必要に応じて、the section called “rdsadmin.set_configuration” を使用してこれらの値を変更します。これらのパラメータを適切に設定することで、大量のデータを含むデータベースを復元する際のパフォーマンスが大幅に向上します。ほとんどの移行シナリオでは、ストレージ要件が軽減され、復元速度が向上するため、USE_STREAMING_RESTORE を TRUE に設定することをお勧めします。

5. rdsadmin.restore_database を呼び出して、RDS for Db2 サーバーでバックアップを復元します。backup_type を ONLINE に設定します。詳細については、「rdsadmin.restore_database」を参照してください。
6. ソースサーバーから S3 バケットにアーカイブログをコピーします。詳細については、IBM Db2 ドキュメントに記載の「Archive logging」を参照してください。
7. rdsadmin.rollforward_database を呼び出して、アーカイブログを必要な回数だけ適用します。データベースを ROLL-FORWARD PENDING 状態に保つには、complete_rollforward を FALSE に設定します。詳細については、「rdsadmin.rollforward_database」を参照してください。
8. すべてのアーカイブログを適用したら、rdsadmin.complete_rollforward を呼び出してデータベースをオンラインにします。詳細については、「rdsadmin.complete_rollforward」を参照してください。
9. データベースのアプリケーションエンドポイントを更新するか、DNS エンドポイントを更新して RDS for Db2 サーバーにトラフィックをリダイレクトして、アプリケーション接続を RDS for Db2 サーバーに切り替えます。RDS for Db2 データベースエンドポイントを使用して、セルフマネージド Db2 データベースで Db2 自動クライアント再ルーティング機能を使用することもできます。詳細については、 IBM Db2 ドキュメントの「Automatic client reroute description and setup」を参照してください。
10. (オプション) ソースデータベースをシャットダウンします。

Amazon RDS for Db2 の Linux から Linux への同期移行

この移行アプローチでは、セルフマネージド Db2 データベースと Amazon RDS for Db2 DB インスタンスとの間のレプリケーションを設定します。セルフマネージドデータベースに加えられた変更は、ほぼリアルタイムで RDS for Db2 DB インスタンスにレプリケートされます。このアプローチにより、継続的な可用性が実現し、移行プロセス中のダウンタイムが最小限に抑えられます。

Amazon RDS for Db2 の AIX または Windows から Linux への移行

この移行アプローチでは、ネイティブ Db2 ツールを使用して、セルフマネージド Db2 データベースを Amazon S3 バケットにバックアップします。ネイティブ Db2 ツールには、 `export` ユーティリティ、 `db2move` システムコマンド、または `db2look` システムコマンドが含まれます。Db2 データベースは、セルフマネージド型または Amazon Elastic Compute Cloud (Amazon EC2) のいずれかです。AIX または Windows システムから Amazon S3 バケットにデータを移行できます。次に、Db2 クライアントを使用して、S3 バケットから Amazon RDS for Db2 データベースにデータを直接ロードします。ダウンタイムは、データベースのサイズによって異なります。Amazon S3 の使用の詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。

Db2 データベースを RDS for Db2 に移行するには

1. データベースをバックアップする準備をします。セルフマネージド Db2 システムでバックアップを保持するのに十分なストレージ量を設定します。
2. データベースをバックアップします。
   a. db2look システムコマンドを実行して、すべてのオブジェクトのデータ定義言語 (DDL) ファイルを抽出します。
   b. Db2 エクスポートユーティリティ、db2move システムコマンド 、または CREATE EXTERNAL TABLE ステートメントを実行して、Db2 テーブルデータを Db2 システムのストレージにアンロードします。
3. バックアップを Amazon S3 バケットに移動します。詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。

   > **Note**
   >
   > データベースが大きく、S3 バケットへの転送に時間がかかる場合は、 AWS Snow Family デバイスを注文して、AWS にバックアップの実行を依頼できます。ファイルをデバイスにコピーして Snow ファミリーチームに返すと、チームはバックアップしたイメージを S3 バケットに転送します。詳細については、「AWS Snow Family ドキュメント」を参照してください。

4. Db2 クライアントを使用して、S3 バケットから RDS for Db2 データベースにデータを直接ロードします。詳細については、「Amazon S3 による移行」を参照してください。

Amazon S3 から Amazon RDS for Db2 への Db2 データの移行

この移行アプローチでは、まず 1 つのテーブルから Amazon S3 バケットに配置するデータファイルにデータを保存します。次に、LOAD コマンドを使用して、そのデータファイルから Amazon RDS for Db2 データベースのテーブルにデータをロードします。Amazon S3 の使用の詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。

トピック

- Amazon S3 にデータを保存する
- RDS for Db2 テーブルへのデータのロード

Amazon S3 にデータを保存する

単一のテーブルから Amazon S3 にデータを保存するには、データベースユーティリティを使用して、データベース管理システム (DBMS) から CSV ファイルにデータを抽出します。次に、各ファイルを Amazon S3 にアップロードします。

Amazon S3 にデータファイルを保存するには、次の AWS コンポーネントが必要です。

- バックアップファイルを保存する Amazon S3 バケット: S3 バケットが既にある場合は、そのバケットを使用できます。S3 バケットがない場合は、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。
- S3 バケットにアクセスするための IAM ロール: IAM ロールが既にある場合は、そのロールを使用できます。ロールがない場合は、「ステップ 2: IAM ロールを作成して IAM ポリシーをアタッチする」を参照してください。
- IAM ロールにアタッチされた信頼関係とアクセス許可を持つ IAM ポリシー: 詳細については、「ステップ 1: IAM ポリシーを作成する」を参照してください。
- RDS for Db2 DB インスタンスに追加された IAM ロール: 詳細については、「ステップ 3: RDS for Db2 DB インスタンスに IAM ロールを追加する」を参照してください。

RDS for Db2 テーブルへのデータのロード

データファイルを Amazon S3 に保存した後、これらのファイルから RDS for Db2 DB インスタンスの個々のテーブルにデータをロードできます。

Db2 テーブルデータを RDS for Db2 DB データベーステーブルにロードするには

1. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次の例で、*master_username* と *master_password* を自分の情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

2. 保存したファイルが保存されている Amazon S3 バケットを指すストレージアクセスエイリアスをカタログ化します。次のステップで使用するこのエイリアスの名前を書き留めます。同じ Amazon S3 バケットに保存されているデータファイルから複数のテーブルをロードする予定の場合、このステップを 1 回だけ実行する必要があります。

   次の例では、*jorge_souza* という名前のユーザーに *amzn-s3-demo-bucket* という名前のバケットへのアクセスを許可する *my_s3_alias* という名前のエイリアスをカタログ化します。

```
db2 "call rdsadmin.catalog_storage_access(?, 'my_s3_alias', 'amzn-s3-demo-bucket',
 'USER', 'jorge_souza')"
```

   このストアドプロシージャの詳細については、「the section called “rdsadmin.catalog_storage_access”」を参照してください。

3. Amazon S3 バケットを指すストレージアクセスエイリアスを使用して、LOAD コマンドを実行します。

> **Note**
>
> LOAD コマンドがエラーを返す場合は、Amazon S3 の VPC ゲートウェイエンドポイントを作成し、セキュリティグループにアウトバウンドルールを追加する必要がある場合があります。詳細については、「the section called “ファイル I/O エラー”」を参照してください。

   次の例では、*my_s3_datafile.csv* という名前のデータファイルから *my_db2_table* という名前のテーブルにデータをロードします。この例では、データファイルが *my_s3_alias* という名前のエイリアスが指す Amazon S3 バケットにあることを前提としています。

```
db2 "load from db2remote://my_s3_alias//my_s3_datafile.csv of DEL insert
 into my_db2_table";
```

次の例では、*my_table1_export.ixf* という名前のデータファイルから *my_db2_table* という名前のテーブルに LOB をロードします。この例では、データファイルが *my_s3_alias* という名前のエイリアスが指す Amazon S3 バケットにあることを前提としています。

```
db2 "call sysproc.admin_cmd('load from
 "db2remote://my_s3_alias//my_table1_export.ixf" of ixf
        lobs from "db2remote://my_s3_alias//" xml from "db2remote://my_s3_alias//"
        modified by lobsinfile implicitlyhiddeninclude identityoverride
 generatedoverride periodoverride transactionidoverride
        messages on server
        replace into "my_schema"."my_db2_table"
                                nonrecoverable
        indexing mode incremental allow no access')"
```

RDS for Db2 DB インスタンスのテーブルにロードする Amazon S3 バケット内のデータファイルごとに、このステップを繰り返します。

LOAD コマンドの詳細については、「LOAD コマンド」を参照してください。

AWS Database Migration Service (AWS DMS) を使用した Amazon RDS for Db2 への移行

AWS DMS を 1 回限りの移行に使用し、Linux、Unix (AIX など)、Windows の Db2 から Amazon RDS for Db2 に同期できます。詳細については、「What is AWS Database Migration Service?」を参照してください。

ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する

複数のネイティブ Db2 ツール、ユーティリティ、およびコマンドを使用して、Db2 データベースから Amazon RDS for Db2 データベースにデータを直接移動できます。これらのネイティブ Db2 ツールを使用するには、クライアントマシンを RDS for Db2 DB インスタンスに接続できる必要があります。詳細については、「クライアントマシンを Amazon RDS for Db2 DB インスタンスに接続する」を参照してください。

> **Note**
>
> データを移動するもう 1 つの方法は、まず Amazon S3 バケットに保存してから、LOAD コマンドを使用して、そのデータを RDS for Db2 データベースのテーブルに転送することです。この方法は、RDS for Db2 と S3 間のネットワーク接続が良好であるため、大量のデータを移行する場合に最適なパフォーマンスを提供します。詳細については、「the section called “Amazon S3 による移行”」を参照してください。

ツール名	ユースケース	制限
db2look	セルフマネージド Db2 データベースから RDS for Db2 データベースにメタデータをコピーする。	• バッファプールの作成、テーブルスペースの作成、ロールの作成の構文を、RDS for Db2 ストアドプロシージャ で使用される構文と一致するように変更する必要がある。
IMPORT コマンド	小さなテーブルとラージオブジェクト (LOB) を含むテーブルをクライアントマシンから RDS for Db2 DB インスタンスに移行する。	• INSERT および DELETE ログ記録オペレーションのため、LOAD ユーティリティよりも低速。 • ネットワーク帯域幅が限られているためパフォーマンスが劣る。
INGEST ユーティリティ	クライアントマシンでラージオブジェクト (LOB) がないファイルやパイプから RDS for Db2 DB インスタンスにデータを継続的にストリーミングする。INSERT および MERGE オペレーションをサポートする。	• LOB を含むデータファイルはストリーミングできない。代わりに IMPORT コマンドを使用する。 • セルフマネージド Db2 データベースと RDS for Db2 データベース間の接続が必要。

ツール名	ユースケース	制限
INSERT コマンド	セルフマネージド Db2 データベースから RDS for Db2 データベースに小さなテーブルのデータをコピーする。	• セルフマネージド Db2 データベースと RDS for Db2 データベース間の接続が必要。 • ネットワーク帯域幅が限られているためパフォーマンスが劣る。
LOAD CLIENT コマンド	ラージオブジェクト (LOB) のない小さなテーブルをクライアントマシンから RDS for Db2 DB インスタンスに移行する。	• LOB を含むデータファイルは移行できない。代わりに IMPORT コマンドを使用する。 • ネットワーク帯域幅が限られているためパフォーマンスが劣る。

クライアントマシンを Amazon RDS for Db2 DB インスタンスに接続する

ネイティブ Db2 ツールのいずれかを使用して Db2 データベースから Amazon RDS for Db2 データベースにデータを移動するには、まずクライアントマシンを RDS for Db2 DB インスタンスに接続する必要があります。

以下のいずれかのクライアントマシンを使用できます。

- Linux、Windows、または macOS 上の Amazon Elastic Compute Cloud (Amazon EC2) インスタンス このインスタンスは、RDS for Db2 DB インスタンス、AWS Cloud9 または AWS CloudShell と同じ仮想プライベートクラウド (VPC) にある必要があります。
- Amazon EC2 インスタンスのセルフマネージド Db2 インスタンス。インスタンスは同じ VPC 内にある必要があります。
- Amazon EC2 インスタンスのセルフマネージド Db2 インスタンス。VPC ピアリングが有効になっている場合は、インスタンスが別の VPC にあってもかまいません。詳細については、「Amazon Virtual Private Cloud ピアリング接続ガイド」の「Create a VPC peering connection」を参照してください。

- セルフマネージド環境で Linux、Windows、または macOS を実行しているローカルマシン。RDS for Db2 へのパブリック接続があるか、セルフマネージド Db2 インスタンスと AWS 間の VPN 接続を有効にする必要があります。

クライアントマシンを RDS for Db2 DB インスタンスに接続するには、IBM Db2 Data Management Console を使用してクライアントマシンにログインします。詳細については、「Amazon RDS DB インスタンスの作成」および「IBM Db2 Data Management Console」を参照してください。

AWS Database Migration Service (AWS DMS) を使用して、データベースに対してクエリを実行し、SQL 実行プランを実行し、データベースをモニタリングできます。詳細については、「AWS Database Migration Service ユーザーガイド」の「What is AWS Database Migration Service?」を参照してください。

クライアントマシンを RDS for Db2 DB インスタンスに正常に接続したら、ネイティブ Db2 ツールを使用してデータをコピーできます。詳細については、「ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。

db2look を使用して Db2 から Amazon RDS for Db2 にデータベースメタデータをコピーする

db2look は、データ定義言語 (DDL) ファイル、オブジェクト、認証、設定、WLM、およびデータベースレイアウトを抽出するネイティブ Db2 ツールです。db2look を使用して、セルフマネージド Db2 データベースから Amazon RDS for Db2 データベースにデータベースメタデータをコピーできます。詳細については、IBM Db2 ドキュメントの「Mimicking databases using db2look」を参照してください。

データベースメタデータをコピーするには

1. セルフマネージド Db2 システムで db2look ツールを実行し、DDL ファイルを抽出します。次の例では、*database_name* を Db2 データベースの名前に置き換えます。

```
db2look -d database_name -e -l -a -f -wlm -cor -createdb -printdbcfg -o db2look.sql
```

2. クライアントマシンがソース (セルフマネージド Db2) データベースと RDS for Db2 DB インスタンスにアクセスできる場合は、リモートインスタンスに直接アタッチすることで、クライアントマシンで db2look.sql ファイルを作成できます。次に、リモートのセルフマネージド Db2 インスタンスをカタログ化します。

a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレス、セルフマネージド Db2 データベースのポート番号に置き換えます。

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

b. データベースをカタログ化します。次の例では、*source_database_name* と *source_database_alias* を、セルフマネージド Db2 データベースの名前と、このデータベースに使用するエイリアスに置き換えます。

```
db2 catalog database source_database_name as source_database_alias at node
 srcnode \
    authentication server_encrypt
```

c. ソースデータベースにアタッチします。次の例では、*source_database_alias*、*user_id*、*user_password* を、前のステップで作成したエイリアスと、セルフマネージド Db2 データベースのユーザー ID とパスワードに置き換えます。

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
    -cor -createdb -printdbcfg -o db2look.sql
```

3. クライアントマシンからリモートのセルフマネージド Db2 データベースにアクセスできない場合は、`db2look.sql` ファイルをクライアントマシンにコピーします。 次に、RDS for Db2 DB インスタンスをカタログ化します。

a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレスと、RDS for Db2 DB インスタンスのポート番号に置き換えます。

```
db2 catalog tcpip node remnode REMOTE dns_ip_address server port
```

b. データベースをカタログ化します。次の例では、*rds_database_name* と *rds_database_alias* を、RDS for Db2 データベースの名前と、このデータベースに使用するエイリアスに置き換えます。

```
db2 catalog database rds_database_name as rds_database_alias at node remnode \
    authentication server_encrypt
```

c. RDS for Db2 を管理する管理データベースをカタログ化します。このデータベースを使用してデータを保存することはできません。

```
db2 catalog database rdsadmin as rdsadmin at node remnode authentication
 server_encrypt
```

4. バッファプールとテーブルスペースを作成します。管理者には、バッファプールまたはテーブルスペースを作成する権限がありません。ただし、Amazon RDS ストアドプロシージャを使用して作成することができます。

   a. db2look.sql ファイル内のバッファプールとテーブルスペースの名前と定義を見つけます。

   b. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して Amazon RDS に接続します。次の例では、*master_username* と *master_password* をユーザー自身の情報に置き換えます。

   ```
   db2 connect to rdsadmin user master_username using master_password
   ```

   c. rdsadmin.create_bufferpool を呼び出してバッファプールを作成します。詳細については、「rdsadmin.create_bufferpool」を参照してください。

   ```
   db2 "call rdsadmin.create_bufferpool(
       'database_name',
       'buffer_pool_name',
       buffer_pool_size,
       'immediate',
       'automatic',
       page_size,
       number_block_pages,
       block_size)"
   ```

   d. rdsadmin.create_tablespace を呼び出してテーブルスペースを作成します。詳細については、「rdsadmin.create_tablespace」を参照してください。

   ```
   db2 "call rdsadmin.create_tablespace(
       'database_name',
       'tablespace_name',
       'buffer_pool_name',
       tablespace_initial_size,
       tablespace_increase_size,
       'tablespace_type')"
   ```

e. 追加するバッファプールまたはテーブルスペースごとに、ステップ c または d を繰り返します。

f. 接続を終了します。

```
db2 terminate
```

5. テーブルとオブジェクトを作成します。

a. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、RDS for Db2 データベースに接続します。次の例では、*rds_database_name*、*master_username*、*master_password* をユーザー自身の情報に置き換えます。

```
db2 connect to rds_database_name user master_username using master_password
```

b. db2look.sql ファイルを実行します。

```
db2 -tvf db2look.sql
```

c. 接続を終了します。

```
db2 terminate
```

IMPORT コマンドを使用してクライアントマシンから Amazon RDS for Db2 にデータをインポートする

クライアントマシンで IMPORT コマンドを使用して、Amazon RDS for Db2 サーバーにデータをインポートできます。

> **Important**
>
> IMPORT コマンドメソッドは、小さなテーブルやラージオブジェクト (LOB) を含むテーブルを移行する場合に便利です。IMPORT コマンドは、INSERT および DELETE のログ記録オペレーションのため、LOAD ユーティリティよりも低速です。クライアントマシンと RDS for Db2 間のネットワーク帯域幅が制限されている場合は、別の移行アプローチを使用することをお勧めします。詳細については、「ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。

RDS for Db2 サーバーにデータをインポートするには

1. IBM Db2 Data Management Console を使用してクライアントマシンにログインします。詳細については、「IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続する」を参照してください。
2. クライアントマシン上の RDS for Db2 データベースをカタログ化します。

   a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレス、セルフマネージド Db2 データベースのポート番号に置き換えます。

   ```
   db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
   ```

   b. データベースをカタログ化します。次の例では、*source_database_name* と *source_database_alias* を、セルフマネージド Db2 データベースの名前と、このデータベースに使用するエイリアスに置き換えます。

   ```
   db2 catalog database source_database_name as source_database_alias at node
    srcnode \
       authentication server_encrypt
   ```

3. ソースデータベースにアタッチします。次の例では、*source_database_alias*、*user_id*、*user_password* を、前のステップで作成したエイリアスと、セルフマネージド Db2 データベースのユーザー ID とパスワードに置き換えます。

   ```
   db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
       -cor -createdb -printdbcfg -o db2look.sql
   ```

4. セルフマネージド Db2 システムで EXPORT コマンドを使用してデータファイルを生成します。次の例では、*directory* を、データファイルが存在するクライアントマシン上のディレクトリに置き換えます。*file_name* と *table_name* をデータファイルの名前とテーブルの名前に置き換えます。

   ```
   db2 "export to /directory/file_name.txt of del lobs to /directory/lobs/ \
       modified by coldel\| select * from table_name"
   ```

5. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、RDS for Db2 データベースに接続します。次の例では、*rds_database_alias*、*master_username*、*master_password* をユーザー自身の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

6. IMPORT コマンドを使用して、クライアントマシン上のファイルからリモート RDS for Db2 データベースにデータをインポートします。詳細については、IBM Db2 ドキュメントの「IMPORT command」を参照してください。次の例では、*directory* と *file_name* を、データファイルが存在するクライアントマシン上のディレクトリとデータファイルの名前に置き換えます。*SCHEMA_NAME* と *TABLE_NAME* をスキーマとテーブルの名前に置き換えます。

```
db2 "IMPORT from /directory/file_name.tbl OF DEL LOBS FROM /directory/lobs/ \
    modified by coldel\| replace into SCHEMA_NAME.TABLE_NAME"
```

7. 接続を終了します。

```
db2 terminate
```

LOAD コマンドを使用してクライアントマシンから Amazon RDS for Db2 にデータをインポートする

LOAD CLIENT コマンドを使用して、クライアントマシン上のファイルから RDS for Db2 サーバーにデータをロードできます。RDS for Db2 サーバーへの SSH 接続は存在しないため、セルフマネージド Db2 サーバーまたは Db2 クライアントマシンのいずれかで LOAD CLIENT コマンドを使用できます。

> **Important**
>
> LOAD CLIENT コマンドメソッドは、小さなテーブルの移行に便利です。クライアントと RDS for Db2 間のネットワーク帯域幅が制限されている場合は、別の移行アプローチを使用することをお勧めします。詳細については、「ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。
> データファイルにラージオブジェクトファイル名への参照が含まれている場合、ラージオブジェクト (LOB) は Db2 サーバー上に存在する必要があるため、LOAD コマンドは機能しません。クライアントマシンから RDS for Db2 サーバーに LOB をロードしようとすると、SQL3025N エラーが発生します。代わりに IMPORT コマンドを使用します。

RDS for Db2 サーバーにデータをロードするには

1. IBM Db2 Data Management Console を使用してクライアントマシンにログインします。詳細については、「IBM Db2 Data Management Console を使用して Amazon RDS for Db2 DB インスタンスに接続する」を参照してください。
2. クライアントマシン上の RDS for Db2 データベースをカタログ化します。

   a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレス、セルフマネージド Db2 データベースのポート番号に置き換えます。

   ```
   db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
   ```

   b. データベースをカタログ化します。次の例では、*source_database_name* と *source_database_alias* を、セルフマネージド Db2 データベースの名前と、このデータベースに使用するエイリアスに置き換えます。

   ```
   db2 catalog database source_database_name as source_database_alias at node
    srcnode \
       authentication server_encrypt
   ```

3. ソースデータベースにアタッチします。次の例では、*source_database_alias*、*user_id*、*user_password* を、前のステップで作成したエイリアスと、セルフマネージド Db2 データベースのユーザー ID とパスワードに置き換えます。

   ```
   db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
       -cor -createdb -printdbcfg -o db2look.sql
   ```

4. セルフマネージド Db2 システムで EXPORT コマンドを使用して、データファイルを生成します。次の例では、*directory* を、データファイルが存在するクライアントマシン上のディレクトリに置き換えます。*file_name* と *TABLE_NAME* をデータファイルの名前とテーブルの名前に置き換えます。

   ```
   db2 "export to /directory/file_name.txt of del modified by coldel\| \
       select * from TPCH.TABLE_NAME"
   ```

5. RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、RDS for Db2 データベースに接続します。次の例では、*rds_database_alias#master_username*、*master_password* をユーザー自身の情報に置き換えます。

```
db2 connect to rds_database_alias user master_username using master_password
```

6. LOAD コマンドを使用して、クライアントマシン上のファイルからリモート RDS for Db2 データベースにデータをロードします。詳細については、IBM Db2 ドキュメントの「LOAD command」を参照してください。次の例では、*directory* を、データファイルが存在するクライアントマシン上のディレクトリに置き換えます。*file_name* と *TABLE_NAME* をデータファイルの名前とテーブルの名前に置き換えます。

```
db2 "LOAD CLIENT from /directory/file_name.txt \
    modified by coldel\| replace into TPCH.TABLE_NAME \
    nonrecoverable without prompting"
```

7. 接続を終了します。

```
db2 terminate
```

INSERT コマンドを使用して Db2 から Amazon RDS for Db2 にデータをインポートする

INSERT コマンドを使用して、セルフマネージド Db2 サーバーから Amazon RDS for Db2 データベースにデータを挿入できます。この移行アプローチでは、リモート RDS for Db2 DB インスタンスのニックネームを使用します。セルフマネージド Db2 データベース (ソース) は、RDS for Db2 データベース (ターゲット) に接続できる必要があります。

> **Important**
>
> INSERT コマンドメソッドは、小さなテーブルの移行に便利です。セルフマネージド Db2 データベースと RDS for Db2 データベース間のネットワーク帯域幅が制限されている場合は、別の移行アプローチを使用することをお勧めします。詳細については、「ネイティブ Db2 ツールを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。

セルフマネージド Db2 データベースから RDS for Db2 データベースにデータをコピーするには

1. セルフマネージド Db2 インスタンスで RDS for Db2 DB インスタンスをカタログ化します。

a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレス、セルフマネージド Db2 データベースのポート番号に置き換えます。

```
db2 catalog tcpip node remnode REMOTE dns_ip_address SERVER port
```

b. データベースをカタログ化します。次の例では、*rds_database_name* を RDS for Db2 DB インスタンスのデータベースの名前に置き換えます。

```
db2 catalog database rds_database_name as remdb at node remnode \
    authentication server_encrypt
```

2. セルフマネージド Db2 インスタンスでフェデレーションを有効にします。次の例では、*source_database_name* をセルフマネージド Db2 インスタンスのデータベースの名前に置き換えます。

```
db2 update dbm cfg using FEDERATED YES source_database_name
```

3. RDS for Db2 DB インスタンスでテーブルを作成します。

a. ノードをカタログ化します。次の例では、*dns_ip_address* と *port* を、DNS 名または IP アドレス、セルフマネージド Db2 データベースのポート番号に置き換えます。

```
db2 catalog tcpip node srcnode REMOTE dns_ip_address server port
```

b. データベースをカタログ化します。次の例では、*source_database_name* と *source_database_alias* を、セルフマネージド Db2 データベースの名前と、このデータベースに使用するエイリアスに置き換えます。

```
db2 catalog database source_database_name as source_database_alias at node
 srcnode \
    authentication server_encrypt
```

4. ソースデータベースにアタッチします。次の例では、*source_database_alias*、*user_id*、*user_password* を、前のステップで作成したエイリアスと、セルフマネージド Db2 データベースのユーザー ID とパスワードに置き換えます。

```
db2look -d source_database_alias -i user_id -w user_password -e -l -a -f -wlm \
    -cor -createdb -printdbcfg -o db2look.sql
```

5. フェデレーションを設定し、セルフマネージド Db2 インスタンスの RDS for Db2 データベーステーブルのニックネームを作成します。

   a. ローカルデータベースに接続します。次の例では、*source_database_name* をセルフマネージド Db2 インスタンスのデータベースの名前に置き換えます。

   ```
   db2 connect to source_database_name
   ```

   b. Db2 データソースにアクセスするためのラッパーを作成します。

   ```
   db2 create wrapper drda
   ```

   c. フェデレーションデータベースでデータソースを定義します。次の例では、*admin* と *admin_password* をセルフマネージド Db2 インスタンスの認証情報に置き換えます。*rds_database_name* を RDS for Db2 DB インスタンスのデータベースの名前に置き換えます。

   ```
   db2 "create server rdsdb2 type DB2/LUW version '11.5.9.0' \
       wrapper drda authorization "admin" password "admin_password" \
       options( dbname 'rds_database_name', node 'remnode')"
   ```

   d. 2 つのデータベースのユーザーをマッピングします。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンスの認証情報に置き換えます。

   ```
   db2 "create user mapping for user server rdsdb2 \
       options (REMOTE_AUTHID 'master_username', REMOTE_PASSWORD
    'master_password')"
   ```

   e. RDS for Db2 サーバーへの接続を確認します。

   ```
   db2 set passthru rdsdb2
   ```

   f. リモート RDS for Db2 データベースのテーブルのニックネームを作成します。次の例では、*NICKNAME* と *TABLE_NAME* をテーブルのニックネームとテーブルの名前に置き換えます。

   ```
   db2 create nickname REMOTE.NICKNAME for RDSDB2.TABLE_NAME.NICKNAME
   ```

6. リモート RDS for Db2 データベースのテーブルにデータを挿入します。セルフマネージド Db2 インスタンスのローカルテーブルの `select` ステートメントでニックネームを使用します。次

の例では、*NICKNAME* と *TABLE_NAME* をテーブルのニックネームとテーブルの名前に置き換えます。

```
db2 "INSERT into REMOTE.NICKNAME select * from RDS2DB2.TABLE_NAME.NICKNAME"
```

INGEST ユーティリティを使用した Db2 から Amazon RDS for Db2 へのデータのインポート

INGEST ユーティリティを使用して、クライアントマシン上のファイルとパイプからターゲットの Amazon RDS for Db2 DB インスタンスにデータを継続的にストリーミングできます。INGEST ユーティリティは、INSERT および MERGE オペレーションをサポートします。詳細については、IBM Db2 ドキュメントの「Ingest utility」を参照してください。

INGEST ユーティリティはニックネームをサポートしているため、このユーティリティを使用して、セルフマネージド Db2 データベースから RDS for Db2 データベースにデータを転送できます。このアプローチは、2 つのデータベース間にネットワーク接続が存在する限り機能します。

> **Important**
>
> INGEST ユーティリティはラージオブジェクト (LOB) はサポートしていません。代わりに IMPORT コマンドを使用します。

INGEST ユーティリティの RESTARTABLE 機能を使用するには、RDS for Db2 データベースで次のコマンドを実行します。

```
db2 "call sysproc.sysinstallobjects('INGEST','C',NULL,NULL)"
```

Amazon RDS for Db2 フェデレーション

Amazon RDS for Db2 データベースをフェデレーテッドデータベースとして使用できます。RDS for Db2 のフェデレーションを設定すると、RDS for Db2 データベースから複数のデータベースのデータにアクセスしてクエリを実行できます。フェデレーションにより、RDS for Db2 データベースにデータを移行したり、データを 1 つのデータベースに統合したりする必要がなくなります。

RDS for Db2 データベースをフェデレーティッドデータベースとして使用することで、データを異なるデータベースに保持しながら、すべての RDS for Db2 機能に引き続きアクセスし、さまざまな AWS のサービス を活用できます。同じタイプの異なるデータベースを接続する同種フェデレーションと、異なるタイプの異なるデータベースを接続する異種フェデレーションの両方を設定できます。

まず、RDS for Db2 の Db2 データベースをリモートデータベースに接続します。その後、接続されているすべてのデータベースに対してクエリを実行できます。例えば、RDS for Db2 データベースのテーブルとリモート Db2 on z/OS データベースのテーブルを結合する SQL JOIN ステートメントを実行できます。

トピック

- 同種フェデレーション
- 異種フェデレーション

同種フェデレーション

RDS for Db2 データベースと次の Db2 製品ファミリーの間で同種フェデレーションを設定できます。

- Linux、UNIX、Windows (LUW) 用の Db2
- Db2 iSeries
- Db2 for z/OS

RDS for Db2 同種フェデレーションは、次のアクションをサポートしていません。

- RDS for Db2 ホストデータベースにノードディレクトリとリモートデータベースをセットアップするための `CATALOG` コマンドの実行
- z/OS で Db2 にフェデレーションするときにワークロードバランシング (WLB) を設定する
- IBM データサーバードライバー設定ファイルの設定 (`db2dsdriver.cfg`)

RDS for Db2 同種フェデレーションには、次の要件があります。

- UNFENCED モードで DRDA ラッパーを作成する必要があります。そうしないと、フェデレーションは RDS for Db2 では機能しません。
- RDS for Db2 ホストデータベースからリモートホストデータベースへの送受信トラフィックを許可する必要があります。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

トピック

- ステップ 1: DRDA ラッパーとフェデレーティッドサーバーを作成する
- ステップ 2: ユーザーマッピングを作成する
- ステップ 3: 接続を確認する

ステップ 1: DRDA ラッパーとフェデレーティッドサーバーを作成する

同種フェデレーションの場合は、DRDA ラッパーとフェデレーテッドサーバーを作成します。リモートホストへの接続では、HOST、PORT、および DBNAME を使用します。

リモート Db2 データベースのタイプに基づいて、次のいずれかの方法を選択します。

- Db2 for Linux、UNIX、および Windows (LUX) データベース – 次の SQL コマンドを実行します。次の例では、*server_name* をフェデレーションに使用するサーバーの名前に置き換えます。*db2_version* をリモート Db2 データベースのバージョンに置き換えます。*username* と *password* を接続先のリモート Db2 データベースの認証情報に置き換えます。*db_name*、*dns_name*、および *port* を、接続先のリモート Db2 データベースの適切な値に置き換えます。

```
create wrapper drda options(DB2_FENCED 'N');
create server server_name type DB2/LUW wrapper drda version 'db2_version'
 authorization "master_username" password "master_password" options (add DBNAME
 'db_name',add HOST 'dns_name',add PORT 'port');
```

例

```
create wrapper drda options(DB2_FENCED 'N');
```

```
create server SERVER1 type DB2/LUW wrapper drda version '11.5' authorization
 "sysuser" password "******" options (add DBNAME 'TESTDB2',add HOST
 'ip-123-45-67-899.us-west-1.compute.internal',add PORT '25010');
```

- Db2 iSeries – 次の SQL コマンドを実行します。次の例では、*rapper_name* と *library_name* を DRDA ラッパーの名前と ラッパーライブラリファイル に置き換えます。*server_name* をフェデレーションに使用するサーバーの名前に置き換えます。*db2_version* をリモート Db2 データベースのバージョンに置き換えます。*username* と *password* を接続先のリモート Db2 データベースの認証情報に置き換えます。*dns_name*、*port*、および *db_name* を、接続先のリモート Db2 データベースの適切な値に置き換えます。

```
create wrapper wrapper_name library 'library name' options(DB2_FENCED 'N');
create server server_name type db2/mvs version db2_version wrapper wrapper_name
 authorization "sername" password "password" options (HOST 'dns_name', PORT 'port',
 DBNAME 'db_name');
```

例

```
create wrapper WRAPPER1 library 'libdb2drda.so' options(DB2_FENCED 'N');
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization
 "sysuser" password "******" options (HOST 'test1.123.com', PORT '446', DBNAME
 'STLEC1');
```

- Db2 for z/OS – 次の SQL コマンドを実行します。次の例では、*rapper_name* と *library_name* を DRDA ラッパーの名前と ラッパーライブラリファイル に置き換えます。*server_name* をフェデレーションに使用するサーバーの名前に置き換えます。*db2_version* をリモート Db2 データベースのバージョンに置き換えます。*username* と *password* を接続先のリモート Db2 データベースの認証情報に置き換えます。*dns_name*、*port*、および *db_name* を、接続先のリモート Db2 データベースの適切な値に置き換えます。

```
create wrapper wrapper_name library 'library_name' options(DB2_FENCED 'N');
create server server_name type db2/mvs version db2_version wrapper wrapper_name
 authorization "username" password "password" options (HOST 'dns_name', PORT 'port',
 DBNAME 'db_name');
```

例

```
create wrapper WRAPPER1 library 'libdb2drda.so' OPTIONS(DB2_FENCED 'N');
```

```
create server SERVER1 type db2/mvs version 11 wrapper WRAPPER1 authorization
 "sysuser" password "******" options (HOST 'test1.123.com', PORT '446', DBNAME
 'STLEC1');
```

ステップ 2: ユーザーマッピングを作成する

次の SQL コマンドを実行することによってユーザーマッピングを作成し、フェデレーテッドサーバーをデータソースサーバーに関連付けます。次の例では、*server_name* を、操作を実行するリモートサーバーの名前に置き換えます。これは、ステップ 1 で作成したサーバーです。*username* と *password* をこのリモートサーバーの認証情報に置き換えます。

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',
 REMOTE_PASSWORD 'password');
```

詳細については、IBM Db2 ドキュメントの「ユーザーマッピング」を参照してください。

ステップ 3: 接続を確認する

接続を確認することによって、フェデレーションの設定が成功したことを確認します。セッションを開いて、SET PASSTHRU コマンドを使用してネイティブ SQL コマンドをリモートデータソースに送信してから、リモートデータサーバーにテーブルを作成します。

1. セッションを開いたり閉じたりして、SQL をデータソースに送信します。次の例では、*server_name* をステップ 1 でフェデレーションのために作成したサーバーの名前に置き換えます。

   ```
   set passthru server_name;
   ```

2. 新しいテーブルを作成します。次の例では、*column_name*、*data_type*、*value* をテーブルの適切な項目に置き換えます。

   ```
   create table table_name
    ( column_name data_type(value), column_name data_type(value);
   ```

   詳細については、IBM Db2 ドキュメントの「CREATE TABLE statement」を参照してください。

3. インデックスを作成し、行の値をテーブルに挿入して、接続をリセットします。接続をリセットすると、接続は切断されますが、バックエンドプロセスは保持されます。次の例で

は、*index_name*、*table_name*、*column_name*、および *columnx_value* を情報に置き換えます。

```
create index index_name on table_name(column_name);
insert into table_name values(column1_value,column2_value,column3_value);
insert into table_name values(column1_value,column2_value,column3_value);
set passthru reset;

connect reset;
```

4. リモート Db2 データベースに接続し、リモートサーバーのニックネームを作成し、操作を実行します。リモート Db2 データベースのデータへのアクセスが完了したら、接続をリセットして終了します。次の例では、*database_name* をリモート Db2 データベースの名前に置き換えます。*nickname* を名前に置き換えます。*server_name* と *table_name* を、操作を実行するサーバーのリモートサーバーとテーブルの名前に置き換えます。*username* をリモートサーバーの情報に置き換えます。*sql_command* を、リモートサーバーで実行する操作に置き換えます。

```
connect to database_name;
create nickname nickname for server_name."username"."table_name";
select sql_command from nickname;
connect reset;
terminate;
```

例

次の例では、フェデレーテッドサーバー testdb10 での操作を許可するパススルーセッションを作成します。

次に、データ型が異なる 3 つの列を持つテーブル t1 を作成します。

次に、この例では、テーブル t1 の 3 つの列にインデックス i1_t1 を作成します。その後、これら 3 つの列の値を持つ 2 つの行を挿入し、切断します。

最後に、この例はリモート Db2 データベース testdb2 に接続し、フェデレーテッドサーバー testdb10 にテーブル　t1 のニックネームを作成します。これにより、そのデータソースのユーザー名 TESTUSER でニックネームが作成されます。SQL コマンドは、テーブル t1 からすべてのデータを出力します。この例では、セッションを切断して終了します。

```
set passthru testdbl0;
```

```
create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);

create index i1_t1 on t1(c3);
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdbl0."TESTUSER"."T1";
select * from remote_t1;
connect reset;
terminate;
```

異種フェデレーション

RDS for Db2 データベースと Oracle や Microsoft SQL Server などの他のデータソースとの間で異種フェデレーションを設定できます。Db2 LUW がサポートするデータソースの完全なリストについては、IBM サポートサイトの「Db2 LUW V11.5 にバンドルされたフェデレーションのデータソースサポートマトリックス」を参照してください。

RDS for Db2 異種フェデレーションは、以下の項目をサポートしていません。

- 他のデータソースのネイティブラッパー
- 他のデータソースの JDBC ラッパー
- これらのデータソースは RDS for Db2 にクライアントソフトウェアをインストールする必要があるため、Sybase、Informix、および Teradata データソースへのフェデレーション

RDS for Db2 異種フェデレーションには、次の要件があります。

- RDS for Db2 は ODBC ラッパーメソッドのみをサポートしています。
- ラッパーの明示的な定義を作成する場合は、オプション DB2_FENCED を 'N' に設定する必要があります。ODBC の有効なラッパーオプションのリストについては、IBM Db2 ドキュメントの「ODBC オプション」を参照してください。
- RDS for Db2 ホストデータベースからリモートホストデータベースへの送受信トラフィックを許可する必要があります。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

Oracle へのフェデレーションの詳細については、IBM サポートサイトの「Db2 フェデレーションと ODBC ドライバーを使用して Oracle にクエリする方法」を参照してください。

フェデレーションをサポートするデータソースの詳細については、IBM サポートサイトの「Db2 LUW V11.5 にバンドルされたフェデレーションのデータソースサポートマトリックス」を参照してください。

トピック

- ステップ 1: ODBC ラッパーを作成する
- ステップ 2: フェデレーテッドサーバーを作成する
- ステップ 3: ユーザーマッピングを作成する
- ステップ 4: 接続を確認する

ステップ 1: ODBC ラッパーを作成する

次のコマンドを実行してラッパーを作成します。

```
db2 "create wrapper odbc options( module '/home/rdsdb/sqllib/federation/odbc/lib/
libodbc.so')"
```

ステップ 2: フェデレーテッドサーバーを作成する

次のコマンドを実行して、ふぇでれ–テッドサーバーを作成します。次の例では、*server_name* をフェデレーションに使用するサーバーの名前に置き換えます。*wrapper_type* を適切なラッパーに置き換えます。*db_version* をリモートデータベースのバージョンに置き換えます。*dns_name*、*port*、および *sservice_name* を、接続先のリモートデータベースの適切な値に置き換えます。

```
db2 "create server server_name type wrapper_type version db_version options (HOST
 'dns_name', PORT 'port', SERVICE_NAME 'service_name')“
```

ラッパータイプの詳細については、IBM サポートサイトの「Db2 LUW V11.5 にバンドルされたフェデレーションのデータソースサポートマトリックス」を参照してください。

例

次の例では、リモート Oracle データベース用のフェデレーテッドサーバーを作成します。

```
db2 "create server server1 type oracle_odbc version 12.1 options (HOST
 'test1.amazon.com', PORT '1521', SERVICE_NAME 'pdborcl.amazon.com')“
```

ステップ 3: ユーザーマッピングを作成する

次の SQL コマンドを実行することによってユーザーマッピングを作成し、フェデレーテッドサーバーをデータソースサーバーに関連付けます。次の例では、*server_name* を、操作を実行するリモートサーバーの名前に置き換えます。これは、ステップ 2 で作成したサーバーです。*username* と *password* をこのリモートサーバーの認証情報に置き換えます。

```
create user mapping for user server server_name options (REMOTE_AUTHID 'username',
 REMOTE_PASSWORD 'password');
```

詳細については、IBM Db2 ドキュメントの「ユーザーマッピング」を参照してください。

ステップ 4: 接続を確認する

接続を確認することによって、フェデレーションの設定が成功したことを確認します。セッションを開いて、SET PASSTHRU コマンドを使用してネイティブ SQL コマンドをリモートデータソースに送信してから、リモートデータサーバーにテーブルを作成します。

1. セッションを開いたり閉じたりして、SQL をデータソースに送信します。次の例では、*server_name* をステップ 2 でフェデレーション用に作成したサーバーの名前に置き換えます。

   ```
   set passthru server_name;
   ```

2. 新しいテーブルを作成します。次の例では、*column_name*、*data_type*、*value* をテーブルの適切な項目に置き換えます。

   ```
   create table table_name
    ( column_name data_type(value), column_name data_type(value);
   ```

   詳細については、IBM Db2 ドキュメントの「CREATE TABLE statement」を参照してください。

3. インデックスを作成し、行の値をテーブルに挿入して、接続をリセットします。接続をリセットすると、接続は切断されますが、バックエンドプロセスは保持されます。次の例では、*index_name*、*table_name*、*column_name*、および *columnx_value* を情報に置き換えます。

```
create index index_name on table_name(column_name);
insert into table_name values(column1_value,column2_value,column3_value);
insert into table_name values(column1_value,column2_value,column3_value);
set passthru reset;

connect reset;
```

4. リモート Db2 データベースに接続し、リモートサーバーのニックネームを作成し、操作を実行します。リモート Db2 データベースのデータへのアクセスが完了したら、接続をリセットして終了します。次の例では、*database_name* をリモート Db2 データベースの名前に置き換えます。*nickname* を名前に置き換えます。*server_name* と *table_name* を、操作を実行するサーバーのリモートサーバーとテーブルの名前に置き換えます。*username* をリモートサーバーの情報に置き換えます。*sql_command* を、リモートサーバーで実行する操作に置き換えます。

```
connect to database_name;
create nickname nickname for server_name."username"."table_name";
select sql_command from nickname;
connect reset;
terminate;
```

例

次の例では、フェデレーテッドサーバー testdb10 での操作を許可するパススルーセッションを作成します。

次に、データ型が異なる 3 つの列を持つテーブル t1 を作成します。

次に、この例では、テーブル t1 の 3 つの列にインデックス i1_t1 を作成します。その後、これら 3 つの列の値を持つ 2 つの行を挿入し、切断します。

最後に、この例はリモート Db2 データベース testdb2 に接続し、フェデレーテッドサーバー testdb10 にテーブル　t1 のニックネームを作成します。これにより、そのデータソースのユーザー名 TESTUSER でニックネームが作成されます。SQL コマンドは、テーブル t1 からすべてのデータを出力します。この例では、セッションを切断して終了します。

```
set passthru testdbl0;

create table t1 ( c1 decimal(13,0), c2 char(200), c3 int);
```

```
create index i1_t1 on t1(c3);
insert into t1 values(1,'Test',1);
insert into t1 values(2,'Test 2',2);
connect reset;

connect to testdb2;
create nickname remote_t1 for testdbl0."TESTUSER"."T1";
select * from remote_t1;
connect reset;
terminate;
```

Amazon RDS for Db2 のレプリカの使用

RDS for Db2 は、リードスケーリングとディザスタリカバリ機能を提供するレプリカデータベースの作成をサポートしています。レプリカは、読み取りワークロードをオフロードするための読み取り専用レプリカと、クロスリージョンディザスタリカバリ用のスタンバイレプリカの 2 つのモードで作成できます。RDS for Db2 は、IBM Db2 高可用性ディザスタリカバリ (HADR) をレプリケーションに使用します。詳細については、IBM Db2 ドキュメントの「High availability disaster recovery (HADR)」を参照してください。

Db2 レプリカデータベースは、プライマリデータベースの物理コピーです。読み取り専用モードの Db2 レプリカは、リードレプリカと呼ばれます。スタンバイモードの Db2 レプリカは、スタンバイレプリカと呼ばれます。Db2 では、レプリカに対する書き込みは許可されませんが、レプリカを昇格させて書き込み可能にすることができます。昇格させたレプリカには、昇格をリクエストした時点までにレプリケートされたデータが含まれます。詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

RDS for Db2 レプリカの機能と動作の概要については、「DB エンジンのリードレプリカ間の違い」を参照してください。

読み取り専用レプリカとスタンバイレプリカ

Db2 レプリカを作成または変更する場合、次のモードのいずれかにすることができます。

読み取り専用モード。

これがデフォルトです。HADR は、ソースデータベースからすべてのリードレプリカデータベースに変更を送信し、適用します。読み取り専用レプリカの場合、Db2 環境変数 `DB2_HADR_ROS` は `ON` に設定されます。レプリカデータベースの読み取りクエリの分離レベルは `Uncommitted Read` です。詳細については、IBM Db2 ドキュメントの「Isolation level on the active standby database」を参照してください。

すべての DB エンジンに適用されるリードレプリカの一般的な情報については、「DB インスタンスのリードレプリカの操作」を参照してください。Db2 HADR の詳細については、IBM Db2 ドキュメントの「High availability disaster recovery (HADR)」を参照してください。

スタンバイ

スタンバイレプリカでは、Db2 環境変数 `DB2_HADR_ROS` を `OFF` に設定し、レプリカデータベースがユーザー接続を受け付けないようにします。スタンバイレプリカの主な用途は、クロスリージョンのディザスタリカバリです。

スタンバイレプリカは、読み取り専用のワークロードを処理できません。スタンバイレプリカにはアーカイブログがありません。

1 つのソース DB インスタンスから最大 3 つのリードレプリカを作成できます。同じソース DB インスタンスに対して、読み取り専用 DB レプリカとスタンバイ DB レプリカを組み合わせて作成できます。レプリカを作成したら、レプリカモードを変更できます。詳細については、「RDS for Db2 レプリカモードの変更」を参照してください。

レプリカを作成する前に、すべての要件を満たしていることを確認してください。詳細については、「RDS for Db2 レプリカの要件と考慮事項」を参照してください。

データベースのアクティブ化

Db2 HADR はデータベースレベルで設定します。レプリカを作成すると、RDS が完全に管理する `rdsadmin` を含むすべての Db2 データベースに HADR が設定されます。Db2 レプリカを作成する前に、すべてのデータベースを明示的にアクティブ化する必要があります。そうしないと、レプリカの作成は失敗し、Amazon RDS でイベントが発生します。DB インスタンスに 1 つ以上のレプリカがある場合、`rdsadmin.activate_database` または `rdsadmin.deactivate_database` ストアドプロシージャを使用して DB インスタンスのデータベースをアクティブ化または非アクティブ化することはできません。詳細については、「RDS for Db2 のデータベースのストアドプロシージャ」を参照してください。

HADR 設定

データベースに接続して `db2 get db cfg` を実行すると、データベースのすべての HADR 設定を表示できます。

アーカイブログの保持

Amazon RDS は、以下の条件が満たされると、プライマリ DB インスタンスからログを消去します。

- ログは 2 時間以上経過している。
- アーカイブログに設定した保持時間が経過している。
- アーカイブログは、すべてのレプリカ DB インスタンスに正常にレプリケートされている。この条件は、同じ AWS リージョン内の DB インスタンスと、クロスリージョンの DB インスタンスの両方に適用されます。

アーカイブログの保持時間の設定については、「rdsadmin.set_archive_log_retention」を参照してください。

Amazon RDS は、各データベースを個別にチェックしてクリーンアップします。データベースが HADR 接続を失った場合や、接続に関する情報が利用できない場合、Amazon RDS はデータベースをスキップし、アーカイブログを消去しません。

Db2 レプリケーション中の停止

リードレプリカを作成すると、Amazon RDS はソース DB インスタンスの DB スナップショットを取得し、レプリケーションを開始します。DB スナップショットオペレーションが始まると、ソース DB インスタンスでごく短時間の I/O 停止が発生します。通常、I/O 停止は約 1 秒続きます。ただし、ソース DB インスタンスがマルチ AZ 配置の場合、ソース DB インスタンスでは I/O 停止が発生しません。マルチ AZ 配置では、スナップショットがセカンダリ DB インスタンスから取得されるためです。

DB スナップショットが Db2 レプリカになります。Amazon RDS は、サービスを中断することなく、ソースデータベースとレプリカに必要なパラメータとアクセス許可を設定します。同様に、レプリカを削除する場合も、停止は発生しません。

RDS for Db2 レプリカの要件と考慮事項

Db2 レプリカの要件は、ライセンスとバージョニング、バックアップと復元に関する考慮事項、レプリケーションの動作、一般的な運用上の考慮事項のいくつかのカテゴリに分類されます。Db2 レプリカを作成する前に、以下の要件と考慮事項をよく確認してください。

RDS for Db2 レプリカのバージョンとライセンスの要件

RDS for Db2 レプリカを作成する前に、バージョンとライセンスモデルに関する次の情報を確認してください。

- サポートされるバージョン – すべての Db2 11.5 および 12.1 バージョンは、レプリカ DB インスタンスをサポートしています。

  ソース DB インスタンスとレプリカ DB インスタンスは、同じメジャーバージョンを使用する必要があります。Db2 レプリカはマイナーバージョンアップグレードをサポートしていますが、メジャーバージョンアップグレードはサポートしていません。DB インスタンスのアップグレードに関する情報については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

> **Note**
>
> ソース DB インスタンスをアップグレードすると、すべてのレプリカが自動的にアップグレードされ、バージョンの互換性が維持されます。

- 有効なライセンスモデルとレプリカモード – すべての Db2 エディション、Advanced Edition (AE)、Community Edition (CE)、および Standard Edition (SE) は、読み取り専用モードまたはスタンバイモードでのレプリカをサポートしています。Db2 AE と Db2 SE は、Bring Your Own License (BYOL) モデルと AWS Marketplace 経由の Db2 ライセンスモデルのいずれでも利用可能です。CE は BYOL でのみ使用できます。CE 用の IBM カスタマー ID と IBM サイト ID を取得するには、IBM ウェブサイトを参照してください。
- カスタムパラメータグループ – レプリカのカスタムパラメータグループを指定する必要があります。

  BYOL モデルを使用するレプリカの場合、このカスタムパラメータグループには IBM Site ID と IBM Customer ID を含める必要があります。詳細については、「Db2 の Bring-Your-Own-License (BYOL) の IBM ID」を参照してください。レプリカのカスタムパラメータグループは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して指定できます。
- vCPU の数は、レプリカモードとライセンスモデルによって異なります。
  - スタンバイレプリカは、DB インスタンスのサイズに関係なく、常に 2 つの vCPU を使用します。
    - BYOL モデル – AWS License Manager の設定は RDS for Db2 DB インスタンスが 2 つの vCPU を使用することを示します。
    - AWS Marketplace モデル経由の Db2 ライセンス – 請求には 2 つの vCPU のライセンスコストが反映されます。
  - 読み取り専用レプリカは、DB インスタンスサイズと同じ vCPU 数を使用します。
    - BYOL モデル – AWS License Manager 設定は、RDS for Db2 DB インスタンスが DB インスタンスサイズに一致する同じ数の vCPU を使用することを示しています。
    - AWS Marketplace モデル経由の Db2 ライセンス – 請求書には、DB インスタンスサイズに一致する同じ数の vCPU ライセンスコストが反映されます。

RDS for Db2 レプリカのバックアップと復元に関する考慮事項

レプリカバックアップの動作は、プライマリデータベースバックアップとは異なります。次のバックアップと復元の要件を検討してください。

- RDS for Db2 レプリカのスナップショットを作成したり、自動バックアップを有効にしたりするには、必ずバックアップ保持期間を手動で設定してください。自動バックアップは、デフォルトでは有効になっていません。
- レプリカのバックアップを復元するときは、バックアップが実行された時刻ではなく、データベース時刻に復元することになります。データベース時刻は、バックアップのデータにトランザクションが最後に適用された時刻を指します。レプリカはプライマリよりも数分または数時間遅れることがあるため、この違いは重要です。複数のデータベースがある場合、RDS for Db2 は最も早いデータベース時間を使用します。

  違いを確認するには、AWS CLI の describe-db-snapshots コマンドを実行するか、RDS API の DescribeDBSnapshots オペレーションを呼び出します。`SnapshotDatabaseTime` 値と `OriginalSnapshotCreateTime` 値を比較します。`SnapshotDatabaseTime` 値は、レプリカバックアップのデータベース時刻です。`OriginalSnapshotCreateTime` 値は、プライマリデータベースに最後に適用されたトランザクションです。

バックアップおよびバックアップを復元する方法については、「RDS for Db2 レプリカのバックアップの使用」を参照してください。

RDS for Db2 レプリカのレプリケーションに関する考慮事項

Db2 レプリカは、特定の制限と動作を持つ HADR テクノロジーを使用します。次のレプリケーションに関する考慮事項を確認してください。

- レプリケーションは、RDS for Db2 DB インスタンスのすべてのデータベースに Db2 HADR を使用します。
- レプリケーションは `LOAD` コマンドをサポートしていません。ソース DB インスタンスから `LOAD` コマンドを実行すると、不整合なデータが返されます。
- RDS for Db2 は、以下の項目をレプリケートしません。
  - ストレージアクセス。ストレージアクセスに依存する、外部テーブルなどのデータに注意してください。
  - ログに記録されない非インライン LOB。
  - 外部ストアドプロシージャのバイナリ (C または Java)。
- スタンバイレプリカの場合、RDS for Db2 は以下の項目をレプリケートします。
  - マスターユーザーを除くローカルユーザー
  - データベース設定パラメータ
- 読み取り専用レプリカの場合、RDS for Db2 は以下の項目をレプリケートします。

- マスターユーザーを除くローカルユーザー
- SID グループマッピング

RDS for Db2 レプリカに関するその他の考慮事項

Db2 レプリカには、追加の運用上の考慮事項が適用されます。以下の項目を確認します。

- RDS for Db2 は、データベース設定をレプリカにレプリケートします。RDS for Db2 は、レプリカの昇格時に、各データベースを非アクティブ化およびアクティブ化します。
- RDS for Db2 は、(マスターユーザーではなく) ローカルユーザーと SID グループマッピングをレプリカにレプリケートします。レプリカのマスターユーザーは変更できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- すべてのデータベースはアクティブ状態である必要があります。データベースのアクティブ化については、「RDS for Db2 のデータベースのストアドプロシージャ」を参照してください。
- レプリカを作成する前に、データベースを作成、削除、復元、またはロールフォワードするすべてのストアドプロシージャを完了する必要があります。これらのストアドプロシージャについては、「RDS for Db2 のデータベースのストアドプロシージャ」を参照してください。
- レプリカが作成されると、Amazon RDS はソース DB インスタンスのすべてのデータベースについて、データベースレベルのパラメータ `blocknonlogged` を `YES` に設定します。ソースレプリカが再びスタンドアロンインスタンスになると、Amazon RDS は値を `NO` に戻します。詳細については、IBM Db2 ドキュメントの「blocknonlogged - Block creation of tables that allow non-logged activity configuration parameter」を参照してください。
- レプリカが作成されると、Amazon RDS はソース DB インスタンスのすべてのデータベースについて、データベースレベルのパラメータ `logindexbuild` を `YES` に設定します。ソースレプリカが再びスタンドアロンインスタンスになると、Amazon RDS は値を `NO` に戻します。詳細については、IBM Db2 ドキュメントの「logindexbuild - Log index pages created configuration parameter」を参照してください。

RDS for Db2 レプリカの作成の準備

RDS for Db2 レプリカを作成する前に、レプリケーションを成功させるために以下のタスクを完了する必要があります。これらのタスクは、一般的な問題を防ぎ、最適なパフォーマンスを確保するのに役立ちます。

タスク 1: 自動バックアップの有効化

DB インスタンスがソース DB インスタンスとして機能するには、ソース DB インスタンスで自動バックアップを有効にする必要があります。これは、すべてのレプリカ作成オペレーションの前提条件です。この手順の実行方法については、「自動バックアップの有効化」を参照してください。

Db2 レプリカに固有のバックアップについては、「RDS for Db2 レプリカのバックアップの使用」を参照してください。

タスク 2: コンピューティングとストレージのリソース計画

ソース DB インスタンスとそのレプリカのサイズが、運用負荷に合わせる上でコンピューティングとストレージの観点から適切に設定されていることを確認してください。レプリカのコンピューティング、ネットワーク、またはストレージがリソースの容量に達すると、レプリカはソースからの変更の受信または適用を停止します。レプリカのパフォーマンスとリソース使用率のモニタリングについては、「リードレプリケーションのモニタリング」を参照してください。

RDS for Db2 が、ソース DB インスタンスとそのレプリカの間におけるレプリカの高遅延を軽減するために介入することはありません。レプリカの遅延が大きい場合は、トラブルシューティングのガイダンスについて「Db2 レプリケーション遅延のモニタリング」を参照してください。

レプリカのストレージや CPU リソースは、そのソースや他のレプリカとは独立して変更することができます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

タスク 3: データベースを準備する

レプリカを作成する前に、次の点に基づいてデータベースの準備が整っていることを確認します。

- DB インスタンスにすべての必要なデータベースが含まれていることを確認してください。レプリカの作成後は、DB インスタンス内のデータベースを作成、削除、またはネイティブ復元することはできません。rdsadmin.create_database、rdsadmin.drop_database、または rdsadmin.restore_database ストアドプロシージャへのすべての呼び出しは失敗します。
- DB インスタンス内のすべてのデータベースはアクティブ状態です。データベースが非アクティブ状態の場合、レプリカの作成は失敗します。データベースのアクティブ化については、「RDS for Db2 のデータベースのストアドプロシージャ」を参照してください。

次のステップ

すべての準備タスクを完了したら、Db2 レプリカを作成する準備が整います。

- 読み取り専用レプリカを作成するには、「リードレプリカの作成」を参照してください。
- スタンバイレプリカを作成するには、「スタンバイ Db2 レプリカの作成」を参照してください。

スタンバイモードでの RDS for Db2 レプリカの作成

デフォルトでは、Db2 レプリカは読み取り専用モードで作成されます。ディザスタリカバリの目的で、スタンバイモードでレプリカを作成できます。スタンバイレプリカはユーザー接続を受け入れませんが、クロスリージョンシナリオでより高速なフェイルオーバー機能を提供します。

スタンバイレプリカを作成する前に、準備タスクが完了していることを確認してください。詳細については、「RDS for Db2 レプリカの作成の準備」を参照してください。スタンバイレプリカを作成したら、レプリカモードを変更できます。詳細については、「RDS for Db2 レプリカモードの変更」を参照してください。

スタンバイレプリカを作成するには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用できます。読み取り専用レプリカの作成については、「リードレプリカの作成」を参照してください。

コンソール

ソース RDS for Db2 DB インスタンスからスタンバイレプリカを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. スタンバイレプリカのソースとして使用する RDS for Db2 DB インスタンスを選択します。
4. アクションでレプリカの作成を選択します。
5. [レプリカモード]で、[スタンバイ]を選択します。
6. 使用する設定を選択します。[DB インスタンス識別子] に、スタンバイレプリカの名前を入力します。必要に応じて他の設定を変更します。
7. [リージョン] で、スタンバイレプリカを起動する AWS リージョンを選択します。
8. インスタンスサイズとストレージタイプを選択します。スタンバイレプリカでもソース DB インスタンスと同じ DB インスタンスクラスとストレージタイプを使用することをお勧めします。
9. [マルチ AZ 配置] で [スタンバイインスタンスを作成] を選択し、スタンバイレプリカのフェイルオーバーをサポートするために、別のアベイラビリティーゾーンにレプリカのスタンバイを作成します。

10. 使用する他の設定を選択します。
11. レプリカの作成を選択します。

[データベース] ページで、スタンバイレプリカにロールとして [レプリカ] が割り当てられます。

AWS CLI

スタンバイモードで Db2 レプリカを作成するには、AWS CLI コマンド create-db-instance-read-replica で `--replica-mode` を `mounted` に設定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier my_standby_replica \
    --source-db-instance-identifier my_db_instance \
    --replica-mode mounted
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier my_standby_replica ^
    --source-db-instance-identifier my_db_instance ^
    --replica-mode mounted
```

RDS API

スタンバイモードで Db2 レプリカを作成するには、RDS API オペレーション CreateDBInstanceReadReplica で `ReplicaMode=mounted` を指定します。

RDS for Db2 レプリカモードの変更

既存の Db2 レプリカのレプリカモードは、読み取り専用モードとスタンバイモードの間で変更できます。この柔軟性により、読み取りワークロードやディザスタリカバリのニーズの変化する要件に基づいてレプリカ設定を適応させることができます。

以下のシナリオでは、レプリカモードを変更することができます。

- 読み取り専用からスタンバイ – 読み取り容量が不要になったが、ディザスタリカバリ機能を維持したい場合

- 読み取り専用へのスタンバイ – レポートまたは分析ワークロードの読み取り容量を追加する必要がある場合

レプリカモードを変更する前に、次の条件が満たされていることを確認してください。

- レプリカは使用可能な状態です。
- レプリカでアクティブなメンテナンスオペレーションが実行されていません。
- DB インスタンスを変更するために必要なアクセス許可があります。

変更には数分かかる場合があります。オペレーション中、DB インスタンスのステータスは modifying に変わります。ステータス変更の詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。読み取り専用モードからスタンバイモードに変更すると、レプリカはすべてのアクティブな接続を切断します。

> **Important**
>
> レプリカモードを変更するとサービスが一時的に中断されるため、メンテナンスウィンドウ中に変更を計画して、アプリケーションへの影響を最小限に抑えます。

レプリカモードは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して変更できます。

コンソール

Db2 レプリカのレプリカモードを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. 変更するレプリカデータベースを選択します。
4. [Modify] (変更) を選択します。
5. [レプリカモード] では、目的のモードを選択します。
   - 読み取り専用 – 読み取りワークロード用
   - スタンバイ — ディザスタリカバリ用
6. 変更する他の設定を選択します。

7. [Continue] を選択します。
8. [変更のスケジューリング] で、[すぐに適用] を選択します。
9. [DB インスタンスの変更] を選択します。
10. 変更が完了したら、[データベース] ページでレプリカモードの変更を確認します。変更が完了すると、レプリカのステータスが [利用可能] と表示されます。

AWS CLI

Db2 レプリカを読み取り専用モードからスタンバイモードに変更するには、AWS CLI コマンド modify-db-instance で `--replica-mode` を `mounted` に設定します。Db2 レプリカをスタンバイモードから読み取り専用モードに変更するには、`--replica-mode` を `open-read-only` に設定します。

次の例では、レプリカを読み取り専用モードからスタンバイモードに変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db2_replica \
    --replica-mode mounted
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db2_replica ^
    --replica-mode mounted
```

次の例では、レプリカをスタンバイモードから読み取り専用モードに変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db2_replica \
    --replica-mode open-read-only
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db2_replica ^
    --replica-mode open-read-only
```

RDS API

Db2 レプリカを読み取り専用モードからスタンバイモードに変更するには、ModifyDBInstance で ReplicaMode=mounted に設定します。Db2 レプリカをスタンバイモードから読み取り専用モードに変更するには、ReplicaMode=open-read-only に設定します。

以下は、レプリカモードを読み取り専用からスタンバイに変更するための API コールの例です。

```
{
    "DBInstanceIdentifier": "my_db2_replica",
    "ReplicaMode": "mounted",
    "ApplyImmediately": true
}
```

以下は、レプリカモードをスタンバイモードから読み取り専用モードに変更するための API コールの例です。

```
{
    "DBInstanceIdentifier": "my_db2_replica",
    "ReplicaMode": "open-read-only",
    "ApplyImmediately": true
}
```

レプリカモードの違いについては、「Amazon RDS for Db2 のレプリカの使用」を参照してください。レプリカの問題のトラブルシューティングについては、「RDS for Db2 レプリケーションの問題のトラブルシューティング」を参照してください。

RDS for Db2 レプリカのバックアップの使用

プライマリデータベースと同様に、RDS for Db2 レプリカのバックアップを作成および復元できます。ただし、レプリカバックアップの仕組み、特に復元のタイミングとバックアップの保持設定には重要な違いがあります。

RDS for Db2 は、レプリカの自動バックアップと手動スナップショットの両方をサポートしています。RDS for Db2 はポイントインタイムリストアをサポートしていません。RDS バックアップの詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

レプリカバックアップの主な違い

レプリカバックアップは、いくつかの重要な点でプライマリデータベースバックアップとは異なります。

- 自動バックアップは、レプリカではデフォルトで有効になっていません。
- 復元オペレーションでは、バックアップの作成時間ではなくデータベース時間を使用します。
- レプリカの遅延は、復元された実際のデータに影響を与える可能性があります。レプリカ遅延のモニタリングの詳細については、「Db2 レプリケーション遅延のモニタリング」を参照してください。

RDS for Db2 レプリカの自動バックアップの有効化

プライマリデータベースとは異なり、RDS for Db2 レプリカでは、自動バックアップがデフォルトで有効になっていません。自動バックアップを有効にするには、バックアップ保持期間を手動で設定する必要があります。バックアップ保持期間を 0 以外の正の値に設定することで自動バックアップを有効にします。

コンソール

自動バックアップをすぐに有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスを選択します。
3. [Modify] (変更) を選択します。
4. [バックアップ保持期間] で、ゼロ以外の正の値 (3 日など) を選択します。
5. [続行] を選択します。
6. [すぐに適用] を選択します。
7. [DB インスタンスを変更] を選択して変更を保存し、自動バックアップを有効にします。

AWS CLI

自動バックアップを有効にするには、AWS CLI の modify-db-instance コマンドを使用します。

以下のパラメータを含めます。

- `--db-instance-identifier`
- `--backup-retention-period`
- `--apply-immediately` 、、または `--no-apply-immediately`

次の例では、バックアップ保持期間を 3 日に設定して、自動バックアップを有効にします。変更はすぐに適用されます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
    --backup-retention-period 3 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

自動バックアップを有効にするには、以下の必須パラメータを指定して RDS API の ModifyDBInstance オペレーションを使用します。

- `DBInstanceIdentifier`
- `BackupRetentionPeriod`

RDS for Db2 レプリカのバックアップの復元

RDS for Db2 レプリカのバックアップは、プライマリデータベースのバックアップを復元するのと同じ方法で復元できます。詳細については、「DB インスタンスへの復元」を参照してください。

レプリカバックアップを復元する際の最も重要な考慮事項は、特にレプリカラグが存在する場合、データベース時間とバックアップ作成時間の違いを理解することです。

レプリケーションの遅延をモニタリングし、バックアップに期待されるデータが含まれていることを確認できます。ReplicaLag メトリクスについては、「Amazon RDS の Amazon CloudWatch メトリクス」をご参照ください。

タイミングの違いについて

レプリカバックアップを復元する場合、復元する時点を決定する必要があります。データベース時刻は、バックアップのデータにトランザクションが最後に適用された時刻を指します。レプリカのバックアップを復元するときは、バックアップが完了した時刻ではなく、データベース時刻に復元することになります。レプリカはプライマリデータベースよりも数分または数時間遅れることがあるため、この違いは重要です。したがって、レプリカバックアップのデータベース時間は、スナップショットの作成時間よりもはるかに早い場合があります。

データベース時刻と作成時刻の違いを確認するには、AWS CLI の describe-db-snapshots コマンドを実行するか、RDS API の DescribeDBSnapshots オペレーションを呼び出します。SnapshotDatabaseTime 値と OriginalSnapshotCreateTime 値を比較します。SnapshotDatabaseTime 値は、レプリカバックアップのすべてのデータベースのうち、最も早いデータベース時刻です。OriginalSnapshotCreateTime 値は、プライマリデータベースに最後に適用されたトランザクションです。レプリケーション遅延は複数のデータベースで異なる場合があり、データベース時刻はこれら 2 つの時刻の間になる可能性があることに注意してください。

次の AWS CLI の例は、2 つの時刻の差を示しています。

Linux、macOS、Unix の場合:

```
aws rds describe-db-snapshots \
    --db-instance-identifier my_db2_replica \
    --db-snapshot-identifier my_replica_snapshot
```

Windows の場合:

```
aws rds describe-db-snapshots ^
    --db-instance-identifier my_db2_replica ^
    --db-snapshot-identifier my_replica_snapshot
```

このコマンドでは、次の例のような出力が生成されます。

```
{
    "DBSnapshots": [
        {
```

```
            "DBSnapshotIdentifier": "my_replica_snapshot",
            "DBInstanceIdentifier": "my_db2_replica",
            "SnapshotDatabaseTime": "2022-07-26T17:49:44Z",
            ...
            "OriginalSnapshotCreateTime": "2021-07-26T19:49:44Z"
        }
    ]
}
```

RDS for Db2 レプリケーションの問題のトラブルシューティング

このトピックでは、RDS for Db2 レプリケーションの一般的な問題について説明し、読み取り専用レプリカとスタンバイレプリカの両方のトラブルシューティングガイダンスを提供します。以下のトラブルシューティング情報を確認するだけでなく、Db2 レプリカを作成する前に、要件と考慮事項に従って準備手順を完了していることを確認してください。

レプリカ作成の失敗

レプリカの作成は、いくつかの理由で失敗することがあります。

- 非アクティブなデータベース – レプリカを作成する前に、ソース DB インスタンス上のすべてのデータベースがアクティブである必要があります。

  データベースのアクティブ化については、「RDS for Db2 のデータベースのストアドプロシージャ」を参照してください。
- 自動バックアップがない – ソース DB インスタンスで自動バックアップが有効になっている必要があります。

  バックアップの有効化については、「RDS for Db2 レプリカの自動バックアップの有効化」を参照してください。
- パラメータグループの問題 – レプリカにはカスタムパラメータグループが必要です。BYOL ライセンスの場合、パラメータグループには IBM Site ID と IBM Customer ID を含める必要があります。

  詳細については、「Db2 の Bring-Your-Own-License (BYOL) の IBM ID」を参照してください。

Db2 レプリケーション遅延のモニタリング

Amazon CloudWatch のレプリケーションラグをモニタリングするには、Amazon RDS `ReplicaLag` メトリクスを表示します。レプリケーションのラグタイムについては、「リードレプリケーション

のモニタリング」および「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。レプリカの遅延の CloudWatch アラームの設定については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

読み取り専用レプリカの場合、遅延時間が長すぎるときは、MON_GET_HADR テーブルをクエリしてレプリカ DB インスタンスのステータスをチェックします。

スタンバイレプリカの場合、遅延時間が長すぎるときは、MON_GET_HADR テーブルをクエリしてソース DB インスタンスのステータスをチェックします。レプリカ DB インスタンスはユーザー接続を受け付けないため、レプリカ DB インスタンスをクエリしないでください。

レプリケーションの遅延が高い一般的な原因には、次のようなものがあります。

- レプリカのコンピューティングリソースが不十分
- ソースとレプリカ間のネットワーク接続の問題
- ソースデータベースでの書き込みアクティビティが高い
- レプリカのストレージパフォーマンスの制限

レプリケーションの遅延が大きい場合は、レプリカリソースのスケーリングを検討してください。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Db2 レプリケーションエラー

Db2 レプリケーションは、さまざまな理由でエラー状態になることがあります。次のアクションを実行します。

- イベントと DB インスタンスの状態をモニタリングして、DB インスタンスがレプリケートされていることを確認してください。

  詳細については、「Amazon RDS イベント通知の操作」を参照してください。
- Amazon RDS コンソールで Db2 レプリカの診断ログを確認します。ログで、HADR メッセージのエラーを探します。ログシーケンス番号とプライマリシーケンス番号を比較します。

  Db2 診断ログへのアクセスと解釈については、「Amazon RDS for Db2 データベースのログファイル」を参照してください。Db2 HADR の設定とトラブルシューティングの詳細については、「Amazon RDS for Db2 のレプリカの使用」を参照してください。

レプリケーションエラーが解決しない場合は、レプリカを再作成する必要がある場合があります。

接続の問題

レプリカに接続できない場合は、レプリカモードに関する次の情報を確認してください。

- スタンバイレプリカ – 設計上、ユーザー接続は受け付けません。読み込みワークロードには読み取り専用レプリカを使用します。
- 読み取り専用レプリカ – セキュリティグループ設定、ネットワーク ACL、パラメータグループ設定を確認します。

  詳細については、「Amazon VPC ユーザーガイド」の「セキュリティグループを使用して AWS リソースへのトラフィックを制御する」、「Amazon VPC ユーザーガイド」の「ネットワークアクセスコントロールリストを使用してサブネットのトラフィックを制御する」、および「Amazon RDS のパラメータグループ」を参照してください。

パフォーマンスの問題

レプリカのパフォーマンスが低い場合は、次の提案を確認してください。

- レプリカに適切なコンピューティングリソースとストレージリソースがあることを確認します。
- Amazon CloudWatch で `ReplicaLag` メトリクスをモニタリングします。
- レプリカ DB インスタンスクラスをスケールアップすることを検討してください。

リソースまたはインスタンスクラスの変更方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

レプリケーション遅延のモニタリングの詳細については、「レプリケーションラグのモニタリング」と「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。レプリカの遅延の CloudWatch アラームの設定については、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。

RDS for Db2 DB インスタンスのオプション

以下は、Db2 DB エンジンを実行する Amazon RDS インスタンスで使用できるオプションまたは追加機能を示しています。これらのオプションを有効にするには、カスタムオプショングループにオプションを追加して、そのオプショングループを DB インスタンスに関連付けます。オプショングループの操作方法の詳細については、「オプショングループを使用する」を参照してください。

Amazon RDS では、以下の Db2 用オプションがサポートされています。

オプション	オプション ID
Db2 監査ログ記録	DB2_AUDIT

Db2 監査ログ記録

Amazon RDS では、Db2 監査プラグインを使用して、データベースへのユーザーのログオンやデータベースに対して実行されたクエリなどのデータベースアクティビティを記録します。RDS は、完了した監査ログを Amazon S3 バケットにアップロードします。この際、指定された AWS Identity and Access Management (IAM) ロールを使用します。

トピック

- Db2 監査ログのセットアップ
- Db2 監査ログ記録の管理
- 監査ログの表示
- 監査ログ記録のトラブルシューティング

Db2 監査ログのセットアップ

Amazon RDS for Db2 データベースの監査ログを有効にするには、RDS for Db2 DB インスタンスで `DB2_AUDIT` オプションを有効にします。次に、監査ポリシーを設定して、特定のデータベースの機能を有効にします。RDS for Db2 DB インスタンスでオプションを有効にするには、`DB2_AUDIT` オプションのオプション設定を行います。そのためには、Amazon S3 バケットの Amazon リソースネーム (ARN) と、バケットへのアクセス許可を持つ IAM ロールを指定します。

RDS for Db2 データベースの Db2 監査ログを設定するには、次のステップを実行します。

トピック

- ステップ 1: Amazon S3 バケットを作成する
- ステップ 2: IAM ポリシーを作成する
- ステップ 3: IAM ロールを作成して IAM ポリシーをアタッチする
- ステップ 4: Db2 監査ログ記録のオプショングループを設定する
- ステップ 5: 監査ポリシーを設定する
- ステップ 6: 監査設定を確認する

ステップ 1: Amazon S3 バケットを作成する

まだ作成していない場合は、Amazon RDS が RDS for Db2 データベースの監査ログファイルをアップロードできる Amazon S3 バケットを作成します。監査ファイルのターゲットとして使用する S3 バケットには、以下の制限が適用されます。

- RDS for Db2 DB インスタンスと同じ AWS リージョン にある必要があります。
- 公開することは禁止されています。
- バケット所有者は IAM ロール所有者でもある必要があります。

S3 バケットを作成するには、「Amazon S3 ユーザーガイド」の「バケットの作成」を参照してください。

監査ログ記録を有効にすると、Amazon RDS は DB インスタンスから次の場所にログを自動的に送信します。

- DB インスタンスレベルのログ – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/
- データベースレベルのログ – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/*db_name*/

バケットの Amazon リソースネーム (ARN) をメモします。この情報は、後続のステップを完了するために必要です。

ステップ 2: IAM ポリシーを作成する

DB インスタンスから Amazon S3 バケットに監査ログファイルを転送するために必要なアクセス許可を持つ IAM ポリシーを作成します。このステップは、S3 バケットがあることを前提としています。

ポリシーを作成する前に、次の情報を集めます。

- バケットの ARN。
- AWS Key Management Service (AWS KMS) キーの ARN (バケットが SSE-KMS 暗号化を使用している場合)

以下のアクセス許可を含む IAM ポリシーを作成します。

```
"s3:ListBucket",
 "s3:GetBucketAcl",
 "s3:GetBucketLocation",
 "s3:PutObject",
 "s3:ListMultipartUploadParts",
 "s3:AbortMultipartUpload",
```

```
 "s3:ListAllMyBuckets"
```

> **Note**
>
> Amazon RDS では、同じ AWS アカウント が S3 バケットと RDS for Db2 DB インスタンスの両方を所有していることを確認するために、内部での `s3:ListAllMyBuckets` アクションが必要です。

バケットが SSE-KMS 暗号化を使用している場合は、IAM ロールと AWS KMS キー用に次のアクセス許可も含めてください。

IAM ロールのポリシーに以下のアクセス許可を含めます。

```
"kms:GenerateDataKey",
 "kms:Decrypt"
```

AWS KMS キーのキーポリシーに以下のアクセス許可を含めます。*111122223333* をアカウント番号に置き換え、*AROA123456789EXAMPLE* を IAM ロール名に置き換えます。

```
{
  "Sid": "Allow RDS role to use the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:sts::111122223333:assumed-role/AROA123456789EXAMPLE/RDS-Db2Audit",
      "arn:aws:iam::111122223333:role/AROA123456789EXAMPLE"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

IAM ポリシーは、AWS マネジメントコンソール または AWS Command Line Interface (AWS CLI) を使用して作成できます。

コンソール

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ポリシーを作成するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール https://console.aws.amazon.com/iam/ を開きます。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーを作成] を選択し、[JSON] を選択します。
4. [アクションを追加] で、「S3」でフィルタリングします。アクセス ListBucket、GetBucketAcl、GetBucketLocation を追加します。
5. [リソースを追加] で、[追加] を選択します。[リソースタイプ] で、[バケット] を選択し、バケットの名前を入力します。[リソースを追加] を選択します。
6. [新しいステートメントを追加] を選択します。
7. [アクションを追加] で、「S3」でフィルタリングします。アクセス PutObject、ListMultipartUploadParts、AbortMultipartUpload を追加します。
8. [リソースを追加] で、[追加] を選択します。[リソースタイプ] で、[オブジェクト] を選択し、*####/** を入力します。[リソースを追加] を選択します。
9. [新しいステートメントを追加] を選択します。
10. [アクションを追加] で、「S3」でフィルタリングします。アクセス ListAllMyBuckets を追加します。
11. [リソースを追加] で、[追加] を選択します。[リソースタイプ] には、[すべてのリソース] を選択します。[リソースを追加] を選択します。
12. 独自の KMS キーを使用してデータを暗号化する場合:
    1. [新しいステートメントを追加] を選択します。
    2. [アクションを追加] で、「KMS」でフィルタリングします。アクセス GenerateDataKey、Decrypt を追加します。
    3. [リソースを追加] で、[追加] を選択します。[リソースタイプ] には、[すべてのリソース] を選択します。[リソースを追加] を選択します。
13. [次へ] を選択します。
14. [ポリシー名] にこのポリシーの名前を入力します。
15. (オプション) [Description (説明)] に、ポリシーの説明を入力します。
16. [Create policy] (ポリシーの作成) を選択します。

AWS CLI

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ポリシーを作成するには

1. create-policy コマンドを実行します。次の例では、*iam_policy_name* と *amzn-s3-demo-bucket* を IAM ポリシーの名前とターゲットの Amazon S3 バケットの名前に置き換えます。

   Linux、macOS、Unix の場合:

```
aws iam create-policy \
    --policy-name iam_policy_name \
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "Statement1",
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket"
                ]
            },
            {
                "Sid": "Statement2",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
                    "s3:ListMultipartUploadParts",
                    "s3:AbortMultipartUpload"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                ]
            },
            {
                "Sid": "Statement3",
                "Effect": "Allow",
                "Action": [
                    "s3:ListAllMyBuckets"
```

```
                ],
                "Resource": [
                    "*"
                ]
            },
            {
                "Sid": "Statement4",
                "Effect": "Allow",
                "Action": [
                    "kms:GenerateDataKey",
                    "kms:Decrypt"
                ],
                "Resource": [
                    "*"
                ]
            }
        ]
  }'
```

Windows の場合:

```
aws iam create-policy ^
    --policy-name iam_policy_name ^
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "Statement1",
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": [
                    "arn:aws:s3:::amzn-s3-demo-bucket"
                ]
            },
            {
                "Sid": "Statement2",
                "Effect": "Allow",
                "Action": [
                    "s3:PutObject",
```

```
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Statement4",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}'
```

2. ポリシーが作成されたら、ポリシーの ARN を書き留めます。ステップ 3: IAM ロールを作成して IAM ポリシーをアタッチする の ARN が必要です。

IAM ポリシーの作成については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

ステップ 3: IAM ロールを作成して IAM ポリシーをアタッチする

このステップは、IAM ポリシーを「ステップ 2: IAM ポリシーを作成する」で作成したことを前提としています。このステップでは、RDS for Db2 DB インスタンスの IAM ロールを作成し、ロールに IAM ポリシーをアタッチします。

コンソールまたは AWS CLI を使用して、DB インスタンスの IAM ロールを作成できます。

コンソール

IAM ロールを作成して IAM ポリシーをアタッチするには

1. AWS マネジメントコンソール にサインインして、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで [Roles] (ロール) を選択します。
3. [ロールの作成] を選択します。
4. [信頼されたエンティティタイプ] で、[AWS のサービス] を選択します。
5. [サービスまたはユースケース] で、[RDS] を選択し、[RDS - データベースにロールを追加] を選択します。
6. [次へ] を選択します。
7. [アクセス許可ポリシー] で、作成した IAM ポリシーの名前を検索して選択します。
8. [次へ] を選択します。
9. [ロール名] に、ロールの名前を入力します。
10. (オプション) [説明] には、新しいロールの説明を入力します。
11. [ロールの作成] を選択します。

AWS CLI

IAM ロールを作成して IAM ポリシーをアタッチするには

1. create-role コマンドを実行します。次の例では、*iam_role_name* を IAM ロールの名前に置き換えます。

   Linux、macOS、Unix の場合:

```
aws iam create-role \
    --role-name iam_role_name \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

Windows の場合:

```
aws iam create-role ^
    --role-name iam_role_name ^
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "rds.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

2. ロールが作成されたら、このロールの ARN を書き留めます。この ARN は次のステップ「ステップ 4: Db2 監査ログ記録のオプショングループを設定する」で必要になります。
3. attach-role-policy コマンドを実行します。次の例では、*iam_policy_arn* を ステップ 2: IAM ポリシーを作成する で作成した IAM ポリシーの ARN に置き換えます。*iam_role_name* は、先ほど作成した IAM ロールの名前に置き換えます。

   Linux、macOS、Unix の場合:

```
aws iam attach-role-policy \
   --policy-arn iam_policy_arn \
   --role-name iam_role_name
```

   Windows の場合:

```
aws iam attach-role-policy ^
   --policy-arn iam_policy_arn ^
   --role-name iam_role_name
```

詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

ステップ 4: Db2 監査ログ記録のオプショングループを設定する

Db2 監査ログ記録オプションを RDS for Db2 DB インスタンスに追加するプロセスは次のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. すべての必須オプションを追加して構成します。
3. オプショングループを DB インスタンスに関連付けます。

Db2 監査ログ記録オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになった時点で、監査を作成して監査ログを S3 バケットに保存できるようになります。

DB インスタンスのオプショングループに Db2 監査ログ記録を追加して設定するには

1. 次のいずれかを選択します。
   - 既存のオプショングループを使用します。
   - カスタムの DB オプショングループを作成し、そのオプショングループを使用します。詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [DB2_AUDIT] オプションを追加し、オプション設定を構成します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
   - [IAM_ROLE_ARN] に、「the section called “IAM ロールを作成して IAM ポリシーをアタッチする”」で作成した IAM ロールの ARN を入力します。
   - [S3_BUCKET_ARN] に、Db2 監査ログに使用する S3 バケットの ARN を入力します。バケットは RDS for Db2 DB インスタンスと同じリージョンにある必要があります。入力した IAM ロールに関連付けられたポリシーでは、このリソースで必要なオペレーションが許可されている必要があります。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。次のいずれかを選択します。

- 新しい DB インスタンスを作成する場合は、インスタンスを起動するときにオプショングループを適用します。
- 既存の DB インスタンスで、インスタンスを変更し、新しいオプショングループをアタッチして、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

ステップ 5: 監査ポリシーを設定する

RDS for Db2 DB データベースの監査ポリシーを設定するには、RDS for Db2 インスタンスのマスターユーザー名とマスターパスワードを使用して rdsadmin データベースに接続します。次に、データベースの DB 名と該当するパラメータ値を使用して、rdsadmin.configure_db_audit ストアドプロシージャを呼び出します。

次の例では、データベースに接続
し、AUDIT、CHECKING、OBJMAINT、SECMAINT、SYSADMIN、VALIDATE のカテゴリを持つ監査ポリシーを testdb に対して設定します。ステータス値 BOTH は成功と失敗をログに記録します。ERROR TYPE はデフォルトで NORMAL です。このストアドプロシージャを使用する方法の詳細については、「the section called “rdsadmin.configure_db_audit”」を参照してください。

```
db2 "connect to rdsadmin user master_user using master_password"
db2 "call rdsadmin.configure_db_audit('testdb', 'ALL', 'BOTH', ?)"
```

ステップ 6: 監査設定を確認する

監査ポリシーが正しく設定されていることを確認するには、監査設定のステータスを確認します。

設定を確認するには、RDS for Db2 DB インスタンスのマスターユーザー名とマスターパスワードを使用して、rdsadmin データベースに接続します。次に、データベースの DB 名を使用して次の SQL ステートメントを実行します。次の例では、DB 名は *testdb* です。

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(null,'testdb','CONFIGURE_DB_AUDIT'))"

Sample Output

TASK_ID              TASK_TYPE            DATABASE_NAME       LIFECYCLE
-------------------- -------------------- --------------- -------------
```

```
                    2  CONFIGURE_DB_AUDIT            DB2DB       SUCCESS

... continued ...
TASK_PARAMS
------------------------------------------------------------
{ "AUDIT_CATEGORY" : "ALL", "CATEGORY_SETTING" : "BOTH" }

... continued ...
                                          TASK_OUTPUT
--------------------------------------------------
2023-12-22T20:27:03.029Z Task execution has started.

2023-12-22T20:27:04.285Z Task execution has completed successfully.
```

Db2 監査ログ記録の管理

Db2 監査ログ記録を設定したら、特定のデータベースの監査ポリシーを変更したり、データベースレベルまたは DB インスタンス全体の監査ログ記録を無効にしたりできます。ログファイルがアップロードされる Amazon S3 バケットを変更することもできます。

トピック

- Db2 監査ポリシーの変更
- ログファイルの場所の変更
- Db2 監査ログ記録の無効化

Db2 監査ポリシーの変更

特定の RDS for Db2 データベースの監査ポリシーを変更するには、`rdsadmin.configure_db_audit` ストアドプロシージャを実行します。このストアドプロシージャでは、監査ポリシーのカテゴリ、カテゴリ設定、エラータイプの設定を変更できます。詳細については、「the section called “rdsadmin.configure_db_audit”」を参照してください。

ログファイルの場所の変更

ログファイルがアップロードされる Amazon S3 バケットを変更するには、次のいずれかを実行します。

- RDS for Db2 DB インスタンスにアタッチされている現在のオプショングループを変更する – 新しいバケットを指すように `DB2_AUDIT` オプションの `S3_BUCKET_ARN` 設定を更新します。また、アタッチされたオプショングループの `IAM_ROLE_ARN` 設定で指定された IAM ロールにアタッチ

された IAM ポリシーを必ず更新してください。この IAM ポリシーで、必要なアクセス許可を新しいバケットに提供する必要があります。IAM ポリシーのアクセス許可の詳細については、「IAM ポリシーを作成する」を参照してください。

- RDS for Db2 DB インスタンスを別のオプショングループにアタッチする – DB インスタンスを変更して、アタッチされているオプショングループを変更します。新しいオプショングループが、正しい S3_BUCKET_ARN および IAM_ROLE_ARN 設定で設定されていることを確認します。DB2_AUDIT オプションこれらの設定方法については、「オプショングループを設定する」を参照してください。

オプショングループを変更する場合は、変更をすぐに適用してください。詳細については、「the section called “DB インスタンスを変更する”」を参照してください。

Db2 監査ログ記録の無効化

Db2 監査ログ記録を無効にするには、次のいずれかを実行します。

- RDS for Db2 DB インスタンスの監査ログ記録を無効にする – DB インスタンスを変更し、DB2_AUDIT オプションを持つオプショングループをそのインスタンスから削除します。詳細については、「the section called “DB インスタンスを変更する”」を参照してください。
- 特定のデータベースの監査ログ記録を無効にする – 監査ログ記録を停止し、データベースの DB 名で rdsadmin.disable_db_audit を呼び出して監査ポリシーを削除します。詳細については、「the section called “rdsadmin.disable_db_audit”」を参照してください。

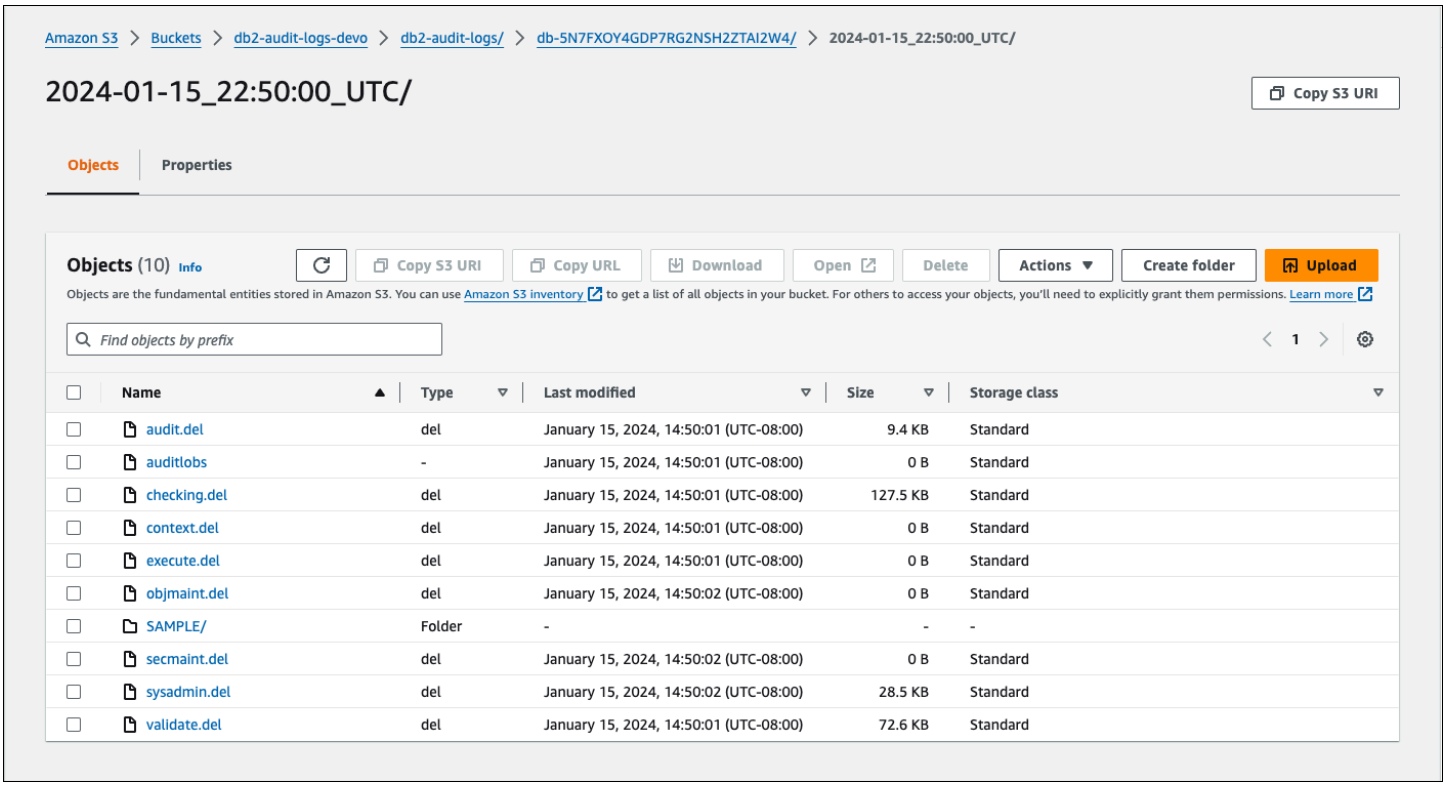
```
db2 "call rdsadmin.disable_db_audit(
    'db_name',
    ?)"
```

監査ログの表示

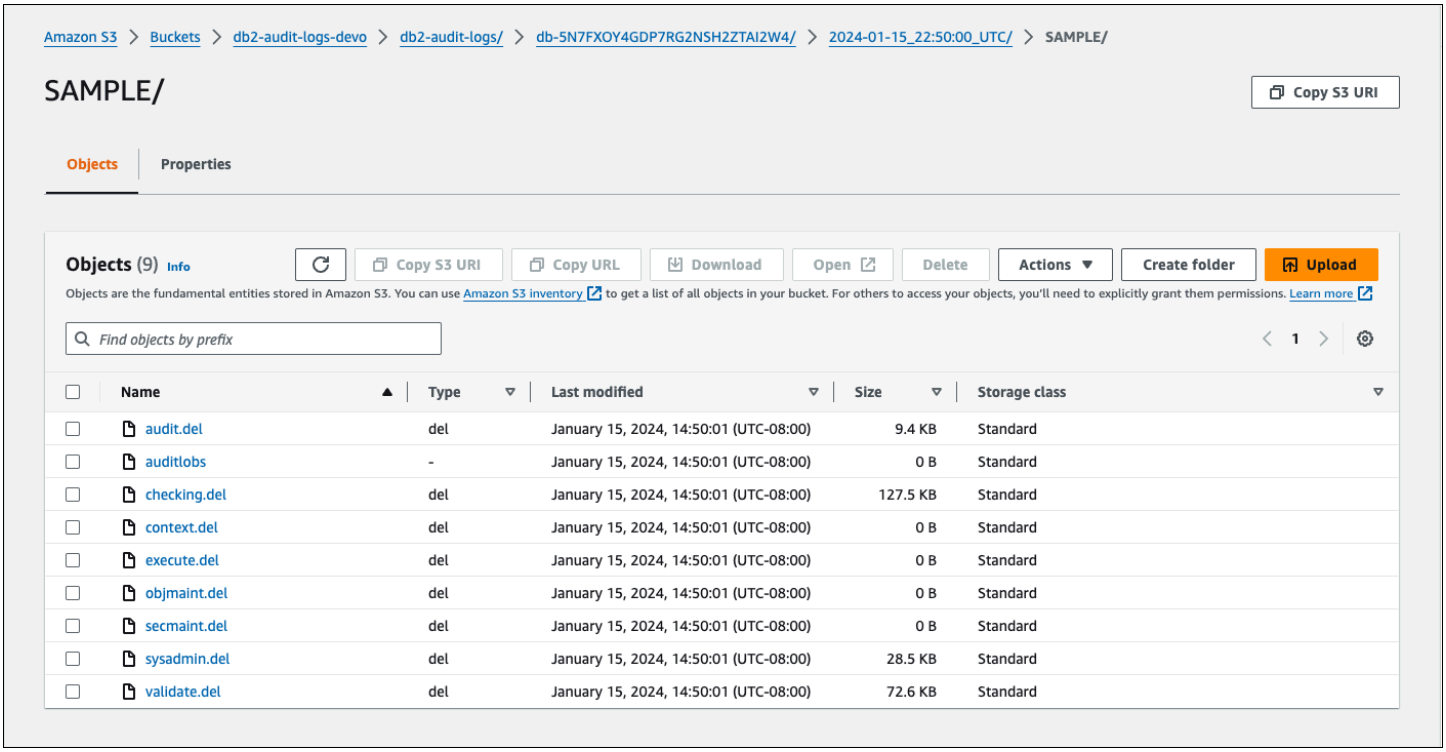
Db2 監査ログ記録を有効にしてから、Amazon S3 バケットの監査データを表示するまで少なくとも 1 時間待ちます。Amazon RDS は、RDS for Db2 DB インスタンスから次の場所にログを自動的に送信します。

- DB インスタンスレベルのログ – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/
- データベースレベルのログ – *bucket_name*/db2-audit-logs/*dbi_resource_id*/*date_time_utc*/*db_name*/

次の Amazon S3 コンソールのスクリーンショット例は、RDS for Db2 DB インスタンスレベルのログファイルのフォルダのリストを示しています。



次の Amazon S3 コンソールのスクリーンショット例は、RDS for Db2 DB インスタンスのデータベースレベルのログファイルを示しています。



監査ログ記録のトラブルシューティング

次の情報を使用して、Db2 監査ログ記録で発生する一般的な問題をトラブルシューティングします。

監査ポリシーを設定できない

ストアドプロシージャ `rdsadmin.configure_db_audit` を呼び出すとエラーが返される場合、`DB2_AUDIT` オプションを持つオプショングループが RDS for Db2 DB インスタンスに関連付けられていない可能性があります。DB インスタンスを変更してオプショングループを追加し、ストアドプロシージャを再度呼び出してください。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon S3 バケットにデータがない

Amazon S3 バケットにログ記録データがない場合は、以下を確認してください。

- Amazon S3 バケットが RDS for Db2 DB インスタンスと同じリージョンにある。
- `IAM_ROLE_ARN` オプション設定で指定したロールに、Amazon S3 バケットにログをアップロードするために必要なアクセス許可が設定されている。詳細については、「IAM ポリシーを作成する」を参照してください。

- RDS for Db2 DB インスタンスに関連付けられたオプション設定 IAM_ROLE_ARN と S3_BUCKET_ARN の ARN が正しい。詳細については、「オプショングループを設定する」を参照してください。

監査ログ記録設定のタスクステータスを確認するには、データベースに接続して SQL ステートメントを実行します。詳細については、「監査設定を確認する」を参照してください。

イベントをチェックして、ログが欠落している理由の詳細を確認することもできます。イベントを表示する方法については、「the section called “Amazon RDS コンソールでのログ、イベント、およびストリーミングの表示”」を参照してください。

Amazon RDS for Db2 の外部ストアドプロシージャ

外部ルーチンを作成し、外部ストアドプロシージャとして Amazon RDS for Db2 データベースに登録できます。現在、RDS for Db2 は外部ストアドプロシージャ用の Java ベースのルーチンのみをサポートしています。

Java ベースの外部ストアドプロシージャ

Java ベースの外部ストアドプロシージャは、外部ストアドプロシージャとして RDS for Db2 データベースに登録する外部 Java ルーチンです。

トピック

- Java ベースの外部ストアドプロシージャの制限
- Java ベースの外部ストアドプロシージャの設定

Java ベースの外部ストアドプロシージャの制限

外部ルーチンを開発する前に、次の制限を考慮してください。

外部ルーチンを作成するには、Db2 が提供する Java 開発キット (JDK) を必ず使用してください。詳細については、「Java software support for Db2 database products」を参照してください。

Java プログラムは `/tmp` ディレクトリにのみファイルを作成でき、Amazon RDS はこれらのファイルに対する実行権限や Set User ID (SUID) 権限の有効化をサポートしていません。また、Java プログラムはソケットシステム呼び出し、または次のシステム呼び出しを使用することはできません。

- _sysctl
- acct
- afs_syscall
- bpf
- capset
- chown
- chroot
- create_module
- delete_module

- fanotify_init
- fanotify_mark
- finit_module
- fsconfig
- fsopen
- fspick
- get_kernel_syms
- getpmsg
- init_module
- mount
- move_mount
- nfsservctl
- open_by_handle_at
- open_tree
- pivot_root
- putpmsg
- query_module
- quotactl
- reboot
- security
- setdomainname
- setfsuid
- sethostname
- sysfs
- tuxcall
- umount2
- uselib
- ustat
- vhangup
- vserver

Db2 の外部ルーチンに関するその他の制限については、「IBM Db2 Documentation」の「Restrictions on external routines」を参照してください。

Java ベースの外部ストアドプロシージャの設定

外部ストアドプロシージャを設定するには、外部ルーチンで.jar ファイルを作成して RDS for Db2 データベースにインストールし、外部ストアドプロシージャとして登録します。

トピック

- ステップ 1: 外部ストアドプロシージャの有効化
- ステップ 2: 外部ルーチンで.jar ファイルをインストールする
- ステップ 3: 外部ストアドプロシージャを登録する
- ステップ 4: 外部ストアドプロシージャを検証する

ステップ 1: 外部ストアドプロシージャの有効化

外部ストアドプロシージャを有効にするには、DB インスタンスに関連付けられているカスタムパラメータグループで、`db2_alternate_authz_behaviour` パラメータを次のいずれかの値に設定します。

- `EXTERNAL_ROUTINE_DBADM` — DBADM 権限を持つユーザー、グループ、またはロールに暗黙的に `CREATE_EXTERNAL_ROUTINE` 権限を付与します。
- `EXTERNAL_ROUTINE_DBAUTH` — DBADM 権限を持つユーザーが任意のユーザー、グループ、またはロールに `CREATE_EXTERNAL_ROUTINE` 権限を付与することを許可します。この場合、DBADM 権限を持つユーザーも含め、ユーザー、グループ、またはロールに暗黙的にこの権限が付与されることはありません。

この設定の詳細については、IBM Db2 ドキュメントの「GRANT (database authorities) statement」を参照してください。

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、カスタムパラメータグループを作成および変更することができます。

コンソール

カスタムパラメータグループで db2_alternate_authz_behaviour パラメータを設定するには

1. DB インスタンスが使用しているものとは異なるカスタム DB パラメータグループを使用する場合は、新しい DB パラメータグループを作成します。Bring-Your-Own-License (BYOL) モデルを使用している場合は、新しいカスタムパラメータグループに IBM ID が含まれていることを確認してください。これらの ID の詳細については、「the section called “Db2 の Bring-Your-Own-License (BYOL) の IBM ID”」を参照してください。DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. カスタムパラメータグループで db2_alternate_authz_behaviour パラメータに値を設定します。パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

AWS CLI

カスタムパラメータグループで db2_alternate_authz_behaviour パラメータを設定するには

1. DB インスタンスが使用しているものとは異なるカスタム DB パラメータグループを使用する場合は、create-db-parameter-group コマンドを実行して、カスタムパラメータグループを作成します。Bring-Your-Own-License (BYOL) モデルを使用している場合は、新しいカスタムパラメータグループに IBM ID が含まれていることを確認してください。これらの ID の詳細については、「the section called “Db2 の Bring-Your-Own-License (BYOL) の IBM ID”」を参照してください。

   次の必須パラメータを含めます。

   - `--db-parameter-group-name` - 作成するパラメータグループの名前。
   - `--db-parameter-group-family` - Db2 エンジンエディションとメジャーバージョン。有効な値は、`db2-se-11.5` および `db2-ae-11.5` です。
   - `--description` - このパラメータグループの説明。

   DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。

   次の例は、`db2-se-11.5` パラメータグループファミリーの `MY_EXT_SP_PARAM_GROUP` カスタムパラメータグループを作成する方法を示しています。

Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
--region us-east-1 \
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
--db-parameter-group-family db2-se-11.5 \
--description "test db2 external routines"
```

Windows の場合:

```
aws rds create-db-parameter-group ^
--region us-east-1 ^
--db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
--db-parameter-group-family db2-se-11.5 ^
--description "test db2 external routines"
```

2. modify-db-parameter-group コマンドを実行して、カスタムパラメータグループの db2_alternate_authz_behaviour パラメータを変更します。

   次の必須パラメータを含めます。

   - --db-parameter-group-name – 作成したパラメータグループの名前。
   - --parameters – パラメータ名、値、およびパラメータ更新用のアプリケーションメソッドの配列。

   パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

   次の例は、db2_alternate_authz_behaviour の値を EXTERNAL_ROUTINE_DBADM に設定して、MY_EXT_SP_PARAM_GROUP パラメータグループを変更する方法を示しています。

   Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name MY_EXT_SP_PARAM_GROUP \
    --parameters
 "ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',App
```

   Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name MY_EXT_SP_PARAM_GROUP ^
    --parameters
 "ParameterName='db2_alternate_authz_behaviour',ParameterValue='EXTERNAL_ROUTINE_DBADM',App
```

RDS API

カスタムパラメータグループで db2_alternate_authz_behaviour パラメータを設定するには

1. DB インスタンスが使用しているものとは異なるカスタム DB パラメータグループを使用する場合は、Amazon RDS API CreateDBParameterGroup オペレーションを使用して、新しい DB パラメータグループを作成します。Bring-Your-Own-License (BYOL) モデルを使用している場合は、新しいカスタムパラメータグループに IBM Db2 ID が含まれていることを確認してください。これらの ID の詳細については、「the section called “Db2 の Bring-Your-Own-License (BYOL) の IBM ID”」を参照してください。

   以下の必須パラメータを含めます。

   - `DBParameterGroupName`
   - `DBParameterGroupFamily`
   - `Description`

   DB パラメータグループの作成の詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。
2. RDS API ModifyDBParameterGroup オペレーションを使用して、作成したカスタムパラメータグループの `db2_alternate_authz_behaviour` パラメータを変更します。

   以下の必須パラメータを含めます。

   - `DBParameterGroupName`
   - `Parameters`

   パラメータグループの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

ステップ 2: 外部ルーチンで.jar ファイルをインストールする

Java ルーチンを作成したら、.jar ファイルを作成して db2 "call sqlj.install_jar('file:*file_path*',*jar_ID*)" を実行し、RDS for Db2 データベースにインストールします。

次の例は、Java ルーチンを作成して、RDS for Db2 データベースにインストールする方法を示しています。この例には、プロセスのテストに使用できる簡単なルーチンのサンプルコードが含まれています。この例では、次を前提としています。

- Java コードは Db2 がインストールされたサーバー上でコンパイルされていること。IBM が提供する JDK を使用してコンパイルしない場合、原因不明のエラーが発生する可能性があるため、このベストプラクティスに従ってください。
- サーバーの RDS for Db2 データベースがローカルでカタログ化されていること。

次のコード例を使用してこのプロセスを試す場合は、コード例をコピーして、MYJAVASP.java という名前のファイルに保存します。

```
import java.sql.*;
public class MYJAVASP
{
public static void my_JAVASP (String inparam) throws SQLException, Exception
{
try
{
// Obtain the calling context's connection details.
Connection myConn = DriverManager.getConnection("jdbc:default:connection");
String myQuery = "INSERT INTO TEST.TEST_TABLE VALUES (?, CURRENT DATE)";
PreparedStatement myStmt = myConn.prepareStatement(myQuery);
myStmt.setString(1, inparam);
myStmt.executeUpdate();
}
catch (SQLException sql_ex)
{
throw sql_ex;
}
catch (Exception ex)
{
throw ex;
}
}
```

次のコマンドは Java ルーチンをコンパイルします。

```
~/sqllib/java/jdk64/bin/javac MYJAVASP.java
```

次のコマンドは.jar ファイルを作成します。

```
~/sqllib/java/jdk64/bin/jar cvf MYJAVASP.jar MYJAVASP.class
```

次のコマンドは、MY_DB2_DATABASE という名前のデータベースに接続して、.jar ファイルをインストールします。

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"

db2 "call sqlj.install_jar('file:/tmp/MYJAVASP.jar','MYJAVASP')"
db2 "call sqlj.refresh_classes()"
```

ステップ 3: 外部ストアドプロシージャを登録する

RDS for Db2 データベースに.jar ファイルをインストールしたら、db2 CREATE PROCEDURE コマンドまたは db2 REPLACE PROCEDURE コマンドを実行してストアドプロシージャとして登録します。

次の例は、データベースに接続し、前のステップで作成した Java ルーチンをストアドプロシージャとして登録する方法を示しています。

```
db2 "connect to MY_DB2_DATABASE user master_username using master_password"

create procedure TESTSP.MYJAVASP (in input char(6))
specific myjavasp
dynamic result sets 0
deterministic
language java
parameter style java
no dbinfo
fenced
threadsafe
modifies sql data
program type sub
external name 'MYJAVASP!my_JAVASP';
```

ステップ 4: 外部ストアドプロシージャを検証する

次の手順を使用して、前のステップで登録したサンプル外部ストアドプロシージャをテストします。

外部ストアドプロシージャを検証するには

1. 次の例の TEST.TEST_TABLE ようなテーブルを作成します。

```
db2 "create table TEST.TEST_TABLE(C1 char(6), C2 date)"
```

2. 新しい外部ストアドプロシージャを呼び出します。呼び出しは 0 のステータスを返します。

```
db2 "call TESTSP.MYJAVASP('test')"
Return Status = 0
```

3. ステップ 1 で作成したテーブルをクエリして、ストアドプロシージャ呼び出しの結果を確認します。

```
db2 "SELECT * from TEST.TEST_TABLE"
```

クエリは次の例のような出力を生成します。

```
C1     C2
------ ----------
test   02/05/2024
```

Amazon RDS for Db2 の既知の問題と制限

Amazon RDS for Db2 を使用する際の既知の問題と制限は、以下のとおりです。

トピック

- 認証の制限
- フェンスされていないルーチン
- 移行中の非自動ストレージテーブルスペース
- db2_compatibility_vector パラメータの設定
- INVALID パッケージを含むデータベースの移行

認証の制限

Amazon RDS はデフォルトで db2auth を JCC_ENFORCE_SECMEC に設定します。ただし、ユーザー ID とパスワードの暗号化をオンラインで強制しない場合は、パラメータグループで db2auth パラメータを CLEAR_TEXT に変更することで、この設定を上書きできます。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

フェンスされていないルーチン

RDS for Db2 は、フェンスされていないルーチンの作成と、データのバックアップと復元によるこれらのルーチンの移行をサポートしていません。データベースにフェンスされていないルーチンが含まれているかどうかを確認するには、次の SQL コマンドを実行します。

```
SELECT 'COUNT:' || count(*) FROM SYSCAT.ROUTINES where fenced='N' and routineschema not
 in ('SQLJ','SYSCAT','SYSFUN','SYSIBM','SYSIBMADM','SYSPROC','SYSTOOLS')
```

移行中の非自動ストレージテーブルスペース

RDS for Db2 は、非自動ストレージテーブルスペースの新規作成をサポートしていません。データベースの 1 回限りの移行にネイティブ復元を使用すると、RDS for Db2 は、非自動ストレージテーブルスペースを自動的に自動ストレージテーブルスペースに変換して、データベースを RDS for Db2 に復元します。1 回限りの移行については、「Amazon RDS for Db2 の Linux から Linux への移行」と「Amazon RDS for Db2 の AIX または Windows から Linux への移行」を参照してください。

db2_compatibility_vector パラメータの設定

Amazon RDS では、DB インスタンスの作成時に初期データベースを作成し、関連するパラメータグループを変更できます。ただし、Db2 では、パラメータグループに db2_compatibility_vector パラメータを設定する場合は、まずカスタムパラメータグループでそのパラメータを変更し、データベースなしで DB インスタンスを作成してから、rdsadmin.create_database ストアドプロシージャを使用してデータベースを作成する必要があります。

db2_compatibility_vector パラメータを設定するには

1. カスタムパラメータグループを作成します。(デフォルトのパラメータグループを変更することはできません。)
2. パラメータを変更します。
3. DB インスタンスを作成します。
4. rdsadmin.create_database ストアドプロシージャを使用してデータベースを作成します。
5. パラメータグループを、データベースを含む DB インスタンスに関連付けます。

INVALID パッケージを含むデータベースの移行

RESTORE コマンドを使用して INVALID パッケージを含む Db2 データベースを RDS for Db2 に移行すると、データベースの使用開始時に問題が発生する可能性があります。INVALID パッケージは、DB インスタンスユーザー rdsdb の認証設定と PUBLIC からの認証の削除が原因で問題を引き起こす可能性があります。INVALID パッケージにより、以下のコマンドは失敗します。

- db2updv115
- db2 "call SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()"

RESTORE コマンドを使用してデータベースを移行する前に、次のコマンドを実行して、データベースに INVALID パッケージが含まれていないことを確認してください。

```
db2 "SELECT 'COUNT:' || count(*) FROM SYSCAT.INVALIDOBJECTS"
```

コマンドが 0 より大きい数を返す場合は、次のコマンドを呼び出します。

```
db2 "call SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()"
```

その後、前のコマンドを呼び出して、データベースに INVALID パッケージが含まれていないことを確認します。

```
db2 "SELECT 'COUNT:' || count(*) FROM SYSCAT.INVALIDOBJECTS"
```

これで、データベースのバックアップを作成し、RDS for Db2 DB インスタンスに復元する準備が整いました。

Amazon RDS for Db2 ストアドプロシージャリファレンス

Db2 エンジンを実行している Amazon RDS for Db2 DB インスタンスを管理するには、組み込みのストアドプロシージャを呼び出します。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.activate_database”	データベース	`rdsadmin.activate_database` ストアドプロシージャを使用して、スタンドアロン RDS for Db2 DB インスタンスのデータベースをアクティブ化します。
the section called “rdsadmin.add_groups”	権限の付与と取り消し	`rdsadmin.add_groups` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのユーザーに 1 つまたは複数のグループを追加します。
the section called “rdsadmin.add_user”	権限の付与と取り消し	`rdsadmin.add_user` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの承認リストにユーザーを追加します。
the section called “rdsadmin.alter_bufferpool”	バッファプール	`rdsadmin.alter_bufferpool` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのバッファプールを変更します。
the section called “rdsadmin.alter_tablespace”	テーブルスペース	`rdsadmin.alter_tablespace` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのテーブルスペースを変更します。
the section called “rdsadmin.backup_database”	データベース	`rdsadmin.backup_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースを Amazon S3 バケットへバックアップします。その後、Amazon S3 から RDS for Db2 DB インスタンスまたはロー

ストアドプロシージャ	Category	説明
		カルサーバーなどの別の場所にバックアップを復元できます。
the section called “rdsadmin.catalog_storage_access”	ストレージアクセス	`rdsadmin.catalog_storage_access` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの Db2 データファイルを含む Amazon S3 バケットにアクセスするためのストレージエイリアスをカタログ化します。
the section called “rdsadmin.change_password”	権限の付与と取り消し	`rdsadmin.change_password` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのユーザーのパスワードを変更します。
the section called “rdsadmin.complete_rollforward”	データベース	`rdsadmin.complete_rollforward` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースを `ROLL-FORWARD PENDING` 状態からオンラインにします。`ROLL-FORWARD PENDING` 状態になるのは、the section called “rdsadmin.rollforward_database” を呼び出したが、`complete_rollforward` パラメータを `FALSE` に設定した場合です。
the section called “rdsadmin.configure_db_audit”	監査ポリシー	`rdsadmin.configure_db_audit` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの監査ポリシーを変更します。監査ポリシーが存在しない場合、このストアドプロシージャを実行すると監査ポリシーが作成されます。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.create_bufferpool”	バッファプール	`rdsadmin.create_bufferpool` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのバッファプールを作成します。
the section called “rdsadmin.create_database”	データベース	`rdsadmin.create_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスにデータベースを作成します。
the section called “rdsadmin.create_role”	権限の付与と取り消し	`rdsadmin.create_role` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースにアタッチするロールを作成します。
the section called “rdsadmin.create_tablespace”	テーブルスペース	`rdsadmin.create_tablespace` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのテーブルスペースを作成します。
the section called “rdsadmin.db2pd_command”	データベース	`rdsadmin.db2pd_command` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースに関する情報を収集します。この情報を、RDS for Db2 のデータベースのモニタリングとトラブルシューティングに役立てることができます。
the section called “rdsadmin.db2support_command”	データベース	`rdsadmin.db2support_command` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの診断情報を収集し、Amazon S3 バケットにアップロードします。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.dbadm_grant”	権限の付与と取り消し	`rdsadmin.dbadm_grant` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの 1 つ以上のロール、ユーザー、またはグループに、1 つ以上の承認タイプ (DBADM、ACCESSCTRL 、または DATAACCESS) を付与します。
the section called “rdsadmin.dbadm_revoke”	権限の付与と取り消し	`rdsadmin.dbadm_revoke` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの 1 つ以上のロール、ユーザー、またはグループから、1 つ以上の承認タイプ (DBADM、ACCESSCTRL 、または DATAACCESS) を取り消します。
the section called “rdsadmin.deactivate_database”	データベース	`rdsadmin.deactivate_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースを非アクティブ化します。データベースを非アクティブ化して、メモリリソースを節約できます。
the section called “rdsadmin.disable_db_audit”	監査ポリシー	`rdsadmin.disable_db_audit` ストアドプロシージャを使用して、監査ログの記録を停止し、RDS for Db2 DB インスタンスのデータベースから監査ポリシーを削除します。
the section called “rdsadmin.drop_bufferpool”	バッファプール	`rdsadmin.drop_bufferpool` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースからバッファプールを破棄します。
the section called “rdsadmin.drop_database”	データベース	`rdsadmin.drop_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスからデータベースを破棄します。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.drop_role”	権限の付与と取り消し	`rdsadmin.drop_role` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースからロールを削除します。
the section called “rdsadmin.drop_tablespace”	テーブルスペース	`rdsadmin.drop_tablespace` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースからテーブルスペースを破棄します。
the section called “rdsadmin.fgac_command”	データベース	`rdsadmin.fgac_command` ストアドプロシージャを使用して、行または列レベルで RDS for Db2 DB インスタンスのデータベース内のテーブルデータへのアクセスを制御します。
the section called “rdsadmin.force_application”	データベース	`rdsadmin.force_application` ストアドプロシージャを使用して、アプリケーションによる RDS for Db2 DB インスタンスのデータベースのメンテナンス実行を強制的にオフにします。
the section called “rdsadmin.grant_role”	権限の付与と取り消し	`rdsadmin.grant_role` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのロール、ユーザー、またはグループにロールを割り当てます。このストアドプロシージャを使用して、ロールを割り当てるための `DBADM` 認証をロールに付与することもできます。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.list_archive_log_information”	データベース	`rdsadmin.list_archive_log_information` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのアーカイブログに関する情報を返します。この情報には、個々のログファイルのサイズや作成日、アーカイブログファイルで使用される合計ストレージなどの詳細が含まれます。
the section called “rdsadmin.list_sid_group_mapping”	権限の付与と取り消し	`rdsadmin.list_sid_group_mapping` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスに設定されているすべてのセキュリティ ID (SID) と Active Directory グループマッピングのリストを返します。
the section called “rdsadmin.list_users”	権限の付与と取り消し	`rdsadmin.list_users` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの承認リストに含まれるユーザーのリストを返します。
the section called “rdsadmin.reactivate_database”	データベース	`rdsadmin.reactivate_database` ストアドプロシージャを使用して、データベース設定の変更後に RDS for Db2 DB インスタンスのデータベースを再アクティブ化します。スタンドアロン DB インスタンスのデータベースの場合、このストアドプロシージャと rdsadmin.activate_database ストアドプロシージャのいずれかを使用できます。レプリカソース DB インスタンス上のデータベースの場合は、`rdsadmin.reactivate_database` ストアドプロシージャを使用する必要があります。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.remove_groups”	権限の付与と取り消し	`rdsadmin.remove_groups` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのユーザーから 1 つまたは複数のグループを削除します。
the section called “rdsadmin.remove_sid_group_mapping”	権限の付与と取り消し	`rdsadmin.remove_sid_group_mapping` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスからセキュリティ ID (SID) および対応する Active Directory グループマッピングを削除します。
the section called “rdsadmin.remove_user”	権限の付与と取り消し	`rdsadmin.remove_user` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの承認リストからユーザーを削除します。
the section called “rdsadmin.rename_tablespace”	テーブルスペース	`rdsadmin.rename_tablespace` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのテーブルスペースの名前を変更します。
the section called “rdsadmin.restore_database”	データベース	`rdsadmin.restore_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースを Amazon S3 バケットから復元します。
the section called “rdsadmin.revoke_role”	権限の付与と取り消し	`rdsadmin.revoke_role` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのロール、ユーザー、またはグループからロールを取り消します。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.rollforward_database”	データベース	`rdsadmin.rollforward_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースをオンラインにし、the section called “rdsadmin.restore_database” を呼び出して RDS for Db2 DB インスタンスのデータベースを復元した後にトランザクションログを適用します。
the section called “rdsadmin.rollforward_status”	データベース	`rdsadmin.rollforward_status` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスで the section called “rdsadmin.rollforward_database” または the section called “rdsadmin.complete_rollforward” ストアドプロシージャを呼び出すロールフォワードステータスをクエリします。
the section called “rdsadmin.set_archive_log_retention”	データベース	`rdsadmin.set_archive_log_retention` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのアーカイブログファイルを保持する期間を設定します。このストアドプロシージャを使用して、アーカイブログの保持を無効にすることもできます。
the section called “rdsadmin.set_configuration”	データベース	`rdsadmin.set_configuration` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースに対して何らかの設定を行います。
the section called “rdsadmin.set_sid_group_mapping”	権限の付与と取り消し	`rdsadmin.set_sid_group_mapping` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのセキュリティ ID (SID) および対応する Active Directory グループとの間のマッピングを作成します。

ストアドプロシージャ	Category	説明
the section called “rdsadmin.show_archive_log_retention”	データベース	`rdsadmin.show_archive_log_retention` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースの現在のアーカイブログ保持設定を取得します。
the section called “rdsadmin.show_configuration”	データベース	`rdsadmin.show_configuration` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースに対して変更可能な 1 つ以上の設定を取得します。
the section called “rdsadmin.uncatalog_storage_access”	ストレージアクセス	`rdsadmin.uncatalog_storage_access` ストアドプロシージャを使用して、Db2 データファイルを使用して Amazon S3 バケットにアクセスするためのストレージエイリアスを削除します。
the section called “rdsadmin.update_db_param”	データベース	`rdsadmin.update_db_param` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスのデータベースのデータベースパラメータを更新します。
the section called “rdsadmin.enable_archive_log_copy”	データベース	`rdsadmin.enable_archive_log_copy` ストアドプロシージャを使用して、RDS Db2 データベースアーカイブログの Amazon S3 へのコピーを有効にします。
the section called “rdsadmin.disable_archive_log_copy”	データベース	`rdsadmin.disable_archive_log_copy` ストアドプロシージャを使用して、RDS Db2 データベースアーカイブログの Amazon S3 へのコピーを無効にします。

トピック

- Amazon RDS for Db2 ストアドプロシージャに関する考慮事項

- RDS for Db2 の権限の付与と取り消しに関するストアドプロシージャ
- RDS for Db2 の監査ポリシーのストアドプロシージャ
- RDS for Db2 のバッファプールのストアドプロシージャ
- RDS for Db2 のデータベースのストアドプロシージャ
- RDS for Db2 のストレージアクセスのストアドプロシージャ
- RDS for Db2 のテーブルスペースのストアドプロシージャ

Amazon RDS for Db2 ストアドプロシージャに関する考慮事項

Db2 エンジンを実行している RDS for Db2 DB インスタンスの Amazon RDS システムストアドプロシージャを使用する前に、次の情報を確認してください。

- ストアドプロシージャを実行する前に、まず RDS for Db2 DB インスタンスのマスターユーザーとして rdsadmin データベースに接続する必要があります。次の例では、*master_username* と *master_password* をユーザー自身の情報に置き換えます。

  ```
  db2 "connect to rdsadmin user master_username using master_password"
  ```

- ストアドプロシージャは、ストアドプロシージャが正常に実行されたかどうか、および正常に実行されなかった理由を示す ERR_MESSAGE パラメータを返します。

  例

  次の例は、ストアドプロシージャが正常に実行されたことを示しています。

  ```
  Parameter Name : ERR_MESSAGE
  Parameter Value : -
  Return Status = 0
  ```

  次の例は、ストアドプロシージャで使用された Amazon S3 バケット名が有効でなかったため、ストアドプロシージャが正常に実行されなかったことを示しています。

  ```
  Parameter Name : ERR_MESSAGE
  Parameter Value : Invalid S3 bucket name
  Return Status = -1006
  ```

  ストアドプロシージャを呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

ストアドプロシージャのステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

RDS for Db2 の権限の付与と取り消しに関するストアドプロシージャ

このトピックで説明する組み込みのストアドプロシージャは、Amazon RDS for Db2 データベースのユーザー、ロール、グループ、承認を管理します。これらのプロシージャを実行する場合、マスターユーザーはまず rdsadmin データベースに接続する必要があります。

これらのストアドプロシージャを使用するタスクについては、「権限の付与と取り消し」と「Kerberos 認証の設定」を参照してください。

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.create_role
- rdsadmin.grant_role
- rdsadmin.revoke_role
- rdsadmin.drop_role
- rdsadmin.add_user
- rdsadmin.change_password
- rdsadmin.list_users
- rdsadmin.remove_user
- rdsadmin.add_groups
- rdsadmin.remove_groups
- rdsadmin.dbadm_grant
- rdsadmin.dbadm_revoke
- rdsadmin.set_sid_group_mapping
- rdsadmin.list_sid_group_mapping
- rdsadmin.remove_sid_group_mapping

rdsadmin.create_role

ロールを作成します。

構文

```
db2 "call rdsadmin.create_role(
```

```
    'database_name',
    'role_name')"
```

パラメータ

以下のパラメータは必須です。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

role_name

作成するロールの名前。データ型は varchar です。

使用に関する注意事項

ロールの作成ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

以下の例は、データベース DB2DB の MY_ROLE というロールを作成します。

```
db2 "call rdsadmin.create_role(
    'DB2DB',
    'MY_ROLE')"
```

rdsadmin.grant_role

ロール、ユーザー、またはグループにロールを割り当てます。

構文

```
db2 "call rdsadmin.grant_role(
    ?,
    'database_name',
    'role_name',
    'grantee',
    'admin_option')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

このタスクで一意の識別子を出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

role_name

作成するロールの名前。データ型は varchar です。

grantee

承認を受け取るロール、ユーザー、またはグループ。データ型は varchar です。有効な値は、ROLE、USER、GROUP、PUBLIC です。

形式は、値の後に名前を続ける必要があります。複数の値と名前はカンマで区切ります。例えば、「USER *user1*, *user2*, GROUP *group1*, *group2*」などです。名前を、ユーザー自身の情報に置き換えます。

次の入力パラメータは、オプションです。

admin_option

被付与者 ROLE がロールを割り当てる DBADM 権限を持っているかどうかを指定します。データ型は char です。デフォルトは N です。

使用に関する注意事項

ロールの割り当てステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: ロール、ユーザー、グループにロールを割り当て、権限を付与する

次の例では、データベース TESTDB の ROLE_TEST というロールを role1 というロール、user1 というユーザー、および group1 というグループに割り当てます。 ROLE_TEST には、ロールを割り当てる管理者権限が付与されています。

```
db2 "call rdsadmin.grant_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'ROLE role1, USER user1, GROUP group1',
    'Y')"
```

例 2: **PUBLIC** にロールを割り当て、権限を付与しない

次の例では、データベース TESTDB の ROLE_TEST というロールを PUBLIC に割り当てます。ROLE_TEST には、ロールを割り当てるための管理者権限は付与されていません。

```
db2 "call rdsadmin.grant_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'PUBLIC')"
```

rdsadmin.revoke_role

ロール、ユーザー、またはグループのロールを取り消します。

構文

```
db2 "call rdsadmin.revoke_role(
    ?,
    'database_name',
    'role_name',
    'grantee')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

このタスクで一意の識別子を出力するパラメータマーカー。このパラメーターが受け入れるのは、? のみです。

次の入力パラメータが必要となります。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

role_name

取り消すロールの名前。データ型は varchar です。

grantee

権限を失うロール、ユーザー、またはグループ。データ型は varchar です。有効な値は、ROLE、USER、GROUP、PUBLIC です。

形式は、値の後に名前を続ける必要があります。複数の値と名前はカンマで区切ります。例えば、「USER *user1*, *user2*, GROUP *group1*, *group2*」などです。名前を、ユーザー自身の情報に置き換えます。

使用に関する注意事項

ロールの取り消しステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: ロール、ユーザー、グループからロールを取り消す

次の例では、ロール role1、ユーザー user1、グループ group1 からデータベース TESTDB のロール ROLE_TEST を取り消します。

```
db2 "call rdsadmin.revoke_role(
    ?,
    'TESTDB',
    'ROLE_TEST',
    'ROLE role1, USER user1, GROUP group1')"
```

例 2: **PUBLIC** からロールを取り消す

次の例は、PUBLIC からデータベース TESTDB のロール ROLE_TEST を取り消します。

```
db2 "call rdsadmin.revoke_role(
    ?,
```

```
    'TESTDB',
    'ROLE_TEST',
    'PUBLIC')"
```

rdsadmin.drop_role

ロールを削除します。

構文

```
db2 "call rdsadmin.drop_role(
    ?,
    'database_name',
    'role_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

このタスクで一意の識別子を出力するパラメータマーカー。このパラメーターが受け入れるのは、? のみです。

次の入力パラメータが必要となります。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

role_name

削除するロールの名前。データ型は varchar です。

使用に関する注意事項

ロールの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

以下の例は、データベース TESTDB の ROLE_TEST というロールを削除します。

```
db2 "call rdsadmin.drop_role(
    ?,
    'TESTDB',
    'ROLE_TEST')"
```

rdsadmin.add_user

承認リストにユーザーを追加します。

構文

```
db2 "call rdsadmin.add_user(
    'username',
    'password',
    'group_name,group_name')"
```

パラメータ

以下のパラメータは必須です。

#####

ユーザーのユーザー名。データ型は varchar です。

######

ユーザーのパスワード。データ型は varchar です。

次のパラメータはオプションです。

group_name

ユーザーを追加するグループの名前。データ型は varchar です。デフォルトは空の文字列または null です。

使用に関する注意事項

グループ名をカンマで区切ることで、ユーザーを 1 つ以上のグループに追加できます。

新しいユーザーを作成するとき、または既存のユーザー にグループを追加するとき、グループを作成できます。グループの作成のみを行うことはできません。

> **Note**
>
> rdsadmin.add_user を呼び出すことで追加できるユーザーの最大数は 5,000 です。

ユーザーの追加ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、jorge_souza という名前のユーザーを作成し、このユーザーを sales グループと inside_sales グループに割り当てます。

```
db2 "call rdsadmin.add_user(
    'jorge_souza',
    '*******',
    'sales,inside_sales')"
```

rdsadmin.change_password

ユーザーのパスワードを変更します。

構文

```
db2 "call rdsadmin.change_password(
    'username',
    'new_password')"
```

パラメータ

以下のパラメータは必須です。

#####

ユーザーのユーザー名。データ型は varchar です。

new_password

ユーザーの新しいパスワード。データ型は varchar です。

使用に関する注意事項

パスワードの変更ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例は、jorge_souza のパスワードを変更します。

```
db2 "call rdsadmin.change_password(
    'jorge_souza',
    '*******')"
```

rdsadmin.list_users

承認リストにユーザーを一覧表示します。

構文

```
db2 "call rdsadmin.list_users()"
```

使用に関する注意事項

ユーザーの一覧表示ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.remove_user

承認リストからユーザーを削除します。

構文

```
db2 "call rdsadmin.remove_user('username')"
```

パラメータ

以下のパラメータは必須です。

#####

ユーザーのユーザー名。データ型は varchar です。

使用に関する注意事項

ユーザーの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、 RDS for Db2 DB インスタンスのデータベースに jorge_souza がアクセスできないようにします。

```
db2 "call rdsadmin.remove_user('jorge_souza')"
```

rdsadmin.add_groups

グループをユーザーに追加します。

構文

```
db2 "call rdsadmin.add_groups(
    'username',
    'group_name,group_name')"
```

パラメータ

以下のパラメータは必須です。

#####

ユーザーのユーザー名。データ型は varchar です。

group_name

ユーザーを追加するグループの名前。データ型は varchar です。デフォルトは空の文字列です。

使用に関する注意事項

グループ名をカンマで区切ることで、1 つ以上のグループをユーザーに追加できます。グループの追加ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例は、direct_sales および b2b_sales グループをユーザー jorge_souza に追加します。

```
db2 "call rdsadmin.add_groups(
    'jorge_souza',
    'direct_sales,b2b_sales')"
```

rdsadmin.remove_groups

ユーザーからグループを削除します。

構文

```
db2 "call rdsadmin.remove_groups(
    'username',
    'group_name,group_name')"
```

パラメータ

以下のパラメータは必須です。

#####

ユーザーのユーザー名。データ型は varchar です。

group_name

ユーザーを削除するグループの名前。データ型は varchar です。

使用に関する注意事項

グループ名をカンマで区切ることで、1 つ以上のグループをユーザーから削除できます。

グループの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例は、direct_sales および b2b_sales グループをユーザー jorge_souza から削除します。

```
db2 "call rdsadmin.remove_groups(
    'jorge_souza',
```

```
    'direct_sales,b2b_sales')"
```

rdsadmin.dbadm_grant

ロール、ユーザー、またはグループに DBADM、ACCESSCTRL、または DATAACCESS 権限を付与します。

構文

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

このタスクで一意の識別子を出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

authorization, *

付与する権限のタイプ。データ型は varchar です。有効な値は、DBADM、ACCESSCTRL、DATAACCESS です。

タイプが複数の場合、カンマで区切ります。

grantee

承認を受け取るロール、ユーザー、またはグループ。データ型は varchar です。有効な値は、ROLE、USER、GROUP です。

形式は、値の後に名前を続ける必要があります。複数の値と名前はカンマで区切ります。例えば、「USER *user1*, *user2*, GROUP *group1*, *group2*」などです。名前を、ユーザー自身の情報に置き換えます。

使用に関する注意事項

アクセス権を受け取るロールが存在している必要があります。

データベース管理者アクセスの許可ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: データベース管理者アクセスをロールに付与する

次の例では、ロール ROLE_DBA に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを許可します。

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'TESTDB',
    'DBADM',
    'ROLE ROLE_DBA')"
```

例 2: ユーザーとグループにデータベース管理者アクセスを付与する

次の例では、user1 および group1 に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを許可します。

```
db2 "call rdsadmin.dbadm_grant(
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, GROUP group1')"
```

例 3: 複数のユーザーとグループにデータベース管理者のアクセス権を付与する

次の例では、user1、user2、group1、group2 に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを許可します。

```
db2 "call rdsadmin.dbadm_grant(
```

```
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, user2, GROUP group1, group2')"
```

rdsadmin.dbadm_revoke

ロール、ユーザー、またはグループの DBADM、ACCESSCTRL、または DATAACCESS 権限を取り消します。

構文

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'database_name',
    'authorization',
    'grantee')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

タスクの一意の識別子。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

コマンドを実行する対象データベースの名前。データ型は varchar です。

authorization, *

取り消す権限のタイプ。データ型は varchar です。有効な値は、DBADM、ACCESSCTRL、DATAACCESS です。

タイプが複数の場合、カンマで区切ります。

grantee

権限を受け取るロール、ユーザー、またはグループ。データ型は varchar です。有効な値は、ROLE、USER、GROUP です。

形式は、値の後に名前を続ける必要があります。複数の値と名前はカンマで区切ります。例えば、「USER *user1*, *user2*, GROUP *group1*, *group2*」などです。名前を、ユーザー自身の情報に置き換えます。

使用に関する注意事項

データベース管理者アクセスの取り消しステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: ロールからデータベース管理者アクセスを取り消す

次の例では、ロール ROLE_DBA に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを取り消します。

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'TESTDB',
    'DBADM',
    'ROLE ROLE_DBA')"
```

例 2: ユーザーとグループからデータベース管理者アクセスを取り消す

次の例では、user1 および group1 に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを取り消します。

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
    'TESTDB',
    'DBADM',
    'USER user1, GROUP group1')"
```

例 3: 複数のユーザーおよびグループからデータベース管理者アクセスを取り消す

次の例では、user1、user2、group1、group2 に対して、TESTDB という名前のデータベースへのデータベース管理者アクセスを取り消します。

```
db2 "call rdsadmin.dbadm_revoke(
    ?,
```

```
    'TESTDB',
    'DBADM',
    'USER user1, user2, GROUP group1, group2')"
```

rdsadmin.set_sid_group_mapping

セキュリティ ID (SID) および対応する Active Directory グループ間のマッピングを作成します。

構文

```
db2 "call rdsadmin.set_sid_group_mapping(
    ?,
    'SID',
    'group_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

SID ()

セキュリティ ID (SID)。データ型は `varchar` です。

group_name

SID にマッピングする Active Directory グループの名前。データ型は `varchar` です。

使用に関する注意事項

Active Directory グループで Kerberos 認証を有効にするには、このストアドプロシージャを使用します。マッピングに `SID` または `group_name` が既に存在する場合、このストアドプロシージャは失敗します。

グループの SID を検索する方法については、「ステップ 8: PowerShell で Active Directory グループ SID を取得する」を参照してください。

マッピングの作成のステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、SID を my_group というグループにマッピングします。

```
db2 "call rdsadmin.set_sid_group_mapping(
    ?,
    'S-1-5-21-9146495592-531070549-834388463-513',
    'my_group')"
```

rdsadmin.list_sid_group_mapping

DB インスタンスで設定されているすべてのセキュリティ ID (SID) および Active Directory グループのマッピングを一覧表示します。

構文

```
db2 "call rdsadmin.list_sid_group_mapping()"
```

使用に関する注意事項

ユーザーの一覧表示のステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.remove_sid_group_mapping

DB インスタンスからセキュリティ ID (SID) および対応する Active Directory グループマッピングを削除します。

構文

```
db2 "call rdsadmin.remove_sid_group_mapping(
    ?,
    'SID')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

SID ()

セキュリティ ID (SID)。データ型は varchar です。

使用に関する注意事項

グループの SID を検索する方法については、「ステップ 8: PowerShell で Active Directory グループ SID を取得する」を参照してください。

マッピングの削除のステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、マッピング先のグループから SID マッピングを削除します。

```
db2 "call rdsadmin.remove_sid_group_mapping(
    ?,
    'S-1-5-21-9146495592-531070549-834388463-513')"
```

RDS for Db2 の監査ポリシーのストアドプロシージャ

このトピックで説明した組み込みのストアドプロシージャでは、監査ログ記録を使用する Amazon RDS for Db2 データベースの監査ポリシーを管理します。詳細については、「the section called "Db2 監査ログ記録"」を参照してください。これらのプロシージャを実行する場合、マスターユーザーはまず `rdsadmin` データベースに接続する必要があります。

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.configure_db_audit
- rdsadmin.disable_db_audit

rdsadmin.configure_db_audit

db_name で指定された RDS for Db2 データベースの監査ポリシーを設定します。設定するポリシーが存在しない場合は、このストアドプロシージャを呼び出すとポリシーが作成されます。このポリシーが存在する場合、このストアドプロシージャを呼び出すと、指定したパラメータ値でポリシーが変更されます。

構文

```
db2 "call rdsadmin.configure_db_audit(
    'db_name',
    'category',
    'category_setting',
    '?')"
```

パラメータ

以下のパラメータは必須です。

db_name

db_name で指定された監査ポリシーを設定する RDS for Db2 データベースの DB 名。データ型は `varchar` です。

category

この監査ポリシーを設定するカテゴリの名前。データ型は varchar です。このパラメータには、次の値が有効です。

- ALL – ALL では、Amazon RDS に CONTEXT、EXECUTE、または ERROR カテゴリは含まれません。
- AUDIT
- CHECKING
- CONTEXT
- ERROR
- EXECUTE - このカテゴリは、データありまたはデータなしで設定できます。データありでは、ホスト変数とパラメータマーカーに指定された入力データ値もログに記録されます。デフォルトでは、データなしになっています。詳細については、*category_setting* パラメータの説明と「the section called “例”」を参照してください。
- OBJMAINT
- SECMAINT
- SYSADMIN
- VALIDATE

これらのコンポーネントの詳細については、IBM Db2 のドキュメントを参照してください。

category_setting

指定された監査カテゴリの設定。データ型は varchar です。

次の表は、各カテゴリの有効なカテゴリ設定値を示しています。

Category	有効なカテゴリ設定
ALL	BOTH|FAILURE|SUCCESS|NONE
AUDIT	
CHECKING	
CONTEXT	
OBJMAINT	

Category	有効なカテゴリ設定
SECMAINT SYSADMIN VALIDATE	
ERROR	AUDIT|NORMAL 。デフォルト値は NORMAL です。
EXECUTE	BOTH,WITH|BOTH,WITHOUT|FAILURE,WITH| FAILURE,WITHOUT|SUCCESS,WITH|SUCCESS ,WITHOUT|NONE

使用に関する注意事項

rdsadmin.configure_db_audit を呼び出す前に、監査ポリシーの設定対象となるデータベースを持つ RDS for Db2 DB インスタンスが、DB2_AUDIT オプションのあるオプショングループに関連付けられていることを確認してください。詳細については、「the section called “Db2 監査ログのセットアップ”」を参照してください。

監査ポリシーを設定したら、「監査設定を確認する」の手順に従ってデータベースの監査設定のステータスを確認できます。

category パラメータに ALL を指定しても CONTEXT、EXECUTE、ERROR カテゴリは含まれません。これらのカテゴリを監査ポリシーに追加するには、追加するカテゴリごとに個別に rdsadmin.configure_db_audit を呼び出します。詳細については、「the section called “例”」を参照してください。

例

以下の例では、TESTDB という名前のデータベースの監査ポリシーを作成または変更します。例 1 ~ 5 では、ERROR カテゴリが以前に設定されていない場合、このカテゴリは NORMAL (デフォルト) に設定されます。その設定を AUDIT に変更するには、「Example 6: Specifying the ERROR category」に従います。

例 1: **ALL** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ALL', 'BOTH', ?)"
```

この例では、呼び出しで監査ポリシーの
AUDIT、CHECKING、OBJMAINT、SECMAINT、SYSADMIN、VALIDATE カテゴリを設定します。BOTH を指定すると、成功したイベントと失敗したイベントの両方に対してこの各カテゴリについての監査が行われます。

例 2: データを使用して **EXECUTE** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'SUCCESS,WITH', ?)"
```

この例では、呼び出しで監査ポリシーの EXECUTE カテゴリを設定します。SUCCESS,WITH を指定すると、このカテゴリのログには成功したイベントのみが含まれ、ホスト変数とパラメータマーカーに対して指定された入力データ値が含まれます。

例 3: データなしで **EXECUTE** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'FAILURE,WITHOUT', ?)"
```

この例では、呼び出しで監査ポリシーの EXECUTE カテゴリを設定します。FAILURE,WITHOUT を指定すると、このカテゴリのログには失敗したイベントのみが含まれ、ホスト変数とパラメータマーカーに対して指定された入力データ値は含まれません。

例 4: ステータスイベントなしで **EXECUTE** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'EXECUTE', 'NONE', ?)"
```

この例では、呼び出しで監査ポリシーの EXECUTE カテゴリを設定します。NONE を指定すると、このカテゴリのイベントは監査されません。

例 5: **OBJMAINT** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'OBJMAINT', 'NONE', ?)"
```

この例では、呼び出しで監査ポリシーの OBJMAINT カテゴリを設定します。NONE を指定すると、このカテゴリのイベントは監査されません。

例 6: **ERROR** カテゴリを指定する

```
db2 "call rdsadmin.configure_db_audit('TESTDB', 'ERROR', 'AUDIT', ?)"
```

この例では、呼び出しで監査ポリシーの ERROR カテゴリを設定します。AUDIT を指定すると、監査ログ記録自体で発生するエラーを含むすべてのエラーがログにキャプチャされます。デフォルトのエラータイプは NORMAL です。NORMAL では、監査によって生成されたエラーは無視され、実行中のオペレーションに関連付けられたエラーの SQLCODE のみがキャプチャされます。

rdsadmin.disable_db_audit

db_name で指定された RDS for Db2 データベースの監査ログ記録を停止し、そのデータベースに対して設定された監査ポリシーを削除します。

> **Note**
>
> このストアドプロシージャは、the section called “rdsadmin.configure_db_audit” の呼び出しによって設定された監査ポリシーのみを削除します。

構文

```
db2 "call rdsadmin.disable_db_audit('db_name', ?)"
```

パラメータ

以下のパラメータは必須です。

db_name

監査ログ記録を無効にする RDS for Db2 データベースの DB 名。データ型は varchar です。

使用に関する注意事項

rdsadmin.disable_db_audit を呼び出しても、RDS for Db2 DB インスタンスの監査ログ記録は無効になりません。DB インスタンスレベルで監査ログ記録を無効にするには、DB インスタンスからオプショングループを削除します。詳細については、「Db2 監査ログ記録の無効化」を参照してください。

例

次の例では、TESTDB という名前のデータベースの監査ログ記録を無効にします。

```
db2 "call rdsadmin.disable_db_audit('TESTDB', ?)"
```

RDS for Db2 のバッファプールのストアドプロシージャ

このトピックで説明した組み込みのストアドプロシージャは、Amazon RDS for Db2 データベースのバッファプールを管理します。これらのプロシージャを実行する場合、マスターユーザーはまず `rdsadmin` データベースに接続する必要があります。

これらのストアドプロシージャは、さまざまなタスクで使用されます。これはすべてを網羅したリストではありません。

- バッファプールの一般的なタスク
- パフォーマンスレポートの生成
- db2look を使用してデータベースメタデータをコピーする
- IBM Db2 Data Management Console 用のリポジトリデータベースの作成

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.create_bufferpool
- rdsadmin.alter_bufferpool
- rdsadmin.drop_bufferpool

rdsadmin.create_bufferpool

バッファプールを作成します。

構文

```
db2 "call rdsadmin.create_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    page_size,
    number_block_pages,
    block_size)"
```

パラメータ

以下のパラメータは必須です。

database_name

コマンドを実行する対象のデータベース名。データ型は varchar です。

buffer_pool_name

作成するバッファプールの名前。データ型は varchar です。

以下のパラメータはオプションです。

buffer_pool_size

ページ数でのバッファプールのサイズ。データ型は integer です。デフォルトは -1 です。

immediate

コマンドをすぐに実行するかどうかを指定します。データ型は char です。デフォルトは Y です。

automatic

バッファプールを自動的に設定するかを指定します。データ型は char です。デフォルトは Y です。

page_size

バッファプールのページサイズ。データ型は integer です。有効な値は、4096、8192、16384、32768 です。デフォルトは 8192 です。

number_block_pages

バッファプールのブロックページ数。データ型は integer です。デフォルトは 0 です。

block_size

ブロックページのブロックサイズ。データ型は integer です。有効な値: 2 から 256。デフォルトは 32 です。

使用に関する注意事項

バッファプールの作成ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: デフォルトパラメータを使用してバッファプールを作成する

次の例では、デフォルトのパラメータを使用して TESTDB というデータベースに BP8 というバッファプールを作成するため、バッファプールでは 8 KB のページサイズが使用されます。

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP8')"
```

例 2: 自動割り当てを使用してバッファプールを作成し、すぐに実行する

次の例では、TESTDB という名前のデータベースのために BP16 という名前のバッファプールを作成します。設定は、ページサイズが 16 KB、初期ページ数は 1,000 で、自動と指定しています。Db2 はこのコマンドを直ちに実行します。初期ページ数 -1 を使用すると、Db2 ではページの自動割り当てを使用します。

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP16',
    1000,
    'Y',
    'Y',
    16384)"
```

例 3: ブロックページを使用してバッファプールを作成し、すぐに実行する

次の例では、TESTDB という名前のデータベースのために BP16 という名前のバッファプールを作成します。このバッファプールのページサイズは 16 KB で、初期ページ数は 10,000 ページです。Db2 は、ブロックサイズ 512 の 500 ブロックページを使用して、このコマンドを直ちに実行します。

```
db2 "call rdsadmin.create_bufferpool(
    'TESTDB',
    'BP16',
    10000,
    'Y',
    'Y',
    16384,
    500,
    512)"
```

rdsadmin.alter_bufferpool

バッファプールを変更します。

構文

```
db2 "call rdsadmin.alter_bufferpool(
    'database_name',
    'buffer_pool_name',
    buffer_pool_size,
    'immediate',
    'automatic',
    change_number_blocks,
    number_block_pages,
    block_size)"
```

パラメータ

以下のパラメータは必須です。

database_name

コマンドを実行する対象のデータベース名。データ型は varchar です。

buffer_pool_name

変更するバッファプールの名前。データ型は varchar です。

buffer_pool_size

ページ数でのバッファプールのサイズ。データ型は integer です。

以下のパラメータはオプションです。

immediate

コマンドをすぐに実行するかどうかを指定します。データ型は char です。デフォルトは Y です。

automatic

バッファプールを自動的に設定するかを指定します。データ型は char です。デフォルトは N です。

change_number_blocks

バッファプール内のブロックページ数を変更するかどうかを指定します。データ型は char です。デフォルトは N です。

number_block_pages

バッファプールのブロックページ数。データ型は integer です。デフォルトは 0 です。

block_size

ブロックページのブロックサイズ。データ型は integer です。有効な値: 2 から 256。デフォルトは 32 です。

使用に関する注意事項

バッファプールの変更ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、TESTDB というデータベースの BP16 というバッファプールを「非自動」に変更し、サイズを 10,000 ページに変更します。Db2 はこのコマンドを直ちに実行します。

```
db2 "call rdsadmin.alter_bufferpool(
    'TESTDB',
    'BP16',
    10000,
    'Y',
    'N')"
```

rdsadmin.drop_bufferpool

バッファプールを削除します。

構文

```
db2 "call rdsadmin.drop_bufferpool(
    'database_name',
    'buffer_pool_name'"
```

パラメータ

以下のパラメータは必須です。

database_name

バッファプールが属するデータベースの名前。データ型は varchar です。

buffer_pool_name

削除するバッファプールの名前。データ型は varchar です。

使用に関する注意事項

バッファプールの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、TESTDB というデータベースの BP16 というバッファプールを削除します。

```
db2 "call rdsadmin.drop_bufferpool(
    'TESTDB',
    'BP16')"
```

RDS for Db2 のデータベースのストアドプロシージャ

このトピックで説明した組み込みのストアドプロシージャは、Amazon RDS for Db2 のデータベースを管理します。これらのプロシージャを実行する場合、マスターユーザーはまず `rdsadmin` データベースに接続する必要があります。

これらのストアドプロシージャは、さまざまなタスクで使用されます。これはすべてを網羅したリストではありません。

- 一般的なデータベースタスク
- EBCDIC 照合を使用したデータベースの作成
- データベースに関する情報の収集
- データベース設定パラメータの変更
- Linux から Linux への移行
- ほぼゼロのダウンタイムで Linux から Linux に移行する

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.create_database
- rdsadmin.deactivate_database
- rdsadmin.activate_database
- rdsadmin.reactivate_database
- rdsadmin.drop_database
- rdsadmin.update_db_param
- rdsadmin.set_configuration
- rdsadmin.show_configuration
- rdsadmin.backup_database
- rdsadmin.restore_database
- rdsadmin.rollforward_database
- rdsadmin.rollforward_status
- rdsadmin.complete_rollforward
- rdsadmin.db2pd_command

- rdsadmin.force_application
- rdsadmin.set_archive_log_retention
- rdsadmin.show_archive_log_retention
- rdsadmin.list_archive_log_information
- rdsadmin.enable_archive_log_copy
- rdsadmin.disable_archive_log_copy
- rdsadmin.fgac_command
- rdsadmin.db2support_command

rdsadmin.create_database

データベースを作成します。

構文

```
db2 "call rdsadmin.create_database(
    'database_name',
    'database_page_size',
    'database_code_set',
    'database_territory',
    'database_collation',
    'database_autoconfigure_str',
    'database_non-restrictive')"
```

パラメータ

> **Note**
>
> このストアドプロシージャでは、必須パラメータの組み合わせは検証されません。rdsadmin.get_task_status を呼び出すと、database_codeset、database_territory、database_collation の組み合わせが無効であるため、ユーザー定義関数がエラーを返す可能性があります。詳細については、 IBM Db2 ドキュメントの「Choosing the code page, territory, and collation for your database」を参照してください。

以下のパラメータは必須です。

database_name

作成するデータベースの名前。データ型は varchar です。

以下のパラメータはオプションです。

database_page_size

データベースのデフォルトのページサイズ。有効な値は、4096、8192、16384、32768 です。データ型は integer です。デフォルトは 8192 です。

> **Important**
>
> Amazon RDS は、4 KiB、8 KiB、16 KiB のページでの書き込みの原子性をサポートしています。一方、32 KiB のページには、Torn Writes (一部のデータしかディスクに書き込まれない) というリスクがあります。32 KiB のページを使用している場合は、ポイントインタイムリカバリと自動バックアップを有効にすることをお勧めします。有効にしない場合、データの不整合があるページから復元できなくなるリスクが発生します。詳細については、「the section called “バックアップの概要”」および「the section called “ポイントインタイムリカバリ”」を参照してください。

database_code_set

データベースのコードセット。データ型は varchar です。デフォルトは UTF-8 です。

database_territory

データベースの 2 文字の国コード。データ型は varchar です。デフォルトは US です。

database_collation

データベースに保存されている文字列をソートして比較する方法を決定する照合順序。データ型は varchar です。

有効な値:

- COMPATIBILITY – IBM Db2 バージョン 2 の照合順序。
- EBCDIC_819_037 – ISO ラテンコードページ、照合、CCSID 037 (EBCDIC 米国英語)。
- EBCDIC_819_500 – ISO ラテンコードページ、照合、CCSID 500 (EBCDIC 国際標準)。
- EBCDIC_850_037 – ASCII ラテンコードページ、照合、CCSID 037 (EBCDIC 米国英語)。

- EBCDIC_850_500 – ASCII ラテンコードページ、照合、CCSID 500 (EBCDIC 国際標準)。
- EBCDIC_932_5026 – ASCII 日本語コードページ、照合、CCSID 5026 (EBCDIC 米国英語)。
- EBCDIC_932_5035 – ASCII 日本語コードページ、照合、CCSID 5035 (EBCDIC 国際標準)。
- EBCDIC_1252_037 – Windows ラテンコードページ、照合、CCSID 037 (EBCDIC 米国英語)。
- EBCDIC_1252_500 – Windows ラテンコードページ、照合、CCSID 500 (EBCDIC 国際標準)。
- IDENTITY - デフォルトの照合順序。文字列はバイト単位で比較されます。
- IDENTITY_16BIT – Compatibility Encoding Scheme for UTF-16: 8-bit (CESU-8) 照合順序。詳細については、Unicode Consortium ウェブサイトの「Unicode Technical Report #26」を参照してください。
- NLSCHAR – タイ語のコードページ (CP874) でのみ使用されます。
- SYSTEM – SYSTEM を使用する場合、データベースは database_codeset と database_territory で自動的に照合順序を使用します。

デフォルトは IDENTITY です。

さらに、RDS for Db2 は照合順序のグループ language-aware-collation および locale-sensitive-collation をサポートしています。詳細については、 IBM Db2 ドキュメントの「Choosing a collation for a Unicode database」を参照してください。

database_autoconfigure_str

AUTOCONFIGURE コマンド構文 ('AUTOCONFIGURE APPLY DB' など)。データ型は varchar です。デフォルトは空の文字列または null です。

詳細については、IBM Db2 ドキュメントの「AUTOCONFIGURE コマンド」を参照してください。

database_non-restrictive

データベース内のデフォルトの権限と特権の付与。データ型は varchar です。デフォルトは N です。

有効な値:

- N – 作成されたデータベースは制限されており、権限や特権を付与しません。
- Y – 作成されたデータベースは制限がなく、特別なグループ PUBLIC にアクセス許可のセットを付与します。詳細については、「IBM Db2 ドキュメント」の「Default privileges granted on creating a database」を参照してください。

使用に関する注意事項

db2_compatibility_vector パラメータを変更する場合は、データベースを作成する前に変更します。詳細については、「db2_compatibility_vector パラメータの設定」を参照してください。

特別な考慮事項:

- Db2 インスタンスに送信される CREATE DATABASE コマンドでは、RESTRICTIVE オプションを使用します。
- RDS for Db2 は AUTOMATIC STORAGE テーブルスペースのみを使用します。
- RDS for Db2 では、 NUMSEGS と DFT_EXTENT_SZ にデフォルト値が使用されます。
- RDS for Db2 ではストレージ暗号化が使用され、データベース暗号化はサポートされていません。

これらのコマンドの詳細については、 IBM Db2 ドキュメントの「CREATE DATABASE command」を参照してください。

rdsadmin.create_database を呼び出す前に、rdsadmin データベースに接続する必要があります。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンス情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

データベースの作成ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.create_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、日本向けの *database_code_set* パラメータ、*database_territory* パラメータ、および *database_collation* パラメータを適切に組み合わせた TESTJP というデータベースを作成します。

```
db2 "call rdsadmin.create_database('TESTJP', 4096, 'IBM-437', 'JP', 'SYSTEM')"
```

rdsadmin.deactivate_database

データベースを非アクティブ化します。

構文

```
db2 "call rdsadmin.deactivate_database(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

非アクティブ化するデータベースの名前。データ型は varchar です。

使用に関する注意事項

データベースを非アクティブ化してメモリリソースを節約したり、他のデータベース設定を変更したりできます。非アクティブ化されたデータベースをオンラインに戻すには、rdsadmin.activate_database ストアドプロシージャを呼び出します。

rdsadmin.deactivate_database ストアドプロシージャを呼び出して、レプリケーション中にソース DB インスタンスのデータベースを非アクティブ化することはできません。

データベースの非アクティブ化ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.deactivate_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、TESTDB という名前のデータベースを非アクティブ化します。

```
db2 "call rdsadmin.deactivate_database(?, 'TESTDB')"
```

rdsadmin.activate_database

データベースをアクティブ化します。

rdsadmin.reactivate_database と rdsadmin.activate_database との違いについては、「使用に関する注意事項」を参照してください。

構文

```
db2 "call rdsadmin.activate_database(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

非アクティブ化するデータベースの名前。データ型は varchar です。

使用に関する注意事項

すべてのデータベースは、作成時にデフォルトでアクティブ化されます。スタンドアロン DB インスタンスのデータベースを非アクティブ化してメモリリソースを節約したり、他のデータベース設定を変更したりする場合は、rdsadmin.activate_database ストアドプロシージャを呼び出してデータベースを再度アクティブ化します。

このストアドプロシージャは、スタンドアロン DB インスタンスにあり、rdsadmin.deactivate_database ストアドプロシージャを呼び出して非アクティブ化されたデータベースのみをアクティブ化します。レプリカソース DB インスタンスでデータベースをアクティブ化するには、rdsadmin.reactivate_database ストアドプロシージャを呼び出す必要があります。

データベースのアクティブ化ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.activate_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、TESTDB という名前のデータベースをアクティブ化します。

```
db2 "call rdsadmin.activate_database(?, 'TESTDB')"
```

rdsadmin.reactivate_database

データベースを再アクティブ化します。

rdsadmin.activate_database と rdsadmin.reactivate_database との違いについては、「使用に関する注意事項」を参照してください。

構文

```
db2 "call rdsadmin.reactivate_database(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

再アクティブ化するデータベースの名前。データ型は varchar です。

使用に関する注意事項

rdsadmin.reactivate_database ストアドプロシージャを呼び出すと、このストアドプロシージャはまず rdsadmin.deactivate_database ストアドプロシージャを呼び出してデータベースを非ア

クティブ化し、次に rdsadmin.activate_database ストアドプロシージャを呼び出してデータベースをアクティブ化します。

データベース設定を変更した後、RDS for Db2 DB インスタンスのデータベースを再アクティブ化する必要が生じる場合があります。データベースを再アクティブ化する必要があるかどうかを判断するには、データベースに接続して db2 get db cfg show detail を実行します。

スタンドアロン DB インスタンスのデータベースの場合、rdsadmin.reactivate_database ストアプロシージャを使用してデータベースを再アクティブ化できます。または、rdsadmin.deactivate_database ストアドプロシージャを既に呼び出している場合は、rdsadmin.activate_database ストアドプロシージャを代わりに呼び出すことができます。

レプリカソース DB インスタンスのデータベースの場合、rdsadmin.reactivate_database ストアプロシージャを使用してデータベースを再アクティブ化する必要があります。

データベースの再アクティブ化ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.reactivate_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、TESTDB という名前のデータベースを再アクティブ化します。

```
db2 "call rdsadmin.reactivate_database(?, 'TESTDB')"
```

rdsadmin.drop_database

データベースを削除します。

構文

```
db2 "call rdsadmin.drop_database('database_name')"
```

パラメータ

以下のパラメータは必須です。

database_name

削除するデータベースの名前。データ型は varchar です。

使用に関する注意事項

次の条件が満たされた場合にのみ、rdsadmin.drop_database を呼び出してデータベースを削除できます。

- Amazon RDS コンソールまたは AWS CLI を使用して RDS for Db2 DB インスタンスを作成したときにデータベースの名前を指定しなかった。詳細については、「DB インスタンスの作成」を参照してください。
- the section called “rdsadmin.create_database” ストアドプロシージャを呼び出してデータベースを作成した。
- the section called “rdsadmin.restore_database” ストアドプロシージャを呼び出して、オフラインまたはバックアップされたイメージからデータベースを復元した。

rdsadmin.drop_database を呼び出す前に、rdsadmin データベースに接続する必要があります。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンス情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

データベースの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.drop_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、TESTDB という名前のデータベースを削除します。

```
db2 "call rdsadmin.drop_database('TESTDB')"
```

rdsadmin.update_db_param

データベースパラメータを更新します。

構文

```
db2 "call rdsadmin.update_db_param(
    'database_name',
```

```
    'parameter_to_modify',
    'changed_value',
    'restart_database')"
```

パラメータ

以下のパラメータは必須です。

database_name

タスクの実行対象のデータベース名。データ型は varchar です。

parameter_to_modify

変更するパラメータの名前。データ型は varchar です。詳細については、「Amazon RDS for Db2 パラメータ」を参照してください。

changed_value

パラメータの値を変更した後の値。データ型は varchar です。

次のパラメータはオプションです。

restart_database

再起動が必要な場合に RDS がデータベースを再起動するかどうかを指定します。データ型は varchar です。logprimary および logfilsiz を変更するには、このパラメータを 'YES' に設定します。

使用に関する注意事項

データベースパラメータの更新ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.update_db_param を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

例 1: パラメータを更新する

次の例では、TESTDB というデータベースで archretrydelay パラメータを 100 に更新します。

```
db2 "call rdsadmin.update_db_param(
    'TESTDB',
    'archretrydelay',
    '100')"
```

例 2: オブジェクトの検証を回避する

次の例では、依存関係のチェックを回避するために、TESTDB というデータベースにおいて作成したオブジェクトの検証を延期します。

```
db2 "call rdsadmin.update_db_param(
    'TESTDB',
    'auto_reval',
    'deferred_force')"
```

rdsadmin.set_configuration

データベースの特定の設定を構成します。

構文

```
db2 "call rdsadmin.set_configuration(
    'name',
    'value')"
```

パラメータ

以下のパラメータは必須です。

name

構成設定の名前。データ型は varchar です。

#

構成設定の値。データ型は varchar です。

使用に関する注意事項

次の表は、rdsadmin.set_configuration で制御できる構成設定を示しています。

名前	説明
ARCHIVE_LOG_COPY_TARGET_S3_ARN	アーカイブログがコピーされる S3 バケット ARN を指定します。この設定は、the section called “Amazon S3 へのアーカイブログのコピー” が有効になっている DB インスタンス上のすべてのデータベースに適用されます。
RDS_PASSWORD_MAX_LENGTH	パスワードの最大長を指定します。値は 8 ~ 128 文字で指定します。デフォルト値は 128 です。
RDS_PASSWORD_MIN_LENGTH	パスワードの最小長を指定します。値は 8 ~ 128 文字で指定します。デフォルト値は 8 です。
RDS_PASSWORD_MIN_NUMERIC	パスワードに必要な数字の最小数を指定します。0 ~ 128 の範囲の値にする必要があります。デフォルト値は 0 です。
RDS_PASSWORD_MIN_UPPER_CASE_CHAR	パスワードに必要な大文字の最小数を指定します。0 ~ 128 の範囲の値にする必要があります。デフォルト値は 0 です。
RDS_PASSWORD_MIN_LOWER_CASE_CHAR	パスワードに必要な小文字の最小数を指定します。0 ~ 128 の範囲の値にする必要があります。デフォルト値は 0 です。
RDS_PASSWORD_RULES	パスワードルールの設定を指定します。有効な値は DEFAULT または CUSTOM です。
RESTORE_DATABASE_NUM_BUFFERS	復元オペレーション中に作成するバッファの数。この値は、DB インスタンスクラスの合計メモリサイズよりも小さくする必要があります。この設定を構成しないと、Db2 は復元オペレーション中に使用する値を決定します。詳細については、「IBM Db2 ドキュメント」の「RESTORE DATABASE コマンド」を参照してください。
RESTORE_DATABASE_NUM_MULTI_PATHS	Amazon S3 オペレーションからの復元中に使用するパス (または I/O ストリーム) の数。この設定を使用するには、複数のバックアップファイルが必要です。この値は、複数のデータベースバックアップファイルを並行して復元するため、大量のデータを含むデータベースを復元する際のパフォーマンスを向上させることができます。データベースバックアップファイルの数と一致するようにこの値を設定することをお勧めしま

名前	説明
	す。詳細については、「IBM Db2 ドキュメント」の「BACKUP DATABASE command」を参照してください。
RESTORE_DATABASE_PARALLELISM	復元オペレーション中に作成するバッファマニピュレータの数。この値は、DB インスタンスの vCPU 数の 2 倍未満にする必要があります。この設定を構成しないと、Db2 は復元オペレーション中に使用する値を決定します。詳細については、「IBM Db2 ドキュメント」の「RESTORE DATABASE コマンド」を参照してください。
USE_STREAMING_RESTORE	最初にバックアップ全体を RDS for Db2 DB インスタンスにダウンロードしてから抽出するのではなく、復元中にバックアップデータを直接ストリーミングするかどうかを指定します。USE_STREAMING_RESTORE を TRUE に設定すると、ストレージ要件が大幅に削減され、復元パフォーマンスが向上します。この設定には、IBM Db2 エンジンバージョン 11.5.9.0.sb00063198.r1 以降と、データベースの Elastic Network Interface (ENI) を介した Amazon S3 接続が必要です。詳細については、「IBM Db2 ドキュメント」の「Remote storage」を参照してください。

> **Note**
>
> パスワード設定パラメータには、次の動作が適用されます。
>
> - パスワードパラメータが設定されると、RDS for Db2 は自動的に RDS_PASSWORD_RULES を CUSTOM に設定します。
> - RDS_PASSWORD_RULES を DEFAULT に設定すると、RDS for Db2 はすべてのパスワードパラメータをデフォルトにリセットします。
> - パスワードパラメータを NULL に設定することはできません。すべてのパスワードパラメータをデフォルトにリセットするには、set_configuration('RDS_PASSWORD_RULES', 'DEFAULT') を使用します。

例

例 1: 作成するバッファの数を指定する

次の例では、RESTORE_DATABASE_NUM_BUFFERS を 150 に設定します。

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_NUM_BUFFERS',
    '150')"
```

例 2: 作成するバッファマニピュレータの数を指定する

次の例では、RESTORE_DATABASE_PARALLELISM を 8 に設定します。

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_PARALLELISM',
    '8')"
```

例 3: 復元時に使用するパスまたは I/O ストリームの数を指定する

次の例では、RESTORE_DATABASE_NUM_MULTI_PATHS を 5 に設定します。

```
db2 "call rdsadmin.set_configuration(
    'RESTORE_DATABASE_NUM_MULTI_PATHS',
    '5')"
```

例 4: バックアップデータをストリーミングするための復元の設定

次の例では、USE_STREAMING_RESTORE を TRUE に設定します。

```
db2 "call rdsadmin.set_configuration(
    'USE_STREAMING_RESTORE',
    'TRUE')"
```

rdsadmin.show_configuration

ストアドプロシージャ rdsadmin.set_configuration を使用して指定できる現在の設定を返します。

構文

```
db2 "call rdsadmin.show_configuration(
```

```
    'name')"
```

パラメータ

次のパラメータはオプションです。

name

情報を返す対象の構成設定の名前。データ型は varchar です。

有効な設定名は次のとおりです。

- RESTORE_DATABASE_NUM_BUFFERS – 復元オペレーション中に作成するバッファの数。
- RESTORE_DATABASE_PARALLELISM – 復元オペレーション中に作成するバッファマニピュレータの数。
- RESTORE_DATABASE_NUM_MULTI_PATHS – Amazon S3 オペレーションからの復元中に使用するパス (または I/O ストリーム) の数。
- USE_STREAMING_RESTORE – 最初にバックアップデータ全体を RDS for Db2 DB インスタンスにダウンロードしてから抽出するのではなく、復元中にバックアップデータを直接ストリーミングするかどうかを指定します。

使用に関する注意事項

構成設定の名前を指定しないと、rdsadmin.show_configuration は、ストアドプロシージャ rdsadmin.set_configuration を使用して設定できるすべての構成設定に関する情報を返します。

例

次の例では、現在の RESTORE_DATABASE_PARALLELISM 設定に関する情報を返します。

```
db2 "call rdsadmin.show_configuration(
    'RESTORE_DATABASE_PARALLELISM')"
```

rdsadmin.backup_database

RDS for Db2 DB インスタンスから Amazon S3 バケットにデータベースをバックアップします。

構文

```
db2 "call rdsadmin.backup_database(
```

```
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    'backup_type',
    'compression_option',
    'util_impact_priority',
    'num_files',
    'parallelism',
    'num_buffers')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

バックアップする RDS for Db2 DB インスタンスのターゲットデータベースの名前。データ型は varchar です。

データベースは DB インスタンスに存在する必要があります。rdsadmin データベースをバックアップすることはできません。

s3_bucket_name

バックアップを保存する Amazon S3 バケットの名前。データ型は varchar です。

S3 バケットは、rdsadmin.backup_database を呼び出す前に存在し、バックアップする RDS for Db2 DB インスタンスのターゲットデータベースと同じ AWS リージョンにあり、DB インスタンスにアタッチされた IAM ロールを介してアクセスできる必要があります。

s3_prefix

RDS for Db2 がバックアップファイルをアップロードする Amazon S3 へのパスのプレフィックス。データ型は varchar です。

プレフィックスは 1024 文字に限定されます。先頭または末尾にスラッシュ (/) を含めることはできません。IBM から Amazon S3 へのストリーミングには制限があるため、プレフィックスにサブディレクトリを含めることをお勧めします。

ファイル管理を向上させるために、RDS for Db2 は *s3_prefix* の後に追加のディレクトリを作成します。RDS for Db2 は、すべてのバックアップファイルを *s3_prefix*/*dbi_resource_id*/*db_name* にアップロードします。*num_files* を 1 より大きく設定した場合、*db_name* ディレクトリには複数のバックアップファイルが含まれます。

バックアップファイルの Amazon S3 の場所の例を次に示します。この例では、backups/daily は *s3_prefix* パラメータに設定された値です。

```
backups/daily/db-5N7FXOY4GDP7RG2NSH2ZTAI2W4/SAMPLEDB
```

backup_type

バックアップ中にデータベースを使用できるかどうかを決定するバックアップのタイプ。データ型は varchar です。

有効な値:

- OFFLINE – バックアップ中はデータベースを使用できません。このタイプは高速ですが、ダウンタイムが発生します。
- ONLINE – データベースはバックアップ中も引き続き使用できます。デフォルトで、ONLINE は INCLUDE LOGS に設定されています。

以下のパラメータはオプションです。

compression_option

バックアップ時間、CPU 使用率、ストレージコストに影響する圧縮アルゴリズムのタイプ。データ型は varchar です。デフォルトは NONE です。

有効な値:

- NONE – 最大ファイルサイズ、最小の CPU 使用率、最低のストレージコスト。
- STANDARD – 標準 Db2 圧縮。libdb2compr.so を使用します。
- ZLIB – 拡張された Db2 圧縮。libdb2zcompr.so を使用しますが、CPU を大量に消費し、ストレージコストが最も高くなります。

util_impact_priority

バックアップがシステムリソースに与える影響を制御する設定。データ型は integer です。有効な値: 1–100 (低から高)。デフォルトは 50 です。

値を小さくすると、バックアップがシステムリソースに与える影響は軽減されますが、データベースのバックアップにかかる時間が長くなる可能性があります。値を大きくすると、データベースのバックアップを迅速に完了できますが、他のオペレーションに影響する可能性があります。実際の影響は、システム全体の使用率と util_impact_lim 設定によって異なります。パラメータグループで util_impact_lim 設定を表示および変更できます。詳細については、「Amazon RDS for Db2 パラメータ」を参照してください。

num_files

Amazon S3 への並列アップロードストリームの数。データ型は integer です。有効値: 1 ~ 256。

このパラメータは、Amazon RDS が自動的に計算するデフォルトのバックアップパフォーマンスを確認した後にのみ設定することをお勧めします。値を大きくすると、特に高帯域幅接続では、大規模なバックアップのパフォーマンスが向上する可能性がありますが、特定の時点では、値を大きくするとパフォーマンスが低下します。また、使用可能なシステムリソースとネットワーク容量も考慮してください。

parallelism

バックアップユーティリティが並行して読み取ることができるテーブルスペースの数。データ型は integer です。有効値: 1 ~ 256。

このパラメータは、Db2 エンジンが最適な値として自動的に計算するバックアップパフォーマンスをデフォルトで確認した後にのみ設定することをお勧めします。このパラメータを設定すると、Amazon RDS は使用可能なプロセッサに対して検証し、処理能力が不十分な場合はバックアップリクエストを実行しません。

num_buffers

使用するバッファの数。データ型は integer です。有効値: 1 ~ 268435456。

このパラメータは、Amazon RDS がメモリに基づいて自動的に計算するデフォルトのバックアップパフォーマンスを確認した後にのみ設定することをお勧めします。このパラメータを設定すると、Amazon RDS は使用可能なメモリに対して検証し、使用可能なメモリが不足している場合はバックアップリクエストを実行しません。複数の場所にバックアップする場合 (num_files が 1 以上に設定されている)、バッファの数が多いほどパフォーマンスが向上する可能性がありま

す。compression_option を NONE に設定しない場合、num_buffers と parallelism を増やすことでパフォーマンスを向上させることができます。

使用に関する注意事項

このストアドプロシージャは、Amazon S3 統合を使用してデータベースのバックアップを Amazon S3 バケットに直接ストリーミングする非同期バックアップタスクを作成します。ローカルサーバーからでも、RDS for Db2 DB インスタンスからでもバックアップを作成し、Amazon S3 にストリーミングして、任意の場所に復元することができます。データベースを RDS for Db2 DB インスタンスに復元する方法については、「rdsadmin.restore_database」を参照してください。

ストアドプロシージャを呼び出す前に、以下の考慮事項を確認してください。

- 一度にバックアップできるデータベースは 1 つだけです。
- DB インスタンスでバックアップと復元を同時に実行することはできません。
- AWS KMS を使用した Amazon S3 サーバー側の暗号化 (SSE-KMS) はサポートされていません。S3 バケットが SSE-KMS に設定されている場合でも、S3 バケットにアップロードされたファイルは SSE-KMS 暗号化を使用しません。
- バックアップファイルを Amazon S3 にストリーミングするには、統合を既に設定しておく必要があります。詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。
- RDS for Db2 DB インスタンスから Amazon S3 と対話できるようにするには、プライベートサブネットが使用する VPC と Amazon S3 ゲートウェイエンドポイントが必要です。詳細については、「ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する」および「ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する」を参照してください。

rdsadmin.backup_database を呼び出す前に、rdsadmin データベースに接続する必要があります。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンス情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

データベースをバックアップしたら、必ず接続を終了してください。

```
terminate
```

データベースのバックアップのステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.backup_database を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

すべての例では、MYDB というデータベースを amzn-s3-demo-bucket という Amazon S3 バケットにバックアップし、*s3_prefix* を backups/daily に設定します。

例 #1: 使用率の中央値と単一のアップロードストリームでデータベースがオフラインで使用不可であることを指定する

次の例では、データベースはオフラインです。これは高速ですが、バックアップ中にデータベースが使用できないことを意味します。この例では、ファイルの圧縮を実行せず、システムリソースへの影響は中程度であり、Amazon S3 への単一のアップロードストリームを使用します。

```
db2 call "rdsadmin.backup_database(
    ?,
    'MYDB',
    'amzn-s3-demo-bucket',
    'backups/daily',
    'OFFLINE',
    'NONE',
    50,
    1)"
```

例 #2: 拡張圧縮、使用率の中央値、少数の並列アップロードストリームを使用して、データベースをオンラインで使用可能に指定する

次の例では、データベースはオンラインであり、バックアップ中に使用できます。この例では、拡張圧縮を実行するため、ファイルサイズは小さくなりますが、CPU を大量に消費します。システムリソースへの影響の中央値をわずかに上回り、Amazon S3 への 5 つのアップロードストリームを使用します。

```
db2 call "rdsadmin.backup_database(
    ?,
    'MYDB',
    'amzn-s3-demo-bucket',
    'backups/daily',
```

```
    'ONLINE',
    'ZLIB',
    60,
    5)"
```

例 #3: デフォルトとシステム計算を使用して、データベースがオフラインで使用できないことを指定する

次の例では、データベースはオフラインです。これは高速ですが、バックアップ中にデータベースが使用できないことを意味します。この例では、ファイルのデフォルトの圧縮を使用し、システムリソースに影響を及ぼします。また、RDS for Db2 では、Amazon S3 への並列アップロードストリームの数、並列で読み取るテーブルスペース、使用するバッファを計算できます。

```
db2 "call rdsadmin.backup_database(
    ?,
    'MYDB',
    'amzn-s3-demo-bucket',
    'backups/daily',
    'OFFLINE')"
```

例 #4: 圧縮なし、高使用率、カスタム計算で、データベースをオフラインで使用不可に指定する

次の例では、データベースはオフラインです。これは高速ですが、バックアップ中にデータベースが使用できないことを意味します。この例では、ファイルの圧縮は行わず、システムリソースに大きな影響を与え、Amazon S3 への 20 のアップロードストリームを使用します。並列に読み取るテーブルスペースの最大数を設定します。これにより、処理能力が不十分な場合にバックアップリクエストが失敗する可能性があります。また、使用するバッファの最大数も設定するため、メモリが不足している場合にバックアップリクエストが失敗する可能性があります。

```
db2 "call rdsadmin.backup_database(
    ?,
    'MYDB',
    'amzn-s3-demo-bucket',
    'backups/daily',
    'OFFLINE',
    'NONE',
    90,
    20,
    256,
    268435456)"
```

rdsadmin.restore_database

Amazon S3 バケットから RDS for Db2 DB インスタンスにデータベースを復元します。

構文

```
db2 "call rdsadmin.restore_database(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix',
    restore_timestamp,
    'backup_type')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

RDS for Db2 で復元するターゲットデータベースの名前。データ型は varchar です。

例えば、ソースデータベース名が TESTDB である場合、*database_name* を NEWDB に設定すると、Amazon RDS はソースデータベースとして NEWDB を復元します。

s3_bucket_name

バックアップが保存されている Amazon S3 バケットの名前。データ型は varchar です。

s3_prefix

ダウンロード中のファイルマッチングに使用するプレフィックス。データ型は varchar です。

このパラメータが空の場合、Amazon S3 バケット内のすべてのファイルが処理されます。プレフィックスの例は、次のとおりです。

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

restore_timestamp

データベースバックアップイメージのタイムスタンプ。データ型は varchar です。

タイムスタンプはバックアップファイル名に含まれます。例えば、 20230615010101 はファイル名 SAMPLE.0.rdsdb.DBPART000.20230615010101.001 のタイムスタンプです。

backup_type

バックアップのタイプ。データ型は varchar です。有効な値: OFFLINE、ONLINE。

ほぼゼロのダウンタイムで移行するには、ONLINE を使用します。詳細については、「Amazon RDS for Db2 のダウンタイムがほぼゼロの Linux から Linux への移行」を参照してください。

使用に関する注意事項

このストアドプロシージャを使用して、Db2 データベースを RDS for Db2 DB インスタンスに移行できます。詳細については、「AWS のサービスを使用して Db2 から Amazon RDS for Db2 にデータを移行する」を参照してください。このストアドプロシージャを使用して、同じデータベースの複数のコピーを異なるデータベース名で作成し、各コピーで同じ復元イメージを使用することもできます。

ストアドプロシージャを呼び出す前に、以下の考慮事項を確認してください。

- データベースを復元する前に、バックアップとディスク上の元の Db2 データベースよりも大きいストレージスペースを RDS for Db2 DB インスタンス用にプロビジョンする必要があります。USE_STREAMING_RESTORE を有効にした場合、バックアップを復元すると、Amazon RDS はバックアップファイルを S3 バケットから RDS for Db2 DB インスタンスに直接ストリーミングします。USE_STREAMING_RESTORE を有効にしない場合、バックアップサイズとディスク上の元の Db2 データベースサイズの合計以上のストレージスペースを RDS for Db2 DB インスタンス用にプロビジョニングする必要があります。詳細については、「Insufficient disk space」を参照してください。
- バックアップを復元すると、Amazon RDS は RDS for Db2 DB インスタンスのバックアップファイルを抽出します。各バックアップファイルは 5 TB 以下である必要があります。バックアップファイルが 5 TB を超える場合は、バックアップファイルを小さいファイルに分割する必要があります。
- rdsadmin.restore_database ストアドプロシージャを使用してすべてのファイルを復元するには、ファイル名でタイムスタンプの後にファイル番号のサフィックスを含めません。例えば、*s3_prefix* backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101 は、以下のファイルを復元します。

```
SAMPLE.0.rdsdb.DBPART000.20230615010101.001
SAMPLE.0.rdsdb.DBPART000.20230615010101.002
SAMPLE.0.rdsdb.DBPART000.20230615010101.003
SAMPLE.0.rdsdb.DBPART000.20230615010101.004
SAMPLE.0.rdsdb.DBPART000.20230615010101.005
```

- RDS for Db2 は、非自動ストレージをサポートしていません。詳細については、「Tablespaces not restored」を参照してください。
- RDS for Db2 は、フェンスされていないルーチンをサポートしていません。詳細については、「Non-fenced routines not allowed」を参照してください。
- データベースの復元オペレーションのパフォーマンスを向上させるには、RDS で使用するバッファの数、バッファマニピュレータの数、および複数のバックアップパスの数を設定できます。ストレージの使用量を最適化し、パフォーマンスを向上させるために、Amazon S3 から直接バックアップをストリーミングすることもできます。現在の設定を確認するには、the section called “rdsadmin.show_configuration” を使用します。設定を変更するには、the section called “rdsadmin.set_configuration” を使用します。

データベースをオンラインにして、データベースを復元した後に追加のトランザクションログを適用するには、「rdsadmin.rollforward_database」を参照してください。

データベースの復元状況の確認については、「rdsadmin.get_task_status」を参照してください。

`rdsadmin.restore_database` を呼び出すときに返されるエラーメッセージについては、「the section called “ストアドプロシージャエラー”」を参照してください。

例

次の例では、*`s3_prefix`* `backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101` を持つ 1 つ以上のファイルでオフラインバックアップを復元します。

```
db2 "call rdsadmin.restore_database(
    ?,
    'SAMPLE',
    'amzn-s3-demo-bucket',
    'backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101',
    20230615010101,
    'OFFLINE')"
```

rdsadmin.rollforward_database

rdsadmin.restore_database を呼び出して、データベースをオンラインにし、データベースを復元した後に追加のトランザクションログを適用します。

構文

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'database_name',
    's3_bucket_name',
    s3_prefix,
    'rollforward_to_option',
    'complete_rollforward')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

オペレーションの実行対象のデータベースの名前。データ型は varchar です。

s3_bucket_name

バックアップが保存されている Amazon S3 バケットの名前。データ型は varchar です。

s3_prefix

ダウンロード中のファイルマッチングに使用するプレフィックス。データ型は varchar です。

このパラメータが空の場合、S3 バケット内のすべてのファイルがダウンロードされます。プレフィックスの例は次のとおりです。

```
backupfolder/SAMPLE.0.rdsdb.DBPART000.20230615010101
```

次の入力パラメータは、オプションです。

rollforward_to_option

ロールフォワードするポイント。データ型は varchar です。有効な値: END_OF_LOGS、END_OF_BACKUP (タイムスタンプの形式は YYYY-MM-DD-HH.MM.SS)。デフォルトは END_OF_LOGS です。

complete_rollforward

ロールフォワードプロセスを完了するかどうかを指定します。データ型は varchar です。デフォルトは TRUE です。

TRUE の場合は、完了後、データベースはオンラインになり、アクセス可能になります。FALSE の場合は、データベースは ROLL-FORWARD PENDING 状態のままになります。

使用に関する注意事項

さまざまなシナリオにおいてオンプレミスで生成されたインクルードログを含むオンラインバックアップに rds.rollforward_database を使用できます。

シナリオ 1: データベースの復元、含まれているログのロールフォワード、データベースのオンライン化

rdsadmin.restore_database() が完了したら、例 1 の構文を使用して、トランザクションログを含むデータベースをオンラインにする。

シナリオ 2: データベースをオンラインにするが、含まれているログをロールフォワードしない。

rdsadmin.restore_database() が完了したら、例 2 の構文を使用して、トランザクションログなしでデータベースをオンラインにします。

シナリオ 3: バックアップに含まれるログをロールフォワードし、オンプレミスで生成された追加のトランザクションログを適用する

rdsadmin.restore_database() が完了したら、例 3 または例 4 の構文を使用して、データベースをオンラインにせずにログをロールフォワードします。

complete_rollforward を FALSE に設定すると、データベースは ROLL-FORWARD PENDING 状態になり、オフラインになります。データベースをオンラインにするには、rdsadmin.complete_rollforward を呼び出す必要があります。

データベースのロールフォワードステータスを確認する方法については、「rdsadmin.rollforward_status」を参照してください。

例

例 1: トランザクションログを含むデータベースをオンラインにする

次の例では、トランザクションログを使用してデータベースのオンラインバックアップにロールフォワードし、データベースをオンラインにします。

```
db2 "call rdsadmin.rollforward_database(
    ?,
    null,
    null,
    'END_OF_LOGS',
    'TRUE')"
```

例 2: トランザクションログなしでデータベースをオンラインにする

次の例では、トランザクションログなしでデータベースのオンラインバックアップにロールフォワードし、データベースをオンラインにします。

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    'amzn-s3-demo-bucket',
    'logsfolder/',
    'END_OF_BACKUP',
    'TRUE')"
```

例 3: トランザクションログを使用してデータベースをオンラインにしない

次の例では、トランザクションログを使用してデータベースのオンラインバックアップにロールフォワードします。ただし、データベースはオンラインにしません。

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    null,
    'onlinebackup/TESTDB',
    'END_OF_LOGS',
    'FALSE')"
```

例 4: 追加のトランザクションログを使用してデータベースをオンラインにしない

次の例では、追加のトランザクションログを使用してデータベースのオンラインバックアップにロールフォワードします。ただし、データベースはオンラインにしません。

```
db2 "call rdsadmin.rollforward_database(
    ?,
    'TESTDB',
    'amzn-s3-demo-bucket',
    'logsfolder/S0000155.LOG',
    'END_OF_LOGS',
    'FALSE')"
```

rdsadmin.rollforward_status

ROLLFORWARD DATABASE *database_name* QUERY STATUS の出力を返します。

構文

```
db2 "call rdsadmin.rollforward_status(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

オペレーションの実行対象のデータベースの名前。データ型は varchar です。

使用に関する注意事項

rdsadmin.rollforward_database を呼び出した後、rdsadmin.rollforward_status を呼び出して、データベースのロールフォワードのステータスを確認できます。

このストアドプロシージャのステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin.complete_rollforward

データベースを ROLL-FORWARD PENDING 状態からオンラインにします。

構文

```
db2 "call rdsadmin.complete_rollforward(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

オンラインにするデータベースの名前。データ型は varchar です。

使用に関する注意事項

complete_rollforward を FALSE に設定して rdsadmin.rollforward_database を呼び出すと、データベースは ROLL-FORWARD PENDING 状態になり、オフラインになります。ロールフォワードプロセスを完了し、データベースをオンラインにするには、rdsadmin.complete_rollforward を呼び出します。

ロールフォワードプロセスの完了のステータスを確認する方法については、「rdsadmin.rollforward_status」を参照してください。

例

次の例では、TESTDB データベースをオンラインにします。

```
db2 "call rdsadmin.complete_rollforward(
    ?,
    'TESTDB')"
```

rdsadmin.db2pd_command

RDS for Db2 データベースに関する情報を収集します。

構文

```
db2 "call rdsadmin.db2pd_command(
                'db2pd_cmd',
                's3_bucket_name',
                's3_prefix')"
```

パラメータ

次の入力パラメータが必要です。

db2pd_cmd

実行する db2pd コマンドの名前。データ型は varchar です。

パラメータはハイフンで始める必要があります。パラメータのリストについては、IBM Db2 ドキュメントの「db2pd - Db2 データベースのモニターおよびトラブルシューティングコマンド」を参照してください。

次のオプションはサポートされていません。

- -addnode
- -alldatabases
- -alldbp
- -alldbs
- -allmembers
- -alm_in_memory
- -cfinfo
- -cfpool
- -command

- -dbpartitionnum
- -debug
- -dump
- -file | -o
- -ha
- -interactive
- -member
- -pages

> **Note**
>
> -pages summary がサポートされています。

- -pdcollection
- -stack
- -totalmem

file サブオプションはサポートされていません。例えば、db2pd -db testdb -tcbstats file=tcbstat.out。

stacks オプションの使用はサポートされていません。例えば、db2pd -edus interval=5 top=10 stacks。

以下のパラメータはオプションです。

s3_bucket_name

出力ファイルをアップロードする Amazon S3 バケットの名前。データ型は varchar です。*s3_bucket_name* も *s3_prefix* も指定されていない場合、収集された情報は rdsadmin.get_task_status を呼び出すことで取得できます。

s3_prefix

RDS for Db2 が出力ファイルをアップロードする Amazon S3 へのパスのプレフィックス。データ型は varchar です。プレフィックスを指定する場合は、*s3_bucket_name* も指定する必要があります。*s3_bucket_name* が存在するのに省略すると、出力ファイルは指定されたバケットのルートディレクトリにアップロードされます。

使用に関する注意事項

このストアドプロシージャは、RDS for Db2 データベース のモニタリングとトラブルシューティングに役立つ情報を収集します。

ストアドプロシージャは、IBM db2pd ユーティリティを使用してさまざまなコマンドを実行します。db2pd ユーティリティには SYSADM 権限が必要です (RDS for Db2 マスターユーザーにはない権限です)。ただし、Amazon RDS ストアドプロシージャを使用すると、マスターユーザーはユーティリティを使用してさまざまなコマンドを実行できます。このユーティリティの詳細については、IBM Db2 ドキュメントの「db2pd - Db2 データベースのモニターおよびトラブルシューティングコマンド」を参照してください。

出力は最大 2 GB に制限されています。

ストアドプロシージャを呼び出す前に、以下の要件を確認してください。

- 出力を Amazon S3 にアップロードするには、事前に統合を設定しておく必要があります。詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。
- RDS for Db2 DB インスタンスから Amazon S3 と対話できるようにするには、プライベートサブネットが使用する VPC と Amazon S3 ゲートウェイエンドポイントが必要です。詳細については、「ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する」および「ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する」を参照してください。
- *`-everything`* および *`-repeat`* オプションを使用するには、有効な Amazon S3 バケットの送信先が必要です。
- *`-repeat`* オプションの場合: デフォルトの構文は `'[command] -repeat [interval_seconds] [count]'` です。間隔は 1 秒以上とし、実行数は 100 回を超えることはできません。合計実行時間 (`interval_seconds` × `count`) は 600 秒を超えることはできません。値を指定しない場合、デフォルトは間隔が 5 秒、回数が 10 回です。

データベースに関する情報の収集状況を確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: DB インスタンスのアップタイムを返す

次の例は、RDS for Db2 DB インスタンスのアップタイムを返します。

```
db2 "call rdsadmin.db2pd_command('-')"
```

例 2: データベースのアップタイムを返す

次の例は、TESTDB という名前のデータベースのアップタイムを返します。

```
db2 "call rdsadmin.db2pd_command('-db TESTDB -')"
```

例 3: DB インスタンスのメモリ使用量を返す

次の例は、RDS for Db2 DB インスタンスのメモリ使用量を返します。

```
db2 "call rdsadmin.db2pd_command('-dbptnmem')"
```

例 4: DB インスタンスとデータベースのメモリセットを返す

次の例は、RDS for Db2 DB インスタンスとデータベース TESTDB のメモリセットを返します。

```
db2 "call rdsadmin.db2pd_command('-inst -db TESTDB -memsets')"
```

例 5: DB インスタンスのメモリ使用量を繰り返し返す

次の例は、RDS for Db2 DB インスタンスのメモリ使用量を 10 秒間隔で 6 回返します。

```
db2 "call rdsadmin.db2pd_command(
                '-dbptnmem -repeat 10 6',
                'amzn-s3-demo-bucket',
                'db2pdResults')"
```

rdsadmin.force_application

RDS for Db2 データベースからアプリケーションを強制的に削除します。

構文

```
db2 "call rdsadmin.force_application(
    ?,
    'applications')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

########

RDS for Db2 データベースから強制的に削除するアプリケーション。データ型は varchar です。有効な値: ALL または *application_handle*。

複数のアプリケーションの名前はカンマで区切ります。例: 「*application_handle_1*, *application_handle_2*」。

使用に関する注意事項

このストアドプロシージャは、すべてのアプリケーションをデータベースから強制的に削除して、メンテナンスを実行できるようにします。

ストアドプロシージャは、IBM FORCE APPLICATION コマンドを使用します。FORCE APPLICATION コマンドには SYSADM、SYSMAINT、または SYSCTRL 権限が必要です (RDS for Db2 マスターユーザーにはない権限です)。ただし、Amazon RDS ストアドプロシージャを使用すると、マスターユーザーはコマンドを使用できます。詳細については、IBM Db2 ドキュメントの「FORCE APPLICATION コマンド」を参照してください。

データベースからのアプリケーションの強制的な削除に関する状態を確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: すべてのアプリケーションを指定する

次の例では、すべてのアプリケーションを RDS for Db2 データベースから強制的に削除します。

```
db2 "call rdsadmin.force_application(
    ?,
    'ALL')"
```

例 2: 複数のアプリケーションを指定する

次の例では、RDS for Db2 データベースからアプリケーションハンドル 9991、8891、1192 を強制的に削除します。

```
db2 "call rdsadmin.force_application(
    ?,
    '9991, 8891, 1192')"
```

rdsadmin.set_archive_log_retention

指定した RDS for Db2 データベースのアーカイブログファイルを保持する時間 (時間単位) を設定します。

構文

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'database_name',
    'archive_log_retention_hours')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

アーカイブログの保持を設定するデータベースの名前。データ型は `varchar` です。

archive_log_retention_hours

アーカイブログファイルを保持する時間数。データ型は `smallint` です。デフォルトは `0` で、最大は `840` (35 日間) です。

値が `0` の場合、Amazon RDS はアーカイブログファイルを保持しません。

使用に関する注意事項

デフォルトでは、RDS for Db2 はログを 5 分間保持します。変更データキャプチャ (CDC) の AWS DMS や IBM Q Replication などのレプリケーションツールを使用する場合は、これらのツールでログ保持を 5 分以上設定することをお勧めします。

the section called “rdsadmin.show_archive_log_retention” を呼び出すと、現在のアーカイブログ保持の設定を表示できます。

rdsadmin データベースに対してアーカイブログ保持の設定を構成することはできません。

例

例 1: 保持時間を設定する

次の例は、TESTDB というデータベースのアーカイブログ保持時間を 24 時間に設定しています。

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'TESTDB',
    '24')"
```

例 2: 保持時間を無効にする

次の例では、TESTDB というデータベースのアーカイブログ保持を無効にします。

```
db2 "call rdsadmin.set_archive_log_retention(
    ?,
    'TESTDB',
    '0')"
```

rdsadmin.show_archive_log_retention

指定したデータベースに関する現在のアーカイブログ保持の設定を返します。

構文

```
db2 "call rdsadmin.show_archive_log_retention(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

アーカイブログ保持の設定を表示するデータベースの名前。データ型は varchar です。

例

次の例は、TESTDB というデータベースのアーカイブログ保持の設定を示しています。

```
db2 "call rdsadmin.show_archive_log_retention(?,'TESTDB')"
```

rdsadmin.list_archive_log_information

アーカイブログファイルのサイズ、作成日時、指定したデータベースの個々のログファイルの名前などの詳細を返します。また、データベース内のログファイルで使用される合計ストレージ量も返します。

構文

```
db2 "call rdsadmin.list_archive_log_information(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

アーカイブログ情報を一覧表示するデータベースの名前。データ型は varchar です。

例

次の例では、TESTDB というデータベースのアーカイブログ情報を返します。

```
db2 "call rdsadmin.list_archive_log_information(
    ?,
    'TESTDB')"
```

rdsadmin.enable_archive_log_copy

RDS Db2 データベースアーカイブログの Amazon S3 へのコピーを有効にします。

構文

```
db2 "call rdsadmin.enable_archive_log_copy(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

Amazon S3 へのアーカイブログのコピーを有効にするデータベースの名前。データ型は varchar です。

例

次の例では、TESTDB というデータベースのアーカイブログのコピーを有効にします。

```
db2 "call rdsadmin.enable_archive_log_copy(
    ?,
    'TESTDB')"
```

rdsadmin.disable_archive_log_copy

RDS Db2 データベースアーカイブログの Amazon S3 へのコピーを無効にします。

構文

```
db2 "call rdsadmin.disable_archive_log_copy(
    ?,
    'database_name')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要です。

database_name

Amazon S3 へのアーカイブログのコピーを無効にするデータベースの名前。データ型は varchar です。

例

次の例では、TESTDB というデータベースのアーカイブログのコピーを無効にします。

```
db2 "call rdsadmin.disable_archive_log_copy(
    ?,
    'TESTDB')"
```

rdsadmin.fgac_command

きめ細かなアクセス制御 (FGAC) コマンドを実行します。

構文

```
db2 "call rdsadmin.fgac_command(
    ?,
    'database_name',
    'fgac_cmd')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

FGAC コマンドを実行するデータベースの名前。データ型は varchar です。

fgac_cmd

実行するきめ細かなアクセス制御コマンド。データ型は varchar です。

次のコマンドは有効です。

- ALTER MASK – 行と列のアクセス制御 (RCAC) の既存の列マスクを変更します。
- ALTER PERMISSION – RCAC の既存の行アクセス許可のプロパティを変更します。
- ALTER SECURITY POLICY – RCAC の既存のセキュリティポリシーを変更します。
- ALTER SECURITY LABEL – ラベルベースアクセス制御 (LBAC) の既存のセキュリティラベルのプロパティを変更します。
- ALTER TABLE – RCAC または LBAC コントロールの追加など、テーブル構造を変更します。
- CREATE MASK – RCAC の新しい列マスクを作成します。
- CREATE PERMISSION – RCAC の新しい行アクセス許可を作成します。
- CREATE SECURITY LABEL – LBAC の新しいセキュリティラベルを作成します。
- CREATE SECURITY POLICY – RCAC の新しいセキュリティポリシーを作成します。
- DROP MASK – 既存の列マスクを削除します。

- DROP PERMISSION – 既存の行アクセス許可を削除します。
- DROP SECURITY LABEL – LBAC からセキュリティラベルを削除します。
- DROP SECURITY POLICY – 既存の RCAC セキュリティポリシーを削除します。
- GRANT EXEMPTION ON RULE – ユーザーが特定の LBAC ルールをバイパスできるようにします。
- GRANT SECURITY LABEL – LBAC セキュリティラベルをユーザーに割り当てます。
- REVOKE EXEMPTION ON RULE – LBAC ルールからユーザーの免除を削除します。
- REVOKE SECURITY LABEL – ユーザーから LBAC セキュリティラベルを削除します。

使用に関する注意事項

このストアドプロシージャは、RDS for Db2 DB インスタンスのデータベース内のテーブルデータへの行または列レベルでのアクセスを制御します。RDS for Db2 は、データベースで 2 種類の FGAC をサポートしています。

- ラベルベースアクセス制御 (LBAC)
- 行と列のアクセス制御 (RCAC)

ストアドプロシージャを呼び出す前に、以下の考慮事項を確認してください。

- 一重引用符 (') をエスケープするには、追加の一重引用符を使用します。次の例は 'apple'、'banana' および 'fruit' をエスケープする方法を示しています。

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
    'CREATE SECURITY LABEL COMPONENT FRUITSET SET{''apple'',''banana''}')"
```

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
    'CREATE SECURITY LABEL COMPONENT FRUITTREE TREE(''fruit'' ROOT, ''apple'' UNDER
 ''fruit'', ''banana'' UNDER ''fruit'')')"
```

- 角括弧 ([ ]) をエスケープするには、バックスラッシュ (\) を使用します。次の例は、[''apple'',''banana''] をエスケープする方法を示しています。

```
db2 "call rdsadmin.fgac_command(
    ?, '
    testdb',
    'CREATE SECURITY LABEL COMPONENT FRUITARRAY ARRAY\[''apple'',''banana''\]')"
```

例

次の例はすべて、testdb というデータベースで FGAC コマンドを実行します。

例 1: **FRUITSET** という名前の新しいセキュリティラベルを作成する

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
    'CREATE SECURITY LABEL COMPONENT FRUITSET SET{''apple'',''banana''}')"
```

例 2: **EMP_ID** が 3 未満に設定されている場合に有効になる **EMP_ID** 列の新しいマスクを作成する

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
    'CREATE MASK id_MASK ON EMPLOYEE FOR COLUMN EMP_ID RETURN CASE WHEN (EMP_ID < 3)
 THEN EMP_ID ELSE NULL END ENABLE')"
```

例 3: **SESSION_USER** が **security_user** に設定されている場合に有効になる **DEPARTMENT** 列の新しいマスクを作成する

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
    'CREATE MASK DEPARTMENT_MASK ON EMPLOYEE FOR COLUMN DEPARTMENT RETURN CASE  WHEN
 SESSION_USER = ''security_user'' THEN DEPARTMENT ELSE NULL END ENABLE')"
```

例 4: **treelabel** という名前の新しいセキュリティラベルを作成する

```
db2 "call rdsadmin.fgac_command(
    ?,
    'testdb',
```

```
    'CREATE SECURITY LABEL COMPONENT treelabel  TREE(''COMPANY'' ROOT, ''HR'' UNDER
 ''COMPANY'', ''FINANCE'' UNDER ''COMPANY'', ''IT'' UNDER ''COMPANY'')')"
```

rdsadmin.db2support_command

RDS for Db2 データベースに関する診断情報を収集し、Amazon S3 バケットにアップロードします。

構文

```
db2 "call rdsadmin.db2support_command(
    ?,
    'database_name',
    's3_bucket_name',
    's3_prefix')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。このパラメータは、? のみを受け入れます。

次の入力パラメータが必要となります。

database_name

診断情報を収集するデータベースの名前。データ型は `varchar` です。

s3_bucket_name

診断情報をアップロードする Amazon S3 バケットの名前。データ型は `varchar` です。

s3_prefix

RDS for Db2 が診断情報ファイルをアップロードする Amazon S3 へのパスのプレフィックス。データ型は `varchar` です。

使用に関する注意事項

このストアドプロシージャは、RDS for Db2 データベースのトラブルシューティングに役立つ診断情報を収集し、その情報を Amazon S3 バケットにアップロードします。

ストアドプロシージャでは、IBM db2support ユーティリティを使用して診断データを収集します。ユーティリティの詳細については、IBM Db2 ドキュメントの「db2support - Problem analysis and environment collection tool command」を参照してください。

ストアドプロシージャを呼び出す前に、以下の考慮事項を確認してください。

- 診断情報ファイルを Amazon S3 にアップロードするには、統合を既に設定しておく必要があります。詳細については、「Amazon RDS for Db2 DB インスタンスと Amazon S3 の統合」を参照してください。
- RDS for Db2 DB インスタンスから Amazon S3 と対話できるようにするには、プライベートサブネットが使用する VPC と Amazon S3 ゲートウェイエンドポイントが必要です。詳細については、「ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する」および「ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する」を参照してください。

rdsadmin.db2support_command を呼び出す前に、rdsadmin データベースに接続する必要があります。次の例では、*master_username* と *master_password* を RDS for Db2 DB インスタンス情報に置き換えます。

```
db2 connect to rdsadmin user master_username using master_password
```

診断情報の収集ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: データベースの診断情報を収集する

次の例では、TESTDB というデータベースの診断情報を収集し、それをプレフィックス diagnostics/testdb を持つ amzn-s3-demo-bucket という Amazon S3 バケットにアップロードします。

```
db2 "call rdsadmin.db2support_command(
    ?,
    'TESTDB',
    'amzn-s3-demo-bucket',
    'diagnostics/testdb')"
```

例 2: 日付ベースのプレフィックスを使用して診断情報を収集する

次の例では、MYDB というデータベースの診断情報を収集し、日付ベースのプレフィックスを持つ amzn-s3-demo-bucket という Amazon S3 バケットにアップロードします。

```
db2 "call rdsadmin.db2support_command(
    ?,
    'MYDB',
    'amzn-s3-demo-bucket',
    'support/2024/01/15')"
```

RDS for Db2 のストレージアクセスのストアドプロシージャ

このトピックで説明した組み込みのストアドプロシージャでは、Amazon S3 を使用してデータを移行する RDS for Db2 データベースのストレージアクセスを管理します。詳細については、「the section called “Amazon S3 による移行”」を参照してください。

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.catalog_storage_access
- rdsadmin.uncatalog_storage_access

rdsadmin.catalog_storage_access

Db2 データファイルを使用して Amazon S3 バケットにアクセスするためのストレージエイリアスをカタログ化します。

構文

```
db2 "call rdsadmin.catalog_storage_access(
    ?,
    'alias',
    's3_bucket_name',
    'grantee_type',
    'grantee'
    )"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。データ型は `varchar` です。

次の入力パラメータが必要となります。

alias

Amazon S3 バケット内のリモートストレージにアクセスするためのエイリアス名。データ型は varchar です。

s3_bucket_name

データが常駐する Amazon S3 バケットの名前。データ型は varchar です。

grantee_type

承認を受け取る受領者のタイプ。データ型は varchar です。有効な値: USER、GROUP。

grantee

承認を受け取るユーザーまたはグループ。データ型は varchar です。

使用に関する注意事項

Amazon RDS には、RDS for Db2 DB インスタンスに追加した IAM ロールにカタログ化されたエイリアスが含まれます。DB インスタンスから IAM ロールを削除すると、Amazon RDS はエイリアスを削除します。詳細については、「the section called “Amazon S3 による移行”」を参照してください。

エイリアスのカタログ化のステータスの確認については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、SAMPLE というエイリアスを登録します。ユーザー jorge_souza には、amzn-s3-demo-bucket という名前の Amazon S3 バケットへのアクセス権が付与されます。

```
db2 "call rdsadmin.catalog_storage_access(
    ?,
    'SAMPLE',
    'amzn-s3-demo-bucket',
    'USER',
    'jorge_souza')"
```

rdsadmin.uncatalog_storage_access

ストレージアクセスエイリアスを削除します。

構文

```
db2 "call rdsadmin.uncatalog_storage_access(
    ?,
    'alias')"
```

パラメータ

次のアウトプットパラメータが必要です。

?

エラーメッセージを出力するパラメータマーカー。データ型は varchar です。

次の入力パラメータが必要です。

alias

削除するストレージエイリアスの名前。データ型は varchar です。

使用に関する注意事項

エイリアスの削除ステータスの確認については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、SAMPLE というエイリアスを削除します。このエイリアスは、関連付けられていた Amazon S3 バケットへのアクセスを提供しなくなりました。

```
db2 "call rdsadmin.uncatalog_storage_access(
    ?,
    'SAMPLE')"
```

RDS for Db2 のテーブルスペースのストアドプロシージャ

このトピックで説明した組み込みのストアドプロシージャは、Amazon RDS for Db2 データベースのテーブルスペースを管理します。これらのプロシージャを実行する場合、マスターユーザーはまず `rdsadmin` データベースに接続する必要があります。

これらのストアドプロシージャは、さまざまなタスクで使用されます。これはすべてを網羅したリストではありません。

- テーブルスペースの一般的なタスク
- パフォーマンスレポートの生成
- db2look を使用してデータベースメタデータをコピーする
- IBM Db2 Data Management Console 用のリポジトリデータベースの作成

構文、パラメータ、使用上の注意、例については、以下の組み込みストアドプロシージャを参照してください。

ストアドプロシージャ

- rdsadmin.create_tablespace
- rdsadmin.alter_tablespace
- rdsadmin.rename_tablespace
- rdsadmin.drop_tablespace

rdsadmin.create_tablespace

テーブルスペースを作成します。

構文

```
db2 "call rdsadmin.create_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_page_size,
    tablespace_initial_size,
    tablespace_increase_size,
    'tablespace_type',
    'tablespace_prefetch_size')"
```

パラメータ

以下のパラメータは必須です。

database_name

テーブルスペースを作成するデータベースの名前。データ型は varchar です。

tablespace_name

作成するテーブルスペースの名前。データ型は varchar です。

テーブルスペース名には以下の制限があります。

- このデータベース内の既存のテーブルスペース名と同じものは使用できません。
- 使用できるのは _$#@a-zA-Z0-9 の文字のみです。
- _ または $ で始めることはできません。
- SYS で始めることはできません。

以下のパラメータはオプションです。

buffer_pool_name

テーブルスペースを割り当てるバッファプールの名前。データ型は varchar です。デフォルトは空の文字列です。

> **Important**
>
> テーブルスペースに関連付けるには、同じページサイズのバッファプールが既に存在している必要があります。

tablespace_page_size

テーブルスペースのページサイズ (バイト単位)。データ型は integer です。有効な値は、4096、8192、16384、32768 です。デフォルトは、rdsadmin.create_database を呼び出してデータベースを作成した際に使用したページサイズです。

> **Important**
>
> Amazon RDS は、4 KiB、8 KiB、16 KiB のページでの書き込みの原子性をサポートしています。これとは対照的に、32 KiB のページでは、Torn Write、つまり一部のデータしか

ディスクに書き込まれないというリスクが発生します。32 KiB のページを使用している場合は、ポイントインタイムリカバリと自動バックアップを有効にすることをお勧めします。有効にしない場合、データの不整合があるページから復元できなくなるリスクが発生します。詳細については、「the section called “バックアップの概要”」および「the section called “ポイントインタイムリカバリ”」を参照してください。

tablespace_initial_size

テーブルスペースの初期サイズ (キロバイト (KB) 単位)。データ型は integer です。有効な値: 48 以上。デフォルトは null です。

値を設定しない場合、Db2 が適切な値を設定します。

> **Note**
>
> 一時的なテーブルスペースはシステムが管理するため、このパラメータは一時テーブルスペースには適用されません。

tablespace_increase_size

テーブルスペースがフルになったときにテーブルスペースを増やす割合。データ型は integer です。有効値: 1 ~ 100。デフォルトは null です。

値を設定しない場合、Db2 が適切な値を設定します。

> **Note**
>
> 一時的なテーブルスペースはシステムが管理するため、このパラメータは一時テーブルスペースには適用されません。

tablespace_type

テーブルスペースのタイプ。データ型は char です。有効な値: U (ユーザーデータの場合)、T (ユーザー一時データの場合)、または S (システム一時データの場合)。デフォルトは U です。

tablespace_prefetch_size

テーブルスペースのプリフェッチページサイズ。データ型は char です。有効な値: AUTOMATIC (大文字と小文字を区別しない)、または 32767 以下のゼロ以外の正の整数。

使用に関する注意事項

RDS for Db2 は常にデータ用の大規模なデータベースを作成します。

テーブルスペースの作成ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

例 1: テーブルスペースを作成してバッファプールを割り当てる

次の例では、SP8 という名前のテーブルスペースを作成し、TESTDB という名前のデータベースに BP8 という名前のバッファプールを割り当てます。このテーブルスペースでは、初期テーブルスペースページサイズは 4,096 バイト、初期テーブルスペースは 1,000 KB、テーブルサイズの増加は 50% に設定されています。

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    4096,
    1000,
    50)"
```

例 2: 一時テーブルスペースを作成してバッファプールを割り当てる

次の例では、SP8 という名前の一時テーブルスペースを作成します。TESTDB というデータベースに対して、サイズが 8 KiB の BP8 というバッファプールを割り当てます。

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    8192,
    NULL,
```

```
    NULL,
    'T')"
```

例 3: テーブルスペースを作成し、プリフェッチページサイズを割り当てる

次の例では、TESTDB というデータベースの SP8 というテーブルスペースを作成します。テーブルスペースの初期テーブルスペース増加サイズは 50、プリフェッチページサイズは 800 です。

```
db2 "call rdsadmin.create_tablespace(
    'TESTDB',
    'SP8',
    NULL,
    NULL,
    NULL,
    50,
    NULL,
    '800')"
```

rdsadmin.alter_tablespace

テーブルスペースを変更します。

構文

```
db2 "call rdsadmin.alter_tablespace(
    'database_name',
    'tablespace_name',
    'buffer_pool_name',
    tablespace_increase_size,
    'max_size',
    'reduce_max',
    'reduce_stop',
    'reduce_value',
    'lower_high_water',
    'lower_high_water_stop',
    'switch_online',
    'tablespace_prefetch_size')"
```

パラメータ

以下のパラメータは必須です。

database_name

テーブルスペースを使用するデータベースの名前。データ型は varchar です。

tablespace_name

変更するテーブルスペースの名前。データ型は varchar です。

以下のパラメータはオプションです。

buffer_pool_name

テーブルスペースを割り当てるバッファプールの名前。データ型は varchar です。デフォルトは空の文字列です。

> **Important**
>
> テーブルスペースに関連付けるには、同じページサイズのバッファプールが既に存在している必要があります。

tablespace_increase_size

テーブルスペースがフルになったときにテーブルスペースを増やす割合。データ型は integer です。有効値: 1 ~ 100。デフォルトは 0 です。

max_size

テーブルスペースの最大サイズ。データ型は varchar です。有効な値: *integer* K | M | G、または NONE。デフォルトは NONE です。

reduce_max

ハイウォーターマークを上限まで減らすかどうかを指定します。データ型は char です。デフォルトは N です。

reduce_stop

前の reduce_maxまたは reduce_value コマンドを中断するかどうかを指定します。データ型は char です。デフォルトは N です。

reduce_value

テーブルスペースのハイウォーターマークを減らす際の数または割合。データ型は varchar です。有効な値: *integer* K| M | G、または 1 ~ 100。デフォルトは N です。

lower_high_water

ALTER TABLESPACE LOWER HIGH WATER MARK コマンドを実行するかどうかを指定します。データ型は char です。デフォルトは N です。

lower_high_water_stop

ALTER TABLESPACE LOWER HIGH WATER MARK STOP コマンドを実行するかどうかを指定します。データ型は char です。デフォルトは N です。

switch_online

ALTER TABLESPACE SWITCH ONLINE コマンドを実行するかどうかを指定します。データ型は char です。デフォルトは N です。

tablespace_prefetch_size

テーブルスペースのプリフェッチページサイズ。データ型は char です。有効な値: AUTOMATIC (大文字と小文字を区別しない)、または 32767 以下のゼロ以外の正の整数。

> **Note**
>
> このパラメータは、buffer_pool_name、table_increase_size、max_size、switch_online でのみ機能します。reduce_max、reduce_stop、reduce_value、lower_high_water、lower_high_wate では機能しません。

使用に関する注意事項

ストアドプロシージャを呼び出す前に、以下の考慮事項を確認してください。

- rdsadmin.alter_tablespace ストアドプロシージャは、ユーザーの一時データに対してテーブルスペースの tablespace_type を T に設定した場合は機能しません。
- オプションパラメータ reduce_max、reduce_stop、reduce_value、lower_high_water、lower_high_water_stop、は、相互に排他的です。rdsadmin.alter_tablespace コマンドで、buffer_pool_name などの他のオプションパラメータと組み合わせることはできません。詳細については、「Statement not valid」を参照してください。

テーブルスペースの変更ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

ストアドプロシージャを呼び出すときに返されるエラーメッセージについては、「the section called "ストアドプロシージャエラー"」を参照してください。

例

例 1: ハイウォーターマークを下げる

次の例では、SP8 というテーブルスペースを変更し、BP8 というデータベースに TESTDB というバッファプールを割り当てて、ハイウォーターマークを下げます。

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
    'SP8',
    'BP8',
    NULL,
    NULL,
    'Y')"
```

例 2: ハイウォーターマークを減らす

次の例では、データベース TBSP_TEST の TESTDB というテーブルスペースで REDUCE MAX コマンドを実行します。

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    'Y')"
```

例 3: コマンドを中断して高いウォーターマークを減らす

次の例では、データベース TBSP_TEST の TESTDB というテーブルスペースで REDUCE STOP コマンドを実行します。

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
```

```
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    NULL,
    'Y')"
```

例 4: 既存のプリフェッチページサイズの変更

次の例では、TSBP_TEST というテーブルスペースに対して 64 コマンドを実行し、既存のプリフェッチページサイズを ALTER TABLESPACE SWITCH ONLINE に変更します。

```
db2 "call rdsadmin.alter_tablespace(
    'TESTDB',
    'TBSP_TEST',
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    ‘Y’,
    ‘64’)"
```

rdsadmin.rename_tablespace

テーブルスペース名を変更します。

構文

```
db2 "call rdsadmin.rename_tablespace(
    ?,
    'database_name',
    'source_tablespace_name',
    'target_tablespace_name')"
```

パラメータ

以下のパラメータは必須です。

?

エラーメッセージを出力するパラメータマーカー。このパラメーターが受け入れるのは、? のみです。

database_name

テーブルスペースが属するデータベースの名前。データ型は varchar です。

source_tablespace_name

変更するテーブルスペースの名前。データ型は varchar です。

target_tablespace_name

新しいテーブルスペースの名前。データ型は varchar です。

新しい名前には以下の制限があります。

- 既存のテーブルスペース名と同じものは使用できません。
- 使用できるのは _$#@a-zA-Z0-9 の文字のみです。
- _ または $ で始めることはできません。
- SYS で始めることはできません。

使用に関する注意事項

テーブルスペースの名前変更ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

rdsadmin データベースに属するテーブルスペースの名前を変更することはできません。

例

次の例は、TESTDB というデータベース内の SP8 という名前のテーブルスペースを SP9 に変更します。

```
db2 "call rdsadmin.rename_tablespace(
    ?,
    'TESTDB',
    'SP8',
    'SP9')"
```

rdsadmin.drop_tablespace

テーブルスペースを削除します。

構文

```
db2 "call rdsadmin.drop_tablespace(
    'database_name',
    'tablespace_name')"
```

パラメータ

以下のパラメータは必須です。

database_name

テーブルスペースが属するデータベースの名前。データ型は varchar です。

tablespace_name

削除するテーブルスペースの名前。データ型は varchar です。

使用に関する注意事項

テーブルスペースの削除ステータスを確認する方法については、「rdsadmin.get_task_status」を参照してください。

例

次の例では、TESTDB というデータベースの SP8 というテーブルスペースを削除します。

```
db2 "call rdsadmin.drop_tablespace(
    'TESTDB',
    'SP8')"
```

Amazon RDS for Db2 ユーザー定義関数リファレンス

Db2 エンジンを実行している Amazon RDS DB インスタンスで使用できるユーザー定義関数について説明します。

トピック

- rdsadmin.get_task_status
- rdsadmin.list_databases
- rdsadmin.list_modifiable_db_params

rdsadmin.get_task_status

タスクのステータスを返します。

構文

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(task_id,'database_name','task_type'))"
```

パラメータ

以下のパラメータはオプションです。パラメータを指定しない場合、ユーザー定義関数はすべてのデータベースのすべてのタスクのステータスを返します。Amazon RDS は、タスク履歴を 35 日間保持します。

task_id

実行中のタスクの ID。この ID は、タスクの実行時に返されます。デフォルト: 0。

database_name

タスクが実行されているデータベースの名前。

task_type

クエリするタスクのタイプ。有効な値:
ADD_GROUPS、ADD_USER、ALTER_BUFFERPOOL、ALTER_TABLESPACE、CHANGE_PASSWORD、COMP

使用に関する注意事項

rdsadmin.get_task_status ユーザー定義関数を使用して、Amazon RDS for Db2 の以下のタスクのステータスを確認できます。これはすべてを網羅したリストではありません。

- バッファプールの作成、変更、削除
- テーブルスペースの作成、変更、削除
- データベースの作成または削除
- Amazon S3 からのデータベースバックアップの復元
- Amazon S3 からのデータベースログのロールフォワード

例

次の例では、 rdsadmin.get_task_status が呼び出されたときに返される列を表示します。

```
db2 "describe select * from table(rdsadmin.get_task_status())"
```

次の例では、すべてのタスクのステータスを一覧表示します。

```
db2 "select task_id, task_type, database_name, lifecycle,
    varchar(bson_to_json(task_input_params), 500) as task_params,
    cast(task_output as varchar(500)) as task_output
    from table(rdsadmin.get_task_status(null,null,null))"
```

次の例では、特定のタスクのステータスを一覧表示します。

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(1,null,null))"
```

次の例では、特定のタスクとデータベースのステータスを一覧表示します。

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(2,'SAMPLE',null))"
```

次の例では、ADD_GROUPS タスクのステータスを一覧表示します。

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(null,null,'add_groups'))"
```

次の例では、特定のデータベースのすべてのタスクのステータスを一覧表示します。

```
db2 "select task_id, task_type, database_name,
    varchar(bson_to_json(task_input_params), 500) as task_params
    from table(rdsadmin.get_task_status(null,'testdb', null))"
```

次の例では、JSON 値を列として出力します。

```
db2 "select varchar(r.task_type,25) as task_type, varchar(r.lifecycle,10) as lifecycle,
 r.created_at, u.* from
    table(rdsadmin.get_task_status(null,null,'restore_db')) as r,
 json_table(r.task_input_params, 'strict $' columns(s3_prefix varchar(500)
    null on empty, s3_bucket_name varchar(500) null on empty) error on error ) as U"
```

応答

`rdsadmin.get_task_status` ユーザー定義関数は次の列を返します。

`TASK_ID`

タスクの ID。

`TASK_TYPE`

入力パラメータによって異なります。

- `ADD_GROUPS` – グループを追加します。
- `ADD_USER` – ユーザーを追加します。
- `ALTER_BUFFERPOOL` – バッファプールを変更します。
- `ALTER_TABLESPACE` – テーブルスペースを変更します。
- `CHANGE_PASSWORD` – ユーザーのパスワードを変更します。
- `COMPLETE_ROLLFORWARD` – `rdsadmin.rollforward_database` タスクを完了し、データベースをアクティブにします。
- `CREATE_BUFFERPOOL` – バッファプールを作成します。

- CREATE_DATABASE – データベースを作成します。
- CREATE_ROLE – ユーザーの Db2 ロールを作成します。
- CREATE_TABLESPACE – テーブルスペースを作成します。
- DROP_BUFFERPOOL – バッファプールを削除します。
- DROP_DATABASE – データベースを削除します。
- DROP_TABLESPACE – テーブルスペースを削除します。
- LIST_USERS – すべてのユーザーを一覧表示します。
- REMOVE_GROUPS – グループを削除します。
- REMOVE_USER – ユーザーを削除します。
- RESTORE_DB – データベースを完全に復元します。
- ROLLFORWARD_DB_LOG – データベースログに対して rdsadmin.rollforward_database タスクを実行します。
- ROLLFORWARD_STATUS – rdsadmin.rollforward_database タスクのステータスを返します。
- UPDATE_DB_PARAM – データパラメータを更新します。

DATABASE_NAME

タスクが関連付けられているデータベースの名前。

COMPLETED_WORK_BYTES

タスクによって復元されたバイト数。

DURATION_MINS

タスクを完了するのにかかる時間。

LIFECYCLE

タスクのステータス。想定されるステータス:

- CREATED – タスクが Amazon RDS に送信されると、Amazon RDS ではステータスを CREATED に設定します。
- IN_PROGRESS - タスクがスタートすると、Amazon RDS ではステータスを IN_PROGRESS に設定します。ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。
- SUCCESS - タスクが完了すると、Amazon RDS ではステータスを SUCCESS に設定します。

- ERROR – 復元タスクが失敗した場合、Amazon RDS ではステータスを ERROR に設定します。エラーの詳細については、「TASK_OUPUT」を参照してください。

CREATED_BY

コマンドを作成した authid。

CREATED_AT

タスクが作成された日時。

LAST_UPDATED_AT

タスクが最終更新された日時。

TASK_INPUT_PARAMS

パラメータはタスクタイプによって異なります。すべての入力パラメータは JSON オブジェクトとして表されます。例えば、RESTORE_DB タスクの JSON キーは次のとおりです。

- DBNAME
- RESTORE_TIMESTAMP
- S3_BUCKET_NAME
- S3_PREFIX

TASK_OUTPUT

タスクに関する追加情報。ネイティブ復元中ににエラーが発生した場合、この列にエラーに関する情報が含まれます。

レスポンスの例

次のレスポンスの例は、 TESTJP というデータベースが正常に作成されたことを示しています。詳細については、the section called “rdsadmin.create_database” ストアドプロシージャを参照してください。

```
`1 SUCCESS CREATE_DATABASE RDSDB 2023-10-24-18.32.44.962689 2023-10-24-18.34.50.038523
 1 TESTJP { "CODESET" : "IBM-437", "TERRITORY" : "JP", "COLLATION" : "SYSTEM",
 "AUTOCONFIGURE_CMD" : "", "PAGESIZE" : 4096 }
2023-10-24-18.33.30.079048 Task execution has started.

2023-10-24-18.34.50.038523 Task execution has completed successfully`.
```

次のレスポンスの例は、データベースの削除が失敗した理由を説明しています。詳細については、the section called “rdsadmin.drop_database” ストアドプロシージャを参照してください。

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -
 2023-10-10-16.33.30.098857 Task execution has started.
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.
Reason Dropping database created via rds CreateDBInstance api is not allowed.
Only database created using rdsadmin.create_database can be dropped
```

次のレスポンスの例は、データベースの正常な復元を示しています。詳細については、the section called “rdsadmin.restore_database” ストアドプロシージャを参照してください。

```
1 RESTORE_DB  SAMPLE  SUCCESS

{ "S3_BUCKET_NAME" : "amzn-s3-demo-bucket", "S3_PREFIX" :
 "SAMPLE.0.rdsdb3.DBPART000.20230413183211.001", "RESTORE_TIMESTAMP" :
 "20230413183211", "BACKUP_TYPE" : "offline" }




2023-11-06-18.31.03.115795 Task execution has started.
2023-11-06-18.31.04.300231 Preparing to download
2023-11-06-18.31.08.368827 Download complete. Starting Restore
2023-11-06-18.33.13.891356 Task Completed Successfully
```

rdsadmin.list_databases

RDS for Db2 DB インスタンスで実行されているすべてのデータベースのリストを返します。

構文

```
db2 "select * from table(rdsadmin.list_databases())"
```

使用に関する注意事項

このユーザー定義関数には、データベースの状態がアクティブか非アクティブかを指定しません。

リストにデータベースが表示されない場合は、the section called “rdsadmin.get_task_status” ユーザー定義関数を呼び出してエラーメッセージを探してください。

応答

rdsadmin.list_databases ユーザー定義関数は次の列を返します。

DATABASE_NAME

データベースの名前。

CREATE_TIME

データベースが作成された日時。

DATABASE_UNIQUE_ID

Db2 データベースを一意に識別するために RDS が作成した GUID。

ARCHIVE_LOG_RETENTION_HOUR

アーカイブログファイルを保持する時間数。

ARCHIVE_LOG_COPY

データベースの機能が ENABLED か DISABLED かを表示します。

ARCHIVE_LOG_LAST_UPLOAD_FILE

S3 にアップロードされた最後のアーカイブログを示します。

ARCHIVE_LOG_LAST_UPLOAD_FILE_TIME

ログファイルがアーカイブされた時刻を示します。

ARCHIVE_LOG_COPY_STATUS

アーカイブログのコピーのステータスを表示します。

UPLOADING: アーカイブログファイルが S3 にアップロードされていることを示します。

CONFIGURATION_ERROR : 注意が必要な設定の問題があることを示します。

詳細なエラーを表示するには、Db インスタンスの RDS イベントメッセージを参照してください。イベントメッセージは、「Amazon RDS イベントの表示」で表示できます。

レスポンスの例

次のレスポンスの例は、データベースのリストと、それらの作成時刻を示しています。rdsadmin は Amazon RDS が管理し、常に出力に表示されるデータベースです。

```
DATABASE_NAME   CREATE_TIME                DATABASE_UNIQUE_ID
           ARCHIVE_LOG_RETENTION_HOURS ARCHIVE_LOG_COPY ARCHIVE_LOG_LAST_UPLOAD_FILE
 ARCHIVE_LOG_LAST_UPLOAD_FILE_TIME ARCHIVE_LOG_COPY_STATUS
-------------- -------------------------
 ------------------------------------------------- -------------------------
 --------------- --------------------------- ---------------------------------
 ------------------------------
RDSADMIN        2026-01-06-02.03.42.569069 RDSADMIN
                                  0 DISABLED         -                              -
                             -
FOO             2026-01-06-02.13.42.885650 F0D81C7E-7213-4565-B376-4F33FCF420E3
                                     0 ENABLED          S0006536.LOG
 2026-01-28-19.15.10.000000        UPLOADING
CODEP           2026-01-14-19.42.42.508476 106EEF95-6E30-4FFF-85AE-B044352DF095
                                  0 DISABLED         -                              -
                             -
...
```

rdsadmin.list_modifiable_db_params

変更可能なすべてのデータベース設定パラメータのリストを返します。

構文

```
db2 "select * from table(rdsadmin.list_modifiable_db_params())"
```

使用に関する注意事項

このユーザー定義関数は、変更可能なデータベースパラメータの選択したリストを表示します。これらのパラメータは、ストアドプロシージャ the section called “rdsadmin.update_db_param” を使用して更新できます。

このリストに含まれていないデータベースパラメータは制限されており、変更できません。

応答

`rdsadmin.list_modifiable_db_params` ユーザー定義関数は次の列を返します。

`PARAM_NAME`

変更できるパラメータの名前。

DEFAULT_VALUE

データベース作成時のデフォルトのパラメータ値。

RESTART_REQUIRED

データベースのリサイクルが必要な場合、または必要でない場合。

Y = はい、データベースの再起動が必要です。

N = いいえ。データベースの再起動は必要ありません。

レスポンスの例

以下は、予想される出力のサンプル (一部抜粋) リストです。

```
PARAM_NAME              DEFAULT_VALUE RESTART_REQUIRED
----------------------- ------------- ----------------
ACT_SORTMEM_LIMIT       NONE          N
ARCHRETRYDELAY          20            N
AUTHN_CACHE_DURATION    3             N
AUTHN_CACHE_USERS       0             N
AUTO_CG_STATS           OFF           N
...
```

Amazon RDS for Db2 のトラブルシューティング

以降のコンテンツは、RDS for Db2 で発生する可能性のある問題のトラブルシューティングに役立ちます。

Amazon RDS の一般的な問題のトラブルシューティングの詳細については、「Amazon RDS のトラブルシューティング」を参照してください。

トピック

- データベース接続エラー
- ファイル I/O エラー
- ストアドプロシージャエラー

データベース接続エラー

次のエラーメッセージは、サーバーに十分なメモリがないため、データベースの接続に失敗したことを示しています。

```
SQL1643C The database manager failed to allocate shared memory because the
database manager instance memory limit has been reached.
```

DB インスタンスのメモリを増やしてから、データベースに再度接続してください。データベースのメモリ使用量と推奨事項については、「the section called “複数の Db2 データベース”」を参照してください。RDS for Db2 データベースのメモリを更新する方法については、「the section called “rdsadmin.update_db_param”」を参照してください。

ファイル I/O エラー

LOAD コマンドを使用したり、rdsadmin.restore_database ストアドプロシージャを呼び出したりすると、さまざまな理由によるファイル I/O エラーが発生することがあります。

例えば、次の LOAD コマンドを実行したとします。

```
db2 "call sysproc.admin_cmd('load from "DB2REMOTE://s3test//public/datapump/t6.del" of
 del lobs from "DB2REMOTE://s3test/public/datapump/" modified by lobsinfile MESSAGES ON
 SERVER insert INTO RDSDB.t6 nonrecoverable ')"
```

LOAD コマンドは次のメッセージを返します。

```
 Result set 1
 --------------

 ROWS_READ             ROWS_SKIPPED          ROWS_LOADED           ROWS_REJECTED
ROWS_DELETED          ROWS_COMMITTED        ROWS_PARTITIONED      NUM_AGENTINFO_ENTRIES
MSG_RETRIEVAL




MSG_REMOVAL
 -------------------- -------------------- -------------------- --------------------
-------------------- -------------------- -------------------- --------------------
--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
                     -                    -                    -                    -
                    -                    -                    -                    -
SELECT SQLCODE, MSG FROM TABLE(SYSPROC.ADMIN_GET_MSGS('1594987316_285548770')) AS MSG




                                                                                    CALL
SYSPROC.ADMIN_REMOVE_MSGS('1594987316_285548770')

 1 record(s) selected.

 Return Status = 0

SQL20397W  Routine "SYSPROC.ADMIN_CMD" execution has completed, but at least
one error, "SQL1652", was encountered during the execution. More information
is available.  SQLSTATE=01H52
```

エラーメッセージを表示するには、前のレスポンスで提案したように SQL コマンドを実行します。SELECT SQLCODE, MSG FROM TABLE(SYSPROC.ADMIN_GET_MSGS('1594987316_285548770')) AS MSG は次のメッセージを返します。

```
SQLCODE   MSG
```

```
---------
 ------------------------------------------------------------------------------------------------
SQL2025N  An I/O error occurred. Error code “438”. Media on which this error occurred:
 “DB2REMOTE://s3test//public/datapump/t6.del”

SQL3500W The utility is beginning the LOAD phase at time “07/05/2024 21:21:48.082954”

SQL1652N File I/O error occurred
```

Db2 診断ログには、次のようなログファイルが含まれています。

```
2024-07-05-21.20.09.440609+000 I1191321E864          LEVEL: Error
PID     : 2710                 TID : 139619509200640 PROC : db2sysc 0
INSTANCE: rdsdb                NODE : 000            DB   : NTP
APPHDL  : 0-12180              APPID: xxx.xx.x.xxx.xxxxx.xxxxxxxxxxxx
UOWID   : 5                    ACTID: 1
AUTHID  : ADMIN                HOSTNAME: ip-xx-xx-x-xx
EDUID   : 147                  EDUNAME: db2lmr 0
FUNCTION: DB2 UDB, oper system services, sqloS3Client_GetObjectInfo, probe:219
MESSAGE : ZRC=0x870F01B6=-2029059658=SQLO_FAILED
          "An unexpected error is encountered"
DATA #1 : String, 29 bytes
S3:HeadObject request failed.
DATA #2 : signed integer, 4 bytes
99
DATA #3 : String, 0 bytes
Object not dumped: Address: 0x00007EFC08A9AE38 Size: 0 Reason: Zero-length data
DATA #4 : String, 33 bytes
curlCode: 28, Timeout was reached
```

このファイル I/O エラーは、さまざまなシナリオによって発生する可能性があります。例えば、RDS for Db2 DB インスタンスの作成に使用したセキュリティグループの VPC に、Amazon S3 ゲートウェイエンドポイントがない可能性があります。このエンドポイントがない場合、RDS for Db2 は Amazon S3 にアクセスできません。RDS for Db2 DB インスタンスがプライベートサブネットに存在する場合は、Amazon S3 ゲートウェイエンドポイントが必要です。Amazon RDS サブネットグループを設定することで、DB インスタンスがプライベートとパブリックのどちらのサブネットを使用するかを指定できます。詳細については、「DB サブネットグループの使用」を参照してください。

トピック

- ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する

- ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する

ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する

RDS for Db2 DB インスタンスから Amazon S3 と対話するには、VPC を作成し、続いてプライベートサブネットの Amazon S3 ゲートウェイエンドポイントを作成します。

S3 の VPC ゲートウェイエンドポイントを作成するには

1. VPC を作成します。詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「VPC を作成する」を参照してください。
2. プライベートサブネットの Amazon S3 ゲートウェイエンドポイントを作成します。詳細については、「AWS PrivateLink ガイド」の「Gateway endpoints」を参照してください。

ステップ 2: Amazon S3 の VPC ゲートウェイエンドポイントが存在することを確認する

AWS マネジメントコンソールまたは AWS CLI を使用して、Amazon S3 ゲートウェイエンドポイントが正常に作成されたことを確認します。

コンソール

Amazon S3 ゲートウェイエンドポイントを確認するには

1. AWS マネジメントコンソール にサインインし、https://console.aws.amazon.com/vpc で Amazon VPC コンソールを開きます。
2. コンソールで VPC の AWS リージョンを選択します。
3. 作成した VPC を選択します。
4. [リソースマップ] タブの [ネットワーク接続] で、Amazon S3 ゲートウェイエンドポイントがリストに表示されていることを確認します。

AWS CLI

Amazon S3 ゲートウェイエンドポイントを確認するには、describe-vpc-endpoints コマンドを実行します。次の例では、*vpc_id* を VPC ID に、*region* を AWS リージョンに、*profile* をプロファイル名に置き換えます。

Linux、macOS、Unix の場合:

```
aws ec2 describe-vpc-endpoints \
    --filters "Name=vpc-id,Values=$vpc_id" \
    "Name=service-name,\
    Values=com.amazonaws.${region}.s3" \
    --region $region --profile=$profile \
    --query "VpcEndpoints[*].VpcEndpointId" --output text
```

Windows の場合:

```
aws ec2 describe-vpc-endpoints ^
    --filters "Name=vpc-id,Values=$vpc_id" ^
    "Name=service-name,^
    Values=com.amazonaws.${region}.s3" ^
    --region $region --profile=$profile ^
    --query "VpcEndpoints[*].VpcEndpointId" --output text
```

Amazon S3 ゲートウェイエンドポイントが存在する場合、このコマンドプロシージャによって、次の例のような出力が生成されます。

```
[
    "vpce-0ea810434ff0b97e4"
]
```

Amazon S3 ゲートウェイエンドポイントが存在しない場合、このコマンドプロシージャによって、次の例のような出力が生成されます。

```
[]
```

Amazon S3 ゲートウェイエンドポイントがリストに表示されない場合は、「ステップ 1: これにより、Amazon S3 用の VPC ゲートウェイエンドポイントを作成する」の手順を実行してください。

ストアドプロシージャエラー

このセクションでは、ストアドプロシージャを呼び出すときに返されるさまざまなエラーとその解決方法について説明します。

Category	ストアドプロシージャエラー
データベース	rdsadmin.activate_database エラー
データベース	rdsadmin.backup_database エラー
データベース	rdsadmin.create_database エラー
データベース	rdsadmin.deactivate_database エラー
データベース	rdsadmin.drop_database エラー
データベース	rdsadmin.reactivate_database エラー
データベース	rdsadmin.restore_database エラー
データベース	rdsadmin.update_db_param エラー
テーブルスペース	rdsadmin.alter_tablespace エラー

rdsadmin.activate_database エラー

the section called “rdsadmin.activate_database” ストアドプロシージャを呼び出すと、以下のエラーが発生する場合があります。

エラー	エラーメッセージ
Failed to allocate shared memory	`SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.`
Unable to activate because of running processes	`The database can’t be activated because it's in the process of being created or restored.`

共有メモリの割り当てに失敗した

次のエラーメッセージは、DB インスタンスに十分なメモリがないため、ストアドプロシージャがデータベースのアクティブ化に失敗したことを示しています。

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB インスタンスのメモリを増やしてから、rdsadmin.activate_database ストアドプロシージャを再度呼び出してください。データベースのメモリ使用量と推奨事項については、「the section called “複数の Db2 データベース”」を参照してください。

実行中のプロセスが原因でアクティブ化できない

次のエラーメッセージは、rdsadmin.create_database または rdsadmin.restore_database ストアドプロシージャが実行されているため、ストアドプロシージャがデータベースをアクティブ化できなかったことを示しています。

```
The database can’t be activated because it's in the process of being created or
 restored.
```

数分間待ってから、再度 rdsadmin.activate_database ストアドプロシージャを呼び出してください。

rdsadmin.alter_tablespace エラー

the section called “rdsadmin.alter_tablespace” ストアドプロシージャを呼び出すと、以下のエラーが発生する場合があります。

エラー	エラーメッセージ
Statement not valid	DB21034E The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned: SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP_TEST" due to reason "12"
tablespace_prefetch_size value not valid	Invalid tablespace_prefetch_size. Set value to AUTOMATIC or to a non-zero positive numerical value.

エラー	エラーメッセージ
tablespace_prefetch_size numerical value not valid	`Invalid tablespace_prefetch_size. The number of pages can't be greater than 32767.`
Parameter can't be used with tablespace_prefetch_size	`You can't use tablespace_prefetch_size with {parameter }.`
Tablespace change failed	`The change to tablespace { tablespace_name}  failed because you can only alter LARGE or REGULAR tablespaces.`

ステートメントが無効です

次のエラーメッセージは、ストアドプロシージャが相互に排他的なオプションパラメータと他のオプションパラメータを組み合わせたことを示しています。rdsadmin.alter_tablespace ストアドプロシージャのオプションパラメータ reduce_max, reduce_stop、reduce_value、lower_high_water、lower_high_water_stop、switch_online は、相互に排他的です。rdsadmin.alter_tablespace ストアドプロシージャで、これらを他のオプションパラメータ (buffer_pool_name など) と組み合わせることはできません。組み合わせると、rdsadmin.get_task_status ユーザー定義関数を呼び出したときに、このエラーメッセージが Db2 から返されます。

```
DB21034E The command was processed as an SQL statement because it was not a valid
 Command Line Processor command. During SQL processing it returned:
SQL1763N Invalid ALTER TABLESPACE statement for table space "TBSP_TEST" due to reason
 "12"
```

相互に排他的なオプションパラメータを他のオプションパラメータと組み合わせずに、rdsadmin.alter_tablespace ストアドプロシージャを再度呼び出します。次に、rdsadmin.get_task_status ユーザー定義関数を呼び出します。詳細については、「rdsadmin.alter_tablespace」および「rdsadmin.get_task_status」を参照してください。

tablespace_prefetch_size 値が無効です

次のエラーメッセージは、tablespace_prefetch_size を AUTOMATIC または正でない数値に設定していないことを示しています。例えば、testinput に設定しようとしました。

```
Invalid tablespace_prefetch_size. Set value to AUTOMATIC or to a non-zero positive
 numerical value.
```

rdsadmin.alter_tablespace ストアドプロシージャを再度呼び出し、tablespace_prefetch_size を AUTOMATIC または正でない数値に設定します。

tablespace_prefetch_size の数値が無効です

次のエラーメッセージは、tablespace_prefetch_size を 32767 より大きい数値に設定していることを示しています。

```
Invalid tablespace_prefetch_size. The number of pages can't be greater than 32767.
```

rdsadmin.alter_tablespace ストアドプロシージャを再度呼び出し、tablespace_prefetch_size を 32767 以下のゼロ以外の正の数値に設定します。

パラメータは tablespace_prefetch_size では使用できません

次のエラーメッセージは、互換性のないパラメータで tablespace_prefetch_size を使用しようとしたことを示しています。

```
You can't use tablespace_prefetch_size with {parameter}.
```

rdsadmin.alter_tablespace ストアドプロシージャを再度呼び出し、互換性のあるパラメータでのみ tablespace_prefetch_size を使用します。tablespace_prefetch_size で使用できるパラメータについては、「rdsadmin.alter_tablespace」を参照してください。

テーブルスペースの変更に失敗しました

次のエラーメッセージは、テーブルスペースを変更しようとしたことを示しています。

```
The change to tablespace {tablespace_name} failed because you can only alter LARGE or
 REGULAR tablespaces.
```

rdsadmin.backup_database エラー

the section called “rdsadmin.backup_database” ストアドプロシージャを呼び出すと、以下のエラーが発生する場合があります。

エラー	エラーメッセージ
Insufficient disk space	`Aborting task. Reason Backing up your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the rdsadmin.backup_database stored procedure.`
Internal error	`Caught exception during executing task id 104, Aborting task. Reason Internal Error`

ディスク容量の不足

次のエラーメッセージは、DB インスタンスにデータベースをバックアップするのに十分なディスク容量がないことを示しています。

```
Aborting task. Reason Backing up your database failed because of insufficient
 disk space. Increase the storage for your DB instance and rerun the
 rdsadmin.backup_database stored procedure.
```

データベースをリモートストレージにバックアップするときは、バックアップセッションと作業ファイルに十分な空きディスク容量があることを確認してください。各バックアップセッションは最大 5 GB のデータを処理しますが、トランザクションログ、一時ファイル、継続的なデータベースオペレーションには追加のスペースが必要です。

データベースサイズに基づいて、バックアップ用に次の空きディスク容量を用意することをお勧めします。

- 5 GB 未満のデータベースの場合 – データベースサイズ + 3 GB バッファ
- 5 GB 以上のデータベースの場合 – 10 GB 以上の空き容量

この空きディスク容量は、バックアップセッション処理、バックアップ中のトランザクションログの蓄積、一時作業ファイル、および設定されている場合は並列バックアップストリームを考慮したものです。詳細については、「DB インスタンスストレージの容量を増加する」を参照してください。

ディスク容量を増やしてから、rdsadmin.backup_database ストアドプロシージャを再度呼び出してください。データベースが正しくバックアップされたことを確認するには、rdsadmin.get_task_status を使用してタスクのステータスを確認します。*s3_prefix/*

dbi_resource_id/db_name の Amazon S3 バケットにバックアップファイルが存在することを確認することもできます。

内部エラー

次のエラーメッセージは、ストアドプロシージャで内部エラーが発生したことを示します。

```
Caught exception during executing task id 104, Aborting task. Reason Internal Error
```

AWS Support にお問い合わせください。

rdsadmin.create_database エラー

the section called “rdsadmin.create_database” ストアドプロシージャを呼び出すと、次のエラーが発生する場合があります。

エラー	エラーメッセージ
Failed to allocate shared memory	`SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.`

共有メモリの割り当てに失敗した

次のエラーメッセージは、DB インスタンスに十分なメモリがないため、ストアドプロシージャがデータベースの作成に失敗したことを示しています。

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB インスタンスのメモリを増やしてから、`rdsadmin.create_database` ストアドプロシージャを再度呼び出してください。データベースのメモリ使用量と推奨事項については、「the section called “複数の Db2 データベース”」を参照してください。

データベースが作成されたことを確認するには、the section called “rdsadmin.list_databases” ユーザー定義関数を呼び出し、新しいデータベースがリストに表示されていることを確認します。

rdsadmin.deactivate_database エラー

the section called “rdsadmin.deactivate_database” ストアドプロシージャを呼び出すと、次のエラーが発生する場合があります。

エラー	エラーメッセージ
Unable to deactivate because of running processes	The database can’t be deactivated because it's in the process of being created or restored.

実行中のプロセスが原因で非アクティブ化できない

次のエラーメッセージは、rdsadmin.create_database または rdsadmin.restore_database ストアドプロシージャが実行されているため、ストアドプロシージャがデータベースを非アクティブ化できなかったことを示しています。

```
The database can’t be deactivated because it's in the process of being created or
 restored.
```

数分間待ってから、再度 rdsadmin.deactivate_database ストアドプロシージャを呼び出してください。

rdsadmin.drop_database エラー

the section called “rdsadmin.drop_database” ストアドプロシージャを呼び出すと、以下のエラーが発生する場合があります。

エラー	エラーメッセージ
Database name doesn't exist	SQL0438N Application raised error or warning with diagnostic text: "Cannot drop database. Database with provided name does not exist". SQLSTATE=99993
Return status = 0	Return Status = 0
Dropping database not allowed	1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 - 2023-10-10-16.33.30.098857 Task execution has started.

エラー	エラーメッセージ
	2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task. Reason Dropping database created via rds CreateDBInstance api is not allowed. Only database created using rdsadmin.create_database can be dropped

データベース名が存在しません

次のエラーメッセージは、rdsadmin.drop_database ストアドプロシージャで誤ったデータベース名を渡したことを示しています。

```
SQL0438N Application raised error or warning with diagnostic text: "Cannot
drop database. Database with provided name does not exist". SQLSTATE=99993
```

正しいデータベース名を使用して rdsadmin.drop_database ストアドプロシージャを再度呼び出します。データベースが破棄されたことを確認するには、the section called “rdsadmin.list_databases” ユーザー定義関数を呼び出し、破棄されたデータベースがリストに表示されていないことを確認します。

戻りステータス = 0

次のエラーメッセージは、ストアドプロシージャを完了できなかったことを示しています。

```
Return Status = 0
```

Return Status = 0 を受け取ったら、rdsadmin.get_task_status ユーザー定義関数を呼び出します。

データベースの削除は許可されていません

次のエラーメッセージは、Amazon RDS コンソールまたは AWS CLI を使用してデータベースを作成したことを示しています。rdsadmin.drop_database ストアドプロシージャは、the section called “rdsadmin.create_database” ストアドプロシージャを呼び出してデータベースを作成した場合のみ使用できます。

```
1 ERROR DROP_DATABASE RDSDB 2023-10-10-16.33.03.744122 2023-10-10-16.33.30.143797 -
 2023-10-10-16.33.30.098857 Task execution has started.
```

```
2023-10-10-16.33.30.143797 Caught exception during executing task id 1, Aborting task.
Reason Dropping database created via rds CreateDBInstance api is not allowed.
Only database created using rdsadmin.create_database can be dropped
```

Amazon RDS コンソールまたは AWS CLI を使用して作成したデータベースを削除するには、クライアントを使用してデータベースに接続し、適切なコマンドを実行します。

rdsadmin.reactivate_database エラー

the section called “rdsadmin.reactivate_database” ストアドプロシージャを呼び出すと、次のエラーが発生する場合があります。

エラー	エラーメッセージ
Failed to allocate shared memory	SQL1643C The database manager failed to allocate shared memory because the database manager instance memory limit has been reached.
Unable to reactivate because of running processes	The database can’t be reactivated because it's in the process of being created or restored.

共有メモリの割り当てに失敗した

次のエラーメッセージは、DB インスタンスに十分なメモリがないため、ストアドプロシージャがデータベースのアクティブ化に失敗したことを示しています。

```
SQL1643C The database manager failed to allocate shared memory because the database
 manager instance memory limit has been reached.
```

DB インスタンスのメモリを増やしてから、rdsadmin.activate_database ストアドプロシージャを再度呼び出してください。データベースのメモリ使用量と推奨事項については、「the section called “複数の Db2 データベース”」を参照してください。

実行中のプロセスが原因で再アクティブ化できない

次のエラーメッセージは、rdsadmin.create_database または rdsadmin.restore_database ストアドプロシージャが実行されているため、ストアドプロシージャがデータベースを再アクティブ化できなかったことを示しています。

```
The database can't be reactivated because it's in the process of being created or
 restored.
```

数分間待ってから、再度 rdsadmin.reactivate_database ストアドプロシージャを呼び出してください。

rdsadmin.restore_database エラー

the section called "rdsadmin.restore_database" ストアドプロシージャを呼び出すと、次のエラーが発生する可能性があります。

エラー	エラーメッセージ
Insufficient disk space	Aborting task. Reason Restoring your database failed because of insufficient disk space. Increase the storage for your DB instance and rerun the rdsadmin.restore_database stored procedure.
Internal error	Caught exception during executing task id 104, Aborting task. Reason Internal Error
Non-fenced routines not allowed	Caught exception during executing task id 2, Aborting task. Reason Non fenced routines are not allowed. Please delete the routines and retry the restore.
Tablespaces not restored	Reason SQL0970N The system attempted to write to a read-only file. Reason SQL2563W The Restore process has completed successfully. However one or more table spaces from the backup were not restored.

ディスク容量の不足

次のエラーメッセージは、DB インスタンスにデータベースを復元するのに十分なディスク容量がないことを示しています。

```
Aborting task. Reason Restoring your database failed because of insufficient
 disk space. Increase the storage for your DB instance and rerun the
 rdsadmin.restore_database stored procedure.
```

DB インスタンスの空き容量は、バックアップイメージの 2 倍以上である必要があります。バックアップイメージが圧縮されている場合、DB インスタンスの空き容量はバックアップイメージの 3 倍以上である必要があります。詳細については、「the section called “DB インスタンスストレージの容量を増加する”」を参照してください。

ディスク容量を増やしてから、rdsadmin.restore_database ストアドプロシージャを再度呼び出してください。データベースが復元されたことを確認するには、the section called “rdsadmin.list_databases” ユーザー定義関数を呼び出し、復元されたデータベースがリストに表示されていることを確認します。

内部エラー

次のエラーメッセージは、ストアドプロシージャで内部エラーが発生したことを示します。

```
Caught exception during executing task id 104, Aborting task. Reason Internal Error
```

AWS Support にお問い合わせください。

フェンスなしのルーチンは許可されません

次のエラーメッセージは、データベースにフェンスのないルーチンが含まれていることを示します。

```
Caught exception during executing task id 2, Aborting task. Reason Non fenced routines
 are not allowed. Please delete the routines and retry the restore.
```

RDS for Db2 は、フェンスされていないルーチンをサポートしていません。フェンスされていないルーチンをソースデータベースから削除し、もう一度 rdsadmin.restore_database を呼び出します。データベースが復元されたことを確認するには、the section called “rdsadmin.list_databases” ユーザー定義関数を呼び出し、復元されたデータベースがリストに表示されていることを確認します。詳細については、「the section called “フェンスされていないルーチン”」を参照してください。

テーブルスペースが復元されていません

次のエラーメッセージは、RDS for Db2 がデータベースを正常に復元したが、1 つ以上のテーブルスペースを復元できなかったことを示しています。

```
Reason SQL0970N The system attempted to write to a read-only file.
Reason SQL2563W The Restore process has completed successfully. However one or more
 table spaces from the backup were not restored.
```

RDS for Db2 は、非自動ストレージをサポートしていません。非自動ストレージを自動ストレージに変換してから、もう一度 rdsadmin.restore_database を呼び出します。詳細については、「IBM Db2 ドキュメント」の「非自動ストレージストレージデータベースを自動ストレージを使用するように変換する」を参照してください。

非自動 SMS ストレージのデータベースは、手動で復元する必要があります。データベースに非自動 SMS ストレージがある場合は、AWS サポートにお問い合わせください。

非自動ストレージと 1 回限りの移行の詳細については、「the section called “移行中の非自動ストレージテーブルスペース”」を参照してください。

rdsadmin.update_db_param エラー

the section called “rdsadmin.update_db_param” ストアドプロシージャを呼び出すと、次のエラーが発生する場合があります。

エラー	エラーメッセージ
Parameter not supported or modifiable	SQL0438N Application raised error or warning with diagnostic text: "Parameter is either not supported or not modifiable to customers". SQLSTATE=99993

パラメータはサポートされていないか、変更不可です

次のエラーメッセージは、変更しようとしたデータベース設定パラメータがサポートされていないか、変更不可であることを示しています。

```
SQL0438N Application raised error or warning with diagnostic text: "Parameter
is either not supported or not modifiable to customers". SQLSTATE=99993
```

パラメータグループを参照することで、変更可能なパラメータを確認できます。詳細については、「the section called “DB パラメータグループのパラメータ値を表示する”」を参照してください。

Amazon RDS for MariaDB

Amazon RDS は、DB インスタンス用に複数のバージョンの MariaDB をサポートしています。サポートされているバージョンの詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。

MariaDB DB を作成するには、Amazon RDS の管理ツールまたはインターフェイスを使用します。これで、Amazon RDS ツールを使用して DB インスタンスの管理アクションを実行できるようになります。これには、以下のようなアクションが含まれます。

- DB インスタンスの再設定またはサイズ変更
- DB インスタンスへの接続の許可
- バックアップまたはスナップショットからの作成と復元
- マルチ AZ セカンダリの作成
- リードレプリカの作成
- DB インスタンスのパフォーマンスのモニタリング

DB インスタンスでデータを保存したり、またはアクセスするには、標準の MariaDB のユーティリティとアプリケーションを使用します。

MariaDB がすべての AWS リージョン で利用可能になりました。AWS リージョン の詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

Amazon RDS for MariaDB 用の Amazon RDS を使用して、HIPAA 準拠のアプリケーションを構築できます。AWS との事業提携契約 (BAA) に基づいて、保護されるべき医療情報 (PHI) を含め、医療関連の情報を保存できます。詳細については、「HIPAA コンプライアンス」を参照してください。AWS対象範囲内のサービスは、サードパーティーの監査人によって十分に評価され、認定、コンプライアンスの証明、または Authority to Operate (ATO) が与えられます。詳細については、コンプライアンスプログラムによる対象範囲内の AWS のサービスを参照してください。

DB インスタンスを作成する前に、「Amazon RDS 環境のセットアップ」の手順を完了してください。DB インスタンスを作成すると、RDS マスターユーザーは DBA 権限を取得します。ただし、いくつかの制限があります。このアカウントは、追加のデータベースアカウントの作成などの管理タスクに使用します。

以下を作成することができます。

- DB インスタンス
- DB スナップショット
- ポイントインタイムの復元
- 自動バックアップ
- 手動バックアップ

Amazon VPC に基づいて Virtual Private Cloud (VPC) 内で MariaDB を実行する DB インスタンスを使用できます。また、さまざまなオプションを有効にして、MariaDB DB インスタンスに機能を追加することもできます。Amazon RDS は、可用性の高いフェイルオーバーソリューションとして MariaDB のマルチ AZ 配置をサポートしています。

> **Important**
>
> マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする、特定のシステムプロシージャやテーブルへのアクセスも制限しています。データベースへのアクセスには、mysql クライアントなどの標準の SQL クライアントアプリケーションを使用します。ただし、Telnet またはセキュアシェル (SSH) を使用してホストに直接アクセスすることはできません。

トピック

- Amazon RDS での MariaDB 機能のサポート
- Amazon RDS の MariaDB のバージョン
- MariaDB DB インスタンスへの接続
- MariaDB DB インスタンス接続の保護
- Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上
- Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上
- MariaDB DB エンジンのアップグレード
- MariaDB DB スナップショットエンジンバージョンのアップグレード
- Amazon RDS for MariaDB DB インスタンスへのデータのインポート
- Amazon RDS での MariaDB のレプリケーションの使用
- MariaDB データベースエンジンのオプション

- MariaDB のパラメータ
- MySQL DB スナップショットから MariaDB DB インスタンスへのデータ移行
- Amazon RDS SQL での MariaDB リファレンス
- MariaDB DB インスタンスのローカルタイムゾーン
- Amazon RDS for MySQL の既知の問題と制限

Amazon RDS での MariaDB 機能のサポート

RDS for MariaDB は MariaDB のほとんどの特徴と機能をサポートしています。一部の機能には、制限付きのサポートまたは制限された特権があります。

[What's New with Database?] (データベースの新機能) ページで新しい Amazon RDS 機能をフィルタリングできます。[製品] で [Amazon RDS] を選択します。その後、**MariaDB 2023** などのキーワードを使用して検索します。

> **Note**
>
> 以下のリストは完全なものではありません。

Amazon RDS での MariaDB 機能のサポートの詳細については、以下のトピックを参照してください。

トピック

- Amazon RDS の MariaDB でサポートされているストレージエンジン
- Amazon RDS での MariaDB のキャッシュウォームアップ
- Amazon RDS でサポートされていない MariaDB の機能

Amazon RDS for MariaDB のメジャーバージョンでの MariaDB 機能のサポート

次のセクションでは、Amazon RDS for MariaDB のメジャーバージョンでの MariaDB 機能のサポートに関する情報について説明しています。

トピック

- Amazon RDS での MariaDB 12.3 のサポート
- Amazon RDS での MariaDB 11.8 のサポート
- Amazon RDS での MariaDB 11.4 のサポート
- Amazon RDS での MariaDB 10.11 のサポート
- Amazon RDS での MariaDB 10.6 のサポート
- Amazon RDS での MariaDB 10.5 のサポート
- Amazon RDS での MariaDB 10.4 のサポート

Amazon RDS for MariaDB のサポートされているマイナーバージョンについては、「Amazon RDS の MariaDB のバージョン」を参照してください。

Amazon RDS での MariaDB 12.3 のサポート

Amazon RDS は、MariaDB バージョン 12.3 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

> **Note**
>
> MariaDB 12.3 は、Amazon RDS が MariaDB 11.8 以降にサポートする最初のメジャーバージョンです。MariaDB 11.8 からアップグレードすると、DB インスタンスも MariaDB バージョン 12.0、12.1、および 12.2 で導入された機能を受け取ります。

- RDS Proxy の予約済みユーザー – `rdsproxyadmin` ユーザーは MariaDB 12.3 以降の予約済みユーザーです。このユーザーを変更または削除することはできません。詳細については、「Amazon RDS での MariaDB のセキュリティ」を参照してください。
- あらゆるホスト値のレプリケーションユーザーアカウントの削除保護 – `rdsrepladmin` ユーザーアカウントは、ホスト値にかかわらず削除することはできません。以前は、この保護は `'rdsrepladmin'@'%'` にのみ適用されていました。詳細については、「Amazon RDS での MariaDB のセキュリティ」を参照してください。
- **`caching_sha2_password`** プラグインによる認証 – RDS for MySQL バージョン 8.4 のデフォルトの認証プラグインである `caching_sha2_password` プラグインを使用して認証するユーザーを作成できます。このプラグインは、`mysql_native_password` よりも強力なパスワードハッシュを提供します。
- セッション認可 – `SET SESSION AUTHORIZATION` ステートメントを使用して、新しい接続を開くことなく、別のユーザーとしてステートメントを実行できます。ユーザーがこのステートメン

トを使用するには、SET USER 権限が必要です。このステートメントは、トランザクション、準備済みステートメント、またはストアドルーチン内で使用することはできません。また、内部ユーザーの設定には使用できません。

- 廃止または削除されたパラメータ – innodb_ft_enable_diag_print パラメータは廃止され、MariaDB バージョン 12.3 DB インスタンスには効果がありません。このパラメータは、パラメータグループに引き続き存在します。詳細については、MariaDB ウェブサイトの「innodb_ft_enable_diag_print」を参照してください。
- パラメータの新しい有効な値 – MariaDB バージョン 12.3 の DB インスタンスでは、以下のパラメータに新しい有効な値が追加されました。
  - read_only パラメータの有効な値が 0 および 1 から OFF、ON、NO_LOCK、NO_LOCK_NO_ADMIN に変更されました。NO_LOCK はさらに LOCK TABLES および SELECT ... IN SHARE MODE を許可せず、NO_LOCK_NO_ADMIN は READ ONLY ADMIN 権限を持つユーザーに制限を適用します。
  - optimizer_switch パラメータの有効な値に、duplicateweedout フラグと reorder_outer_joins フラグが含まれるようになりました。
  - slave_type_conversions パラメータの有効値に、ERROR_IF_MISSING_FIELD が含まれるようになりました。
  - group_concat_max_len パラメータの最大値が 18446744073709547520 から 1073741824 に変更されました。
- 新しいパラメータ — MariaDB バージョン 12.3 の DB インスタンスでは、以下のパラメータが新しくなります。
  - max_open_cursors パラメータは、セッションごとに開いているカーソルの最大数を設定します。デフォルト値は 50 で、有効な値は 0 ~ 65536 です。
  - aria_pagecache_segments パラメータは、Aria ページキャッシュ内のセグメント数を設定し、ミューテックス競合を減らします。デフォルト値は 1 で、有効な値は 1 ~ 128 です。
  - metadata_locks_instances パラメータは、メタデータロックの高速レーンの数を設定します。これにより、データ操作言語 (DML) のスケーラビリティが向上します。デフォルト値は 8 で、有効な値は 1 ~ 256 です。
  - create_tmp_table_binlog_formats パラメータは、一時テーブルに許可されるバイナリログ形式を設定します。デフォルト値は STATEMENT で、有効な値は MIXED および STATEMENT です。

MariaDB 12.3 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「MariaDB 12.3 の変更点と改善点」および「リリースノート - MariaDB 12.3 シリーズ」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 11.8 のサポート

Amazon RDS は、MariaDB バージョン 11.8 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

> **Note**
>
> MariaDB 11.8 では、`require_secure_transport` のデフォルト値は `1` になり、安全な SSL/TLS 接続が必要です。非セキュア接続が必要な場合は、`0` に設定します。

- パラメータの新しいデフォルト値 – `require_secure_transport` パラメータのデフォルト値が `0` から `1` に変更され、デフォルトで安全なトランスポート接続が強制されます。詳細については、「Amazon RDS 上の MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求する」を参照してください。
- ベクトルサポート – MariaDB Vector を使用して、AI で生成されたベクトルを MariaDB に直接保存および検索できます。この機能では、次のシステム変数が導入されています。
  - 変数 `mhnsw_default_distance` は、MHNSW ベクトルインデックス作成のデフォルトの距離メトリクスを指定します。
  - 変数 `mhnsw_default_m` は、MHNSW ベクトルインデックス作成の `M` パラメータのデフォルト値を定義します。
  - 変数 `mhnsw_ef_search` は、ベクトルインデックス検索の結果候補の最小数を定義します。
  - 変数 `mhnsw_max_cache_size` は、1 つの MHNSW ベクトルインデックスキャッシュの上限を設定します。
- 一時ファイルサイズ制限 – RDS Maria DB 11.8 パラメータグループで使用できる 2 つのシステム変数を使用して、作成されたディスク一時ファイルとテーブルのサイズを制限できるようになりました。
  - 変数 `max_tmp_session_space_usage` は、ユーザーあたりの一時スペース許容量を制限します。
  - 変数 `max_tmp_total_space_usage` は、すべてのユーザーの一時スペースの許容量を制限します。
- 一時テーブルスペース管理 – 一時テーブルスペースは一時テーブルを保存し、データが追加されると拡大します。一時テーブルが削除されても、スペースは自動的に再利用されません。手順

mysql.rds_execute_operation を使用して、一時テーブルスペースを縮小し、ディスク容量を再利用できます。

MariaDB 11.8 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「Changes and improvements in MariaDB 11.8」および「Release notes - MariaDB 11.8 series」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 11.4 のサポート

Amazon RDS は、MariaDB バージョン 11.4 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- 暗号化ライブラリ – RDS for MariaDB は OpenSSL を FIPS 140-3 認定の AWS Libcrypto (AWS-LC) に置き換えました。
- Simple Password Check プラグイン – MariaDB の Simple Password Check プラグインを使用して、パスワードに特定のタイプの文字が少なくとも特定の数だけ含まれているかどうかを確認できます。詳細については、「the section called “パスワード検証プラグイン”」を参照してください。
- Cracklib Password Check プラグイン – MariaDB の Cracklib Password Check プラグインを使用して、新しいパスワードの強度を確認できます。詳細については、「the section called “パスワード検証プラグイン”」を参照してください。
- InnoDB の機能強化 — これらの機能には、次の項目があります。
  - 変更バッファが削除されました。詳細については、「InnoDB Change Buffering」を参照してください。
  - InnoDB デフラグが削除されました。詳細については、「InnoDB Defragmentation」を参照してください。
- 新しい権限 – 管理者ユーザーに `SHOW CREATE ROUTINE` 権限も付与されました。この権限により、被付与者は、他のユーザーが所有するルーチンの `SHOW CREATE` 定義ステートメントを表示できます。詳細については、「Database Privileges」を参照してください。
- レプリケーションの改善 – MariaDB バージョン 11.4 の DB インスタンスは、バイナリログのインデックス作成をサポートしています。バイナリログファイルごとに GTID インデックスを作成できます。これらのインデックスは、GTID の検索にかかる時間を短縮することで、レプリケーションのパフォーマンスを向上させます。詳細については、「Binlog Indexing」を参照してください。

- 廃止または削除されたパラメータ — MariaDB バージョン 11.4 の DB インスタンスでは、以下のパラメータが廃止または削除されました。
  - `engine_condition_pushdown` を optimizer_switch から削除
  - innodb_change_buffer_max_size
  - innodb_defragment
  - `TLSv1.0` と `TLSv1.1` を tls_version から削除
- パラメータの新しいデフォルト値 – innodb_undo_tablespaces パラメータのデフォルト値が `0` から `3` に変更されました。
- パラメータの新しい有効な値 – MariaDB バージョン 11.4 の DB インスタンスでは、以下のパラメータに新しい有効な値が追加されました。
  - binlog_row_image パラメータの有効な値として `FULL_NODUP` が追加されました。
  - OLD_MODE パラメータの有効な値として `NO_NULL_COLLATION_IDS` が追加されました。
- 新しいパラメータ — MariaDB バージョン 11.4 の DB インスタンスでは、以下のパラメータが新しくなります。
  - transaction_isolation パラメータは tx_isolation パラメータに置き換わります。
  - transaction_read_only パラメータは tx_read_only パラメータに置き換わります。
  - block_encryption_mode パラメータは、AES_ENCRYPT() 関数と AES_DECRYPT() 関数のデフォルトのブロック暗号化モードを定義します。
  - character_set_collations は、文字セットのデフォルトの照合順序のオーバーライドを定義します。
  - binlog_gtid_index、binlog_gtid_index_page_size、binlog_gtid_index_span_min は、バイナリログの GTID インデックスのプロパティを定義します。詳細については、「Binlog Indexing」を参照してください。

MariaDB 11.4 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「Changes and improvements in MariaDB 11.4」および「Release notes - MariaDB 11.4 series」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 10.11 のサポート

Amazon RDS は、MariaDB バージョン 10.11 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- パスワード再利用チェックプラグイン — MariaDB パスワード再利用チェックプラグインを使用して、ユーザーがパスワードを再利用できないようにしたり、パスワードの保持期間を設定したりできます。詳細については、「パスワード再利用チェックプラグイン」を参照してください。
- GRANT TO PUBLIC 認可 — サーバーにアクセスできるすべてのユーザーに、権限を付与できます。詳細については、「GRANT TO PUBLIC」を参照してください。
- SUPER と READ ONLY ADMIN 権限の分離 — 以前に SUPER 権限を持っていたユーザーも含め、すべてのユーザーから READ ONLY ADMIN 権限を削除できます。
- セキュリティ — MariaDB クライアントのデフォルトとしてオプション `--ssl` を設定できるようになりました。MariaDB は、設定が正しくない場合でも SSL をサイレントに無効にすることがなくなりました。
- SQL コマンドと関数 — `SHOW ANALYZE FORMAT=JSON` コマンドと関数 `ROW_NUMBER`、`SFORMAT`、および `RANDOM_BYTES`、`SFORMAT` を使用して、文字列の書式設定を可能にできるようになりました。これはデフォルトで有効になっています。1 つのコマンドでパーティションをテーブルに、テーブルをパーティションに変換できます。`JSON_*( )` 関数に関して他にもいくつかの改善点があります。`DES_ENCRYPT` と `DES_DECRYPT` 関数はバージョン 10.10 以降では廃止されました。詳細については、「SFORMAT」を参照してください。
- InnoDB の機能強化 — これらの機能には、次の項目があります。
  - REDO ログのパフォーマンスが向上し、書き込みの増幅が減り、同時実行性が向上しました。
  - データディレクトリを再初期化しなくても、UNDO テーブルスペースを変更できるようになりました。この機能強化により、コントロールプレーンのオーバーヘッドが軽減されます。再起動は必要ですが、undo テーブルスペースを変更した後に再初期化する必要はありません。
  - `CHECK TABLE … EXTENDED` および内部での降順インデックスのサポート。
  - 一括挿入が改善されました。
- バイナリログの変更 — これらの変更には、次の項目があります。
  - 2 つのフェーズに `ALTER` のログを記録することにより、レプリケーションのレイテンシーを減少させます。`binlog_alter_two_phase` パラメータはデフォルトでは無効になっていますが、パラメータグループを通じて有効にできます。
  - `explicit_defaults_for_timestamp` のログ記録。
  - トランザクションを安全にロールバックできる場合、`INCIDENT_EVENT` のログ記録は行われなくなりました。
- レプリケーションの改善 — MariaDB バージョン 10.11 DB インスタンスは、デフォルトで GTID レプリケーションを使用します (マスターがサポートしている場合)。また、`Seconds_Behind_Master` はより正確です。

- クライアント — mysqlbinlog および mariadb-dump の新しいコマンドラインオプションを使用できます。mariadb-dump を使用して、履歴データをダンプして復元できます。
- システムバージョン管理 — 履歴を変更できます。MariaDB は自動的に新しいパーティションを作成します。
- アトミック DDL — CREATE OR REPLACE がアトミックになりました。ステートメントは連続するか、完全に反転するかのどちらかです。
- ログの書き込みをやり直す — ログの書き込みを非同期でやり直します。
- ストアド関数 — ストアド関数が、ストアドプロシージャと同じIN、OUT、と INOUT パラメータをサポートするようになりました。
- 非推奨または削除されたパラメータ — 次のパラメータは、MariaDB バージョン 10.11 DB インスタンスに対して廃止または削除されました。
  - innodb_change_buffering
  - innodb_disallow_writes
  - innodb_log_write_ahead_size
  - innodb_prefix_index_cluster_optimization
  - keep_files_on_create
  - old
- ダイナミックパラメータ — 次のパラメータには、MariaDB バージョン 10.11 DB インスタンスに対して動的になりました。
  - innodb_log_file_size
  - innodb_write_io_threads
  - innodb_read_io_threads
- パラメータの新しいデフォルト値 – 次のパラメータには、MariaDB バージョン 10.11 DB インスタンスの新しいデフォルト値があります。
  - explicit_defaults_for_timestamp パラメータのデフォルト値が、OFF から ON に変更されました。
  - optimizer_prune_level パラメータのデフォルト値が、1 から 2 に変更されました。
- パラメータの新しい有効値 – 次のパラメータには、MariaDB バージョン 10.11 DB インスタンスの新しい有効値があります。
  - old パラメータの有効値が、old_mode パラメータの有効値に統合されました。
  - histogram_type パラメータの有効値に、JSON_HB が含まれるようになりました。
  - innodb_log_buffer_size パラメータの有効値の範囲が、262144 から 4294967295 (256 キロバイトから 4096 メガバイト) になりました。

  - innodb_log_file_size パラメータの有効値の範囲が、4194304 から 512GB (4 メガバイトから 512 ギガバイト) になりました。
  - optimizer_prune_level パラメータの有効値に、2 が含まれるようになりました。
- 新しいパラメータ — 次のパラメータには、MariaDB バージョン 10.11 DB インスタンスに対して新しくなりました。
  - binlog_alter_two_phase パラメータは、レプリケーションパフォーマンスを改善できます。
  - log_slow_min_examined_row_limit は、パフォーマンスを改善できます。
  - log_slow_query パラメータと log_slow_query_file パラメータは、slow_query_log と slow_query_log_file それぞれのエイリアスです。
  - optimizer_extra_pruning_depth
  - system_versioning_insert_history

MariaDB 10.11 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「Changes and improvements in MariaDB 10.11」および「Release notes - MariaDB 10.11 series」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 10.6 のサポート

Amazon RDS は、MariaDB バージョン 10.6 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- MyRocks ストレージエンジン – MyRocks ストレージエンジンを RDS for MariaDB とともに使用して、書き込み負荷の高い高性能ウェブアプリケーションのストレージ消費を最適化できます。詳細については、「Amazon RDS の MariaDB でサポートされているストレージエンジン」と「MyRocks」を参照してください。
- AWS Identity and Access ManagementIAM DB authentication (IAM データベース認証) – IAM DB 認証を使用して、MariaDB DB インスタンスへの接続のセキュリティを強化し、一元管理できます。詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。
- アップグレードオプション – 以前のメジャーリリース (10.3、10.4、10.5) から RDS for MariaDB バージョン 10.6 にアップグレードできるようになりました。既存の MySQL 5.6 または 5.7 DB インスタンスのスナップショットを MariaDB 10.6 インスタンスに復元することもできます。(詳しくは、「MariaDB DB エンジンのアップグレード」を参照してください。)

- レプリケーションの遅延 – リードレプリカがソースデータベースより遅延する時間を設定できるようになりました。標準の MariaDB レプリケーション設定では、ソースとレプリカ間のレプリケーション遅延は最小限に抑えられます。レプリケーションの遅延では、災害対策用の戦略として意図的に遅延を設定できます。詳細については、「MariaDB での遅延レプリケーションの設定」を参照してください。
- Oracle PL/SQL の互換性 – RDS for MariaDB バージョン 10.6 を使用すると、レガシー Oracle アプリケーションを Amazon RDS にさらに簡単に移行できます。詳細については、「SQL_MODE=ORACLE」を参照してください。
- アトミック DDL – RDS for MariaDB バージョン 10.6 では、動的データ言語 (DDL) ステートメントは比較的クラッシュセーフです。`CREATE TABLE`、`ALTER TABLE`、`RENAME TABLE`、`DROP TABLE`、`DROP DATABASE`、および関連する DDL ステートメントがアトミックになりました。ステートメントは連続するか、完全に反転するかのどちらかです。詳細については、「Atomic DDL」(アトミック DDL) を参照してください。
- その他の機能強化 – SQL 内で JSON データをリレーショナル形式に変換する `JSON_TABLE` 関数や、Innodb による空のテーブルデータのロードの高速化などの機能強化が行われました。また、分析とトラブルシューティングのための新しい `sys_schema`、未使用のインデックスを無視するためのオプティマイザの機能強化、およびパフォーマンスの向上も行いました。詳細については、「JSON_TABLE」を参照してください。
- パラメータの新しいデフォルト値 – 次のパラメータには、MariaDB バージョン 10.6 DB インスタンスの新しいデフォルト値があります。
  - 次のパラメータのデフォルト値が `utf8` から `utf8mb3` に変更されました。
    - character_set_client
    - character_set_connection
    - character_set_results
    - character_set_system

    これらのパラメータのデフォルト値は変更されていますが、機能の変更はありません。詳細については、MariaDB ドキュメントの「Supported Character Sets and Collations」(サポートされている文字セットと照合順序) を参照してください。
  - collation_connection パラメータのデフォルト値が `utf8_general_ci` から `utf8mb3_general_ci` に変更されました。このパラメータのデフォルト値は変更されていますが、機能の変更はありません。
  - old_mode パラメータのデフォルト値が未設定から `UTF8_IS_UTF8MB3` に変更されました。このパラメータのデフォルト値は変更されていますが、機能の変更はありません。

MariaDB 10.6 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「Changes and improvements in MariaDB 10.6」(MariaDB 10.6 の変更点と改善点) および「Release notes - MariaDB 10.6 series」(リリースノート - MariaDB 10.6 シリーズ) を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 10.5 のサポート

Amazon RDS は、MariaDB バージョン 10.5 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- InnoDB の強化 - MariaDB バージョン 10.5 には、InnoDB の機能強化が含まれています。詳細については、MariaDB ドキュメントの「 InnoDB: パフォーマンスの改善など」を参照してください。
- Performance Schemaの更新 - MariaDB バージョン 10.5 には、Performance Schemaの更新が含まれています。詳細については、MariaDB ドキュメントの「MySQL 5.7 インストルメンテーションとテーブルと一致する Performance Schema の更新」を参照してください。
- InnoDB 再実行ログ内の 1 つのファイル - MariaDB のバージョン 10.5 より前のバージョンでは、`innodb_log_files_in_group` パラメータの値が 2 に設定されていました。MariaDB バージョン 10.5 では、このパラメータの値は 1 に設定されます。

  以前のバージョンから MariaDB バージョン 10.5 にアップグレードし、パラメータを変更しない場合、`innodb_log_file_size` パラメータ値は変更されません。ただし、2 つのログファイルではなく 1 つのログファイルに適用されます。その結果、アップグレードされた MariaDB バージョン 10.5 DB インスタンスは、アップグレード前に使用していた再実行ログサイズの半分を使用します。この変更は、パフォーマンスに顕著な影響を与える可能性があります。この問題に対処するには、`innodb_log_file_size` パラメータの値を 2 倍にします。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。
- SHOW SLAVE STATUS コマンドはサポートされていません。- MariaDB のバージョン 10.5 より前のバージョンでは、`SHOW SLAVE STATUS`コマンドに `REPLICATION SLAVE` 特権が必要でした。MariaDB バージョン 10.5 では、同等の `SHOW REPLICA STATUS` コマンドに `REPLICATION REPLICA ADMIN` 権限が必要です。この新しい特権は RDS マスターユーザーに付与されません。

  `SHOW REPLICA STATUS` コマンドを使用する代わりに、新しい `mysql.rds_replica_status` 保存済み手順を実行して、同様の情報を返します。詳細については、「mysql.rds_replica_status」を参照してください。

- SHOW RELAYLOG EVENTS コマンドはサポートされていません。-MariaDB のバージョン 10.5 より前のバージョンでは、SHOW RELAYLOG EVENTS コマンドには REPLICATION SLAVE 特権が必要でした。MariaDB バージョン 10.5 では、このコマンドには REPLICATION REPLICA ADMIN 特権が必要です。この新しい特権は RDS マスターユーザーに付与されません。
- パラメータの新しいデフォルト値 - 次のパラメータには、MariaDB バージョン 10.5 DB インスタンスの新しいデフォルト値があります。
  - max_connections パラメータのデフォルト値が LEAST({DBInstanceClassMemory/25165760},12000) に変更されました。LEASTパラメータ関数の詳細については、「DB パラメータ関数」を参照してください。
  - innodb_adaptive_hash_index パラメータのデフォルト値が OFF (0) に変更されました。
  - innodb_checksum_algorythmのパラメータのデフォルト値が full_crc32 に変更されました。
  - innodb_log_file_size パラメータのデフォルト値が 2 GB に変更されました。

MariaDB 10.5 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「MariaDB 10.5 の変更点と改善点」および「リリースノート - MariaDB 10.5 シリーズ」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS での MariaDB 10.4 のサポート

Amazon RDS は、MariaDB バージョン 10.4 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- ユーザーアカウントのセキュリティの強化 - パスワードの有効期限とアカウントのロックの改善
- オプティマイザの機能強化 - オプティマイザのトレース機能
- InnoDB の機能強化 - インスタント DROP COLUMN のサポートと VARCHAR および ROW_FORMAT=DYNAMIC に対する ROW_FORMAT=COMPACT のエクステンション
- 新しいパラメータ - tcp_nodedelay、tls_version、および gtid_cleanup_batch_size を含む

MariaDB 10.4 のすべての機能と関連ドキュメントのリストについては、MariaDB ウェブサイトで「MariaDB 10.4 の変更点と改善点」および「リリースノート - MariaDB 10.4 シリーズ」を参照してください。

サポートされない機能の一覧については、「Amazon RDS でサポートされていない MariaDB の機能」を参照してください。

Amazon RDS の MariaDB でサポートされているストレージエンジン

RDS for MariaDB では、次のストレージエンジンがサポートされています。

トピック

- InnoDB ストレージエンジン
- MyRocks ストレージエンジン

他のストレージエンジンは現在、RDS for MariaDB ではサポートされていません。

InnoDB ストレージエンジン

MariaDB では様々な機能を持つ複数のストレージエンジンがサポートされていますが、それらすべてがリカバリとデータ耐久性に最適化されているわけではありません。InnoDB は、Amazon RDS での MariaDB DB インスタンスについて推奨されているストレージエンジンです。Amazon RDS でのポイントインタイムの復元やスナップショット復元などの機能には、回復可能なストレージエンジンが必要であり、MariaDB バージョンで推奨された ストレージエンジンのみがサポートされています。

詳細については、「InnoDB」を参照してください。

MyRocks ストレージエンジン

MyRocks ストレージエンジンは RDS で MariaDB バージョン 10.6 以降で使用可能です。MyRocks ストレージエンジンを本番データベースで使用する前に、徹底したベンチマークとテストを実行して、ユースケースに対する InnoDB の潜在的な利点を検証することをお勧めします。

MariaDB バージョン 10.6 のデフォルトのパラメータグループには MyRocks パラメータが含まれています。詳細については、「MariaDB のパラメータ」および「Amazon RDS のパラメータグループ」を参照してください。

MyRocks ストレージエンジンを使用するテーブルを作成するには、`CREATE TABLE` ステートメントに `ENGINE=RocksDB` を指定します。次の例では、MyRocks ストレージエンジンを使用するテーブルを作成します。

```
CREATE TABLE test (a INT NOT NULL, b CHAR(10)) ENGINE=RocksDB;
```

InnoDB テーブルと MyRocks テーブルの両方にまたがるトランザクションを実行しないことを強くお勧めします。MariaDB は、ストレージエンジン間のトランザクションに対して ACID (不可分性、整合性、分離性、耐久性) を保証するものではありません。DB インスタンスに InnoDB テーブルと MyRocks テーブルの両方を含めることは可能ですが、一方のストレージエンジンから別のストレージエンジンへの移行中を除き、この方法はお勧めしません。InnoDB テーブルと MyRocks テーブルの両方が DB インスタンスに存在する場合、各ストレージエンジンには独自のバッファプールがあり、パフォーマンスが低下する可能性があります。

MyRocks では SERIALIZABLE アイソレーションまたはギャップロックはサポートされていません。したがって、一般的に MyRocks をステートメントベースのレプリケーションで使用することはできません。詳細については、「MyRocks and Replication」(MyRocks とレプリケーション) を参照してください。

現在、次の MyRocks パラメータのみ変更できます。

- rocksdb_block_cache_size
- rocksdb_bulk_load
- rocksdb_bulk_load_size
- rocksdb_deadlock_detect
- rocksdb_deadlock_detect_depth
- rocksdb_max_latest_deadlocks

MyRocks ストレージエンジンと InnoDB ストレージエンジンは、rocksdb_block_cache_size および innodb_buffer_pool_size パラメータの設定により競合する可能性があります。場合によっては、特定の DB インスタンスで MyRocks ストレージエンジンのみを使用したいことがあります。その場合は、innodb_buffer_pool_size minimal パラメータを最小値に設定し、rocksdb_block_cache_size をできる限り高く設定することをお勧めします。

DescribeDBLogFiles および DownloadDBLogFilePortion オペレーションを使用して MyRocks ログファイルにアクセスできます。

MyRocks の詳細については、MariaDB のウェブサイトの「MyRocks」を参照してください。

Amazon RDS での MariaDB のキャッシュウォームアップ

InnoDB キャッシュウォームアップでは、DB インスタンスがシャットダウンされたときのバッファプールの最新の状態を保存し、DB インスタンスが起動されたときに保存された情報からバッファ

プールを再ロードすることによって、MariaDB DB インスタンスのパフォーマンスを向上させることができます。このアプローチにより、通常のデータベースの使用からバッファプールを「ウォームアップする」必要がなくなり、既知の一般的なクエリのページを使用してバッファプールを事前にロードします。キャッシュウォームアップの詳細については、MariaDB ドキュメントの「バッファプールのダンプと復元」を参照してください。

MariaDB 10.3 以上の DB インスタンスでは、キャッシュウォームアップはデフォルトで有効になっています。有効にするには、DB インスタンスのパラメータグループで `innodb_buffer_pool_dump_at_shutdown` および `innodb_buffer_pool_load_at_startup` パラメータを 1 に設定します。パラメータグループのこれらのパラメータ値を変更すると、パラメータグループを使用するすべての MariaDB DB インスタンスに影響します。特定の MariaDB DB インスタンスの キャッシュウォームアップを有効にするには、それらの DB インスタンスの新しいパラメータグループを作成することが必要になる場合があります。パラメータグループについては、「Amazon RDS のパラメータグループ」を参照してください。

キャッシュウォームアップは、主に、スタンダードストレージを使用する DB インスタンスにパフォーマンス上のメリットをもたらします。PIOPS ストレージを使用する場合、一般的に大きなパフォーマンス上のメリットは見られません。

Important

フェイルオーバー時など、MariaDB DB インスタンスが正常にシャットダウンしなかった場合、バッファプールの状態はディスクに保存されません。この場合、DB インスタンスが再開されるときに、MariaDB は利用可能なバッファプールファイルをロードします。特に害はありませんが、復元済みバッファプールは、再起動前のバッファプールの最新の状態を反映していない可能性があります。スタートアップ時に キャッシュをウォームアップするために、バッファプールの最新の状態を利用できるように、定期的に「オンデマンド」でバッファプールをダンプすることをお勧めします。バッファプールをオンデマンドでダンプまたはロードできます。

自動で定期的にバッファプールをダンプするイベントを作成できます。例えば、次のステートメントは、1 時間ごとにバッファプールをダンプする `periodic_buffer_pool_dump` という名前のイベントを作成します。

```
CREATE EVENT periodic_buffer_pool_dump
   ON SCHEDULE EVERY 1 HOUR
   DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

詳細については、MariaDB のドキュメントの「イベント」を参照してください。

オンデマンドでのバッファプールのダンプとロード

次のストアドプロシージャを使用して、 キャッシュをオンデマンドで格納しロードできます。

- バッファプールの現在の状態をディスクにダンプするには、mysql.rds_innodb_buffer_pool_dump_now ストアドプロシージャを呼び出します。
- バッファプールの保存された状態をディスクからロードするには、mysql.rds_innodb_buffer_pool_load_now ストアドプロシージャを呼び出します。
- 進行中のロードオペレーションをキャンセルするには、mysql.rds_innodb_buffer_pool_load_abort ストアドプロシージャを呼び出します。

Amazon RDS でサポートされていない MariaDB の機能

以下の MariaDB 機能は、Amazon RDS ではサポートされていません。

- S3 ストレージエンジン
- 認証用プラグイン - GSSAPI
- 認証用プラグイン - Unix ソケット
- AWS Key Management 暗号化プラグイン
- 10.6 より前のバージョンの MariaDB の遅延レプリケーション
- InnoDB および Aria で保管時のネイティブ MariaDB 暗号化

  MariaDB DB インスタンス用に保管時の暗号化を有効にするには、「Amazon RDS リソースの暗号化」の指示に従います。
- HandlerSocket
- 10.6 より前のバージョンの MariaDB の JSON テーブルタイプ
- MariaDB ColumnStore
- MariaDB Galera クラスター
- マルチ出典レプリケーション
- 10.6 より前のバージョンの MariaDB の MyRocks ストレージエンジン
- 11.4 より前のバージョンの MariaDB のパスワード検証プラグイン `simple_password_check` および `cracklib_password_check`
- InnoDB for MariaDB バージョン 12.3 の新しいバイナリログ実装

RDS for MariaDB は、MariaDB 12.3 で導入される InnoDB でのバイナリログ実装をサポートしていません。`binlog_storage_engine` パラメータは、`mariadb12.3` パラメータグループファミリーでは使用できません。このコミュニティ機能の詳細については、MariaDB ウェブサイトの「New binlog implementation in MariaDB 12.3」を参照してください。

- `PATH` 変数は `mariadb12.3` パラメータグループファミリーで使用できないため、グローバル変数として設定することはできません。`PATH` をセッションレベルで設定することは、引き続き可能です。
- `SHOW REPLICA HOSTS` の出力のレプリケーションユーザー情報

  `show_slave_auth_info` 変数は `mariadb12.3` パラメータグループファミリーで使用できないため、`SHOW REPLICA HOSTS` の出力にはユーザー情報およびパスワード情報は含まれません。
- Spider ストレージエンジン
- Sphinx ストレージエンジン
- TokuDB ストレージエンジン
- MariaDB ドキュメントの「エンジン定義の新しいテーブル/フィールド/インデックス属性」で説明されている、ストレージエンジン固有のオブジェクト属性。
- テーブルとテーブルスペースの暗号化
- Hashicorp Key Management プラグイン
- 2 つのアップグレードを平行実行

マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスはできません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。Amazon RDS は、任意のスタンダード SQL クライアントアプリケーションによる、DB インスタンス上のデータベースへのアクセスがサポートされています。Amazon RDS では、Telnet、Secure Shell (SSH)、または Windows のリモートデスクトップ接続を使用しての、DB インスタンスに対するダイレクトホストアクセスは行えません。

Amazon RDS の MariaDB のバージョン

MariaDB の場合、バージョン番号は「バージョン X.Y.Z 」の形式で記述されます。Amazon RDS の用法では、X.Y はメジャーバージョン番号を、Z はマイナーバージョン番号を表します。Amazon RDS 実装では、メジャーバージョン番号が変更された場合 (10.5 から 10.6 へなど)、そのバージョン変更はメジャーと考えます。マイナーバージョン番号のみが変更された場合 (10.6.14 から 10.6.16 へなど)、そのバージョン変更はマイナーと考えます。

トピック

- バージョン最新性タイムライン
- Amazon RDS でサポートされている MariaDB のマイナーバージョン
- Amazon RDS でサポートされている MariaDB のメジャーバージョン
- データベースプレビュー環境の使用
- データベースプレビュー環境の MariaDB バージョン 12.3.1
- Amazon RDS for MariaDB の非推奨バージョン

バージョン最新性タイムライン

RDS for MariaDB は、コミュニティデータベースエンジンのリリースを定義された頻度で追跡します。これらのバージョン最新性タイムラインは、その頻度を明確にするために公開されます。これらのタイムラインを使用して、次のことができます。

- メジャーバージョンのアップグレードを計画し、新しい RDS for MariaDB メジャーバージョンが利用可能になる時期を見積もります。
- メンテナンス期間中にマイナーバージョンアップグレードをスケジュールします。

次の表に、RDS for MariaDB のバージョン最新性タイムラインを示します。

リリースタイプ	タイムライン
メジャーバージョン	最初のパッチバージョンのコミュニティリリースから 3 か月以内
マイナーバージョン	コミュニティリリースから 30 日以内

Amazon RDS でサポートされている MariaDB のマイナーバージョン

Amazon RDS は、現在、MariaDB の以下のマイナーバージョンをサポートしています。

> **Note**
>
> 月と年のみの日付はおおよその日付であり、確定後に正確な日付で更新されます。

次の表に、Amazon RDS が現在サポートしている MariaDB 12.3 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
12.3.2	2026 年 5 月 28 日	2026 年 8 月 7 日	2027 年 8 月

次の表に、Amazon RDS が現在サポートしている MariaDB 11.8 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
11.8.8	2026 年 5 月 27 日	2026 年 6 月 5 日	2027 年 6 月
11.8.6	2026 年 2 月 4 日	2026 年 2 月 10 日	2027 年 2 月
11.8.5	2025 年 11 月 14 日	2025 年 11 月 20 日	2026 年 10 月 31 日
11.8.3	2025 年 8 月 6 日	2025 年 8 月 25 日	2026 年 10 月 31 日

次の表に、Amazon RDS が現在サポートしている MariaDB 11.4 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
11.4.12	2026 年 5 月 27 日	2026 年 6 月 5 日	2027 年 6 月
11.4.10	2026 年 2 月 4 日	2026 年 2 月 10 日	2027 年 2 月

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
11.4.9	2025 年 11 月 6 日	2025 年 11 月 18 日	2026 年 10 月 31 日
11.4.8	2025 年 8 月 6 日	2025 年 8 月 13 日	2026 年 10 月 31 日
11.4.7	2025 年 5 月 22 日	2025 年 6 月 4 日	2026 年 10 月 31 日

次の表に、Amazon RDS が現在サポートしている MariaDB 10.11 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
10.11.18	2026 年 5 月 27 日	2026 年 6 月 5 日	2027 年 6 月
10.11.16	2026 年 2 月 4 日	2026 年 2 月 10 日	2027 年 2 月
10.11.15	2025 年 11 月 6 日	2025 年 11 月 18 日	2026 年 10 月 31 日
10.11.14	2025 年 8 月 6 日	2025 年 8 月 13 日	2026 年 10 月 31 日
10.11.13	2025 年 5 月 22 日	2025 年 6 月 4 日	2026 年 10 月 31 日

次の表に、Amazon RDS が現在サポートしている MariaDB 10.6 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
10.6.27	2026 年 5 月 27 日	2026 年 6 月 5 日	2026 年 12 月 31 日
10.6.25	2026 年 2 月 4 日	2026 年 2 月 10 日	2026 年 12 月 31 日
10.6.24	2025 年 11 月 6 日	2025 年 11 月 18 日	2026 年 11 月
10.6.23	2025 年 8 月 6 日	2025 年 8 月 13 日	2026 年 11 月

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
10.6.22	2025 年 5 月 6 日	2025 年 5 月 20 日	2026 年 11 月

次の表に、Amazon RDS が現在サポートしている MariaDB 10.5 のマイナーバージョンを示します。

MariaDB エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
10.5.29	2025 年 5 月 6 日	2025 年 5 月 20 日	2026 年 8 月
10.5.28	2025 年 2 月 4 日	2025 年 2 月 24 日	2026 年 8 月

新しい DB インスタンスを作成するときは、現在サポートされているいずれかの MariaDB バージョンを指定できます。メジャーバージョン (MariaDB 10.5 など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいずれかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合は、Amazon RDS では、お客様が指定したメジャーバージョンの最新リリースにデフォルトで設定されます。サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示するには、AWS CLI の `describe-db-engine-versions` コマンドを使用します。

例えば、RDS for MariaDB でサポートされているエンジンバージョンを一覧表示するには、次の CLI コマンドを実行します。

```
aws rds describe-db-engine-versions --engine mariadb --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

デフォルトの MariaDB バージョンは、AWS リージョン によって異なる場合があります。特定のマイナーバージョンで DB インスタンスを作成するには、DB インスタンスの作成中にマイナーバージョンを指定します。次の AWS リージョン コマンドを実行して、AWS CLI のデフォルトのマイナーバージョンを確認できます。

```
aws rds describe-db-engine-versions --default-only --engine mariadb
 --engine-version major_engine_version --region region --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

major-engine-version をメジャーエンジンバージョンに置き換え、*#####*を AWS リージョンに置き換えます。例えば、次の AWS CLI コマンドは、10.5 メジャーバージョンおよび米国西部 (オレゴン) AWS リージョン (us-west-2) 用の、デフォルトの MariaDB マイナーエンジンのバージョンを返します。

```
aws rds describe-db-engine-versions --default-only --engine mariadb --engine-version
 10.5 --region us-west-2 --query "*[].{Engine:Engine,EngineVersion:EngineVersion}" --
output text
```

Amazon RDS の MariaDB マイナーバージョン

マイナーバージョン

- MariaDB バージョン 12.3.2
- MariaDB バージョン 11.8.8
- MariaDB バージョン 11.8.6
- MariaDB バージョン 11.8.5
- MariaDB バージョン 11.8.3
- MariaDB バージョン 11.4.12
- MariaDB バージョン 11.4.10
- MariaDB バージョン 11.4.9
- MariaDB バージョン 11.4.8
- MariaDB バージョン 11.4.7
- MariaDB バージョン 11.4.5
- MariaDB バージョン 11.4.4
- MariaDB バージョン 10.11.18
- MariaDB バージョン 10.11.16
- MariaDB バージョン 10.11.15
- MariaDB バージョン 10.11.14
- MariaDB バージョン 10.11.13
- MariaDB バージョン 10.11.11
- MariaDB バージョン 10.11.10
- MariaDB バージョン 10.6.27
- MariaDB バージョン 10.6.25

- MariaDB バージョン 10.6.24
- MariaDB バージョン 10.6.23
- MariaDB バージョン 10.6.22
- MariaDB バージョン 10.6.21
- MariaDB バージョン 10.6.20
- MariaDB バージョン 10.5.29
- MariaDB バージョン 10.5.28
- MariaDB バージョン 10.5.27

MariaDB バージョン 12.3.2

Amazon RDS で MariaDB バージョン 12.3.2 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- RDS Proxy の予約済みユーザー – `rdsproxyadmin` ユーザーは予約済みユーザーです。このユーザーを変更または削除することはできません。詳細については、「Amazon RDS での MariaDB のセキュリティ」を参照してください。
- レプリケーションユーザーアカウントの削除保護 – RDS for MariaDB バージョン 12.3 以降、削除保護はホスト値にかかわらず `rdsrepladmin` アカウントに適用されます。どのホストの `'rdsrepladmin'@'host'` であっても削除を試みると、エラーが発生します。詳細については、「Amazon RDS での MariaDB のセキュリティ」を参照してください。

MariaDB バージョン 11.8.8

Amazon RDS で MariaDB バージョン 11.8.8 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2026b` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 11.8.6

Amazon RDS で MariaDB バージョン 11.8.6 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025c に基づいてタイムゾーン情報を更新しました。
- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MariaDB バージョン 11.8.5

Amazon RDS で MariaDB バージョン 11.8.5 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 11.8.3

Amazon RDS で MariaDB バージョン 11.8.3 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- パラメータの新しいデフォルト値 – require_secure_transport パラメータのデフォルト値が 0 から 1 に変更され、デフォルトで安全なトランスポート接続が強制されます。詳細については、「Amazon RDS 上の MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求する」を参照してください。

MariaDB バージョン 11.4.12

Amazon RDS で MariaDB バージョン 11.4.12 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2026b に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 11.4.10

Amazon RDS で MariaDB バージョン 11.4.10 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025c に基づいてタイムゾーン情報を更新しました。

- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MariaDB バージョン 11.4.9

Amazon RDS で MariaDB バージョン 11.4.9 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 11.4.8

Amazon RDS で MariaDB バージョン 11.4.8 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 11.4.7

Amazon RDS で MariaDB バージョン 11.4.7 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025b` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 11.4.5

Amazon RDS で MariaDB バージョン 11.4.5 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025a` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 11.4.4

Amazon RDS で MariaDB バージョン 11.4.4 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- ポイントインタイムリカバリ (PITR) が失敗する原因となる 2 つの MariaDB コミュニティの変更を元に戻しました。詳細については、「MariaDB Server Jira issue MDEV-35528」を参照してください。

MariaDB バージョン 10.11.18

Amazon RDS で MariaDB バージョン 10.11.18 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2026b に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.11.16

Amazon RDS で MariaDB バージョン 10.11.16 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025c に基づいてタイムゾーン情報を更新しました。
- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MariaDB バージョン 10.11.15

Amazon RDS で MariaDB バージョン 10.11.15 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 10.11.14

Amazon RDS で MariaDB バージョン 10.11.14 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 10.11.13

Amazon RDS で MariaDB バージョン 10.11.13 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025b に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.11.11

Amazon RDS で MariaDB バージョン 10.11.11 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025a` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.11.10

Amazon RDS で MariaDB バージョン 10.11.10 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- ポイントインタイムリカバリ (PITR) が失敗する原因となる 2 つの MariaDB コミュニティの変更を元に戻しました。詳細については、「MariaDB Server Jira issue MDEV-35528」を参照してください。

MariaDB バージョン 10.6.27

Amazon RDS で MariaDB バージョン 10.6.27 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2026b` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.6.25

Amazon RDS で MariaDB バージョン 10.6.25 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025c` に基づいてタイムゾーン情報を更新しました。
- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MariaDB バージョン 10.6.24

Amazon RDS で MariaDB バージョン 10.6.24 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 10.6.23

Amazon RDS で MariaDB バージョン 10.6.23 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MariaDB バージョン 10.6.22

Amazon RDS で MariaDB バージョン 10.6.22 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025b` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.6.21

Amazon RDS で MariaDB バージョン 10.6.21 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025a` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.6.20

Amazon RDS で MariaDB バージョン 10.6.20 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- ポイントインタイムリカバリ (PITR) が失敗する原因となる 2 つの MariaDB コミュニティの変更を元に戻しました。詳細については、「MariaDB Server Jira issue MDEV-35528」を参照してください。

MariaDB バージョン 10.5.29

Amazon RDS で MariaDB バージョン 10.5.29 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025b` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.5.28

Amazon RDS で MariaDB バージョン 10.5.28 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025a` に基づいてタイムゾーン情報を更新しました。

MariaDB バージョン 10.5.27

Amazon RDS で MariaDB バージョン 10.5.27 が使用可能になりました。このリリースには、MariaDB コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- ポイントインタイムリカバリ (PITR) が失敗する原因となる 2 つの MariaDB コミュニティの変更を元に戻しました。詳細については、「MariaDB Server Jira issue MDEV-35528」を参照してください。

Amazon RDS でサポートされている MariaDB のメジャーバージョン

RDS for MariaDB メジャーバージョンは、少なくとも対応するコミュニティバージョンのコミュニティが終了するまでの期間利用可能です。次の日付を参考にすると、テストおよびアップグレードのサイクルを計画することができます。Amazon は、RDS for MariaDB バージョンのサポートを当初発表よりも長く延長した場合、新しい日付を反映してこの表を更新するようにします。

> **Note**
>
> 月と年のみの日付はおおよその日付であり、確定後に正確な日付で更新されます。

describe-db-major-engine-versions AWS CLI コマンドを実行するか、DescribeDBMajorEngineVersions RDS API オペレーションを使用して、メジャーエンジンバージョンのサポート日に関する情報を表示することもできます。

MariaDB メジャーバージョン	コミュニティリリース日	RDS リリース日	コミュニティサポート終了日	RDS 標準サポート終了日
MariaDB 12.3	2026 年 5 月 28 日	2026 年 8 月 7 日	June 2029	June 2029
MariaDB 11.8	2025 年 8 月 6 日	2025 年 8 月 25 日	June 2028	June 2028
MariaDB 11.4	2024 年 8 月 8 日	2024年 10 月 15 日	2029 年 5 月	2029 年 5 月
MariaDB 10.11	2023 年 2 月 16 日	2023 年 8 月 21 日	2028 年 2 月 16 日	2028 年 2 月
MariaDB 10.6	2021 年 7 月 6 日	2022 年 2 月 3 日	2026 年 7 月 6 日	2026 年 12 月 31 日
MariaDB 10.5	2020 年 6 月 24 日	2021 年 1 月 21 日	2025 年 6 月 24 日	2026 年 8 月 31 日

データベースプレビュー環境の使用

データベースプレビュー環境での RDS for MariaDB DB インスタンスは、他の RDS for MariaDB DB インスタンスと機能的に同様です。ただし、データベースプレビュー環境は本稼働に使用できません。

プレビュー環境には以下の制限があります。

- Amazon RDS は、すべての DB インスタンスを作成から 60 日後にバックアップおよびスナップショットとともに削除します。
- 汎用 SSD およびプロビジョンド IOPS SSD ストレージのみを使用できます。

- DB インスタンスに関して サポート からヘルプを受けることはできません。代わりに、AWS マネージド Q&A コミュニティ、AWS re:Post に質問を投稿できます。
- DB インスタンスのスナップショットを本稼働環境にコピーすることはできません。

プレビューでは、以下のオプションがサポートされています。

- db.m6i、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5 DB インスタンスクラスを使って DB インスタンスを作成できます。RDS インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。
- シングル AZ 配置とマルチ AZ 配置の両方を使用できます。
- 標準の MariaDB ダンプおよびロード機能を使用して、データベースをデータベースプレビュー環境からエクスポートしたり、データベースプレビュー環境にインポートしたりできます。

データベースプレビュー環境でサポートされない機能

以下の機能は、データベースプレビュー環境で使用できません。

- クロスリージョンスナップショットのコピー
- クロスリージョンリードレプリカ
- RDS Proxy

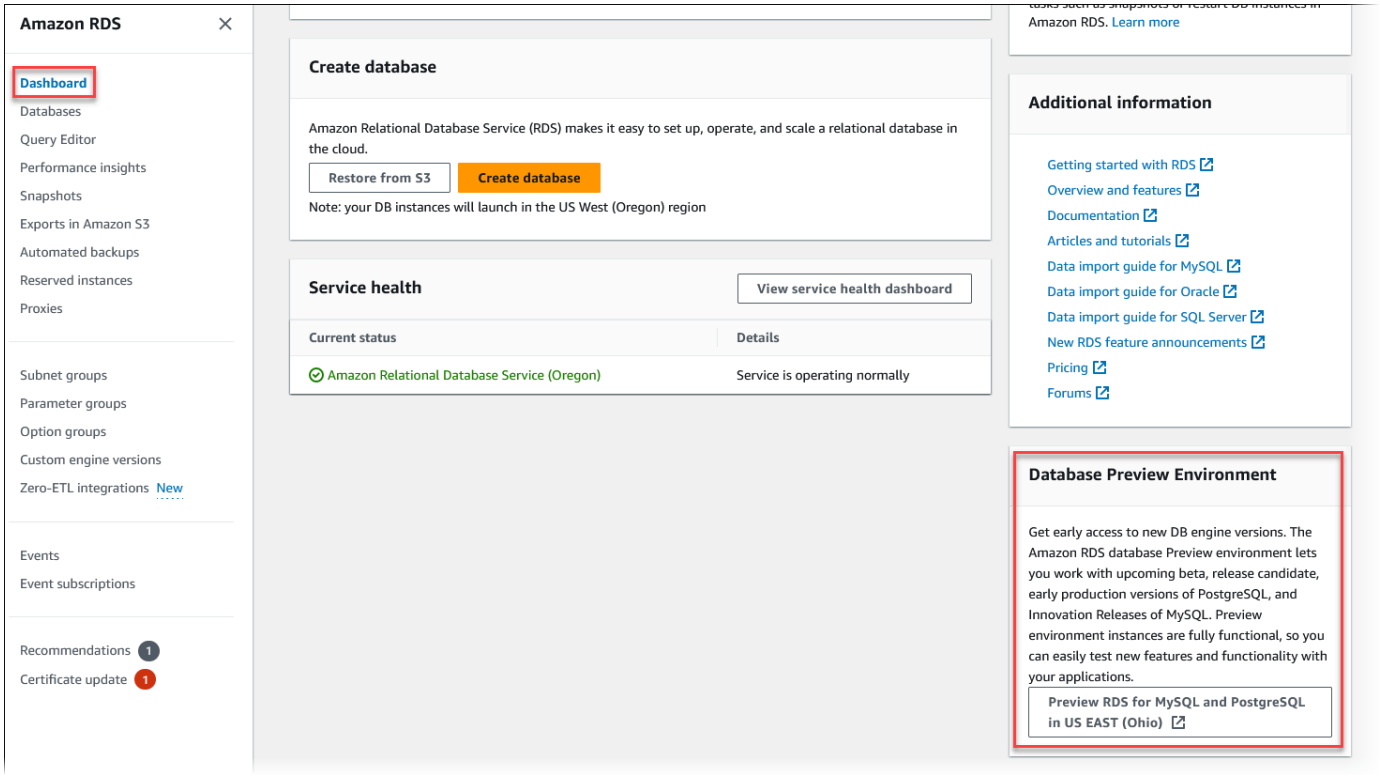
データベースプレビュー環境での新しい DB インスタンスの作成

AWS マネジメントコンソール、AWS CLI または RDS API を使用して、データベースプレビュー環境で DB インスタンスを作成できます。

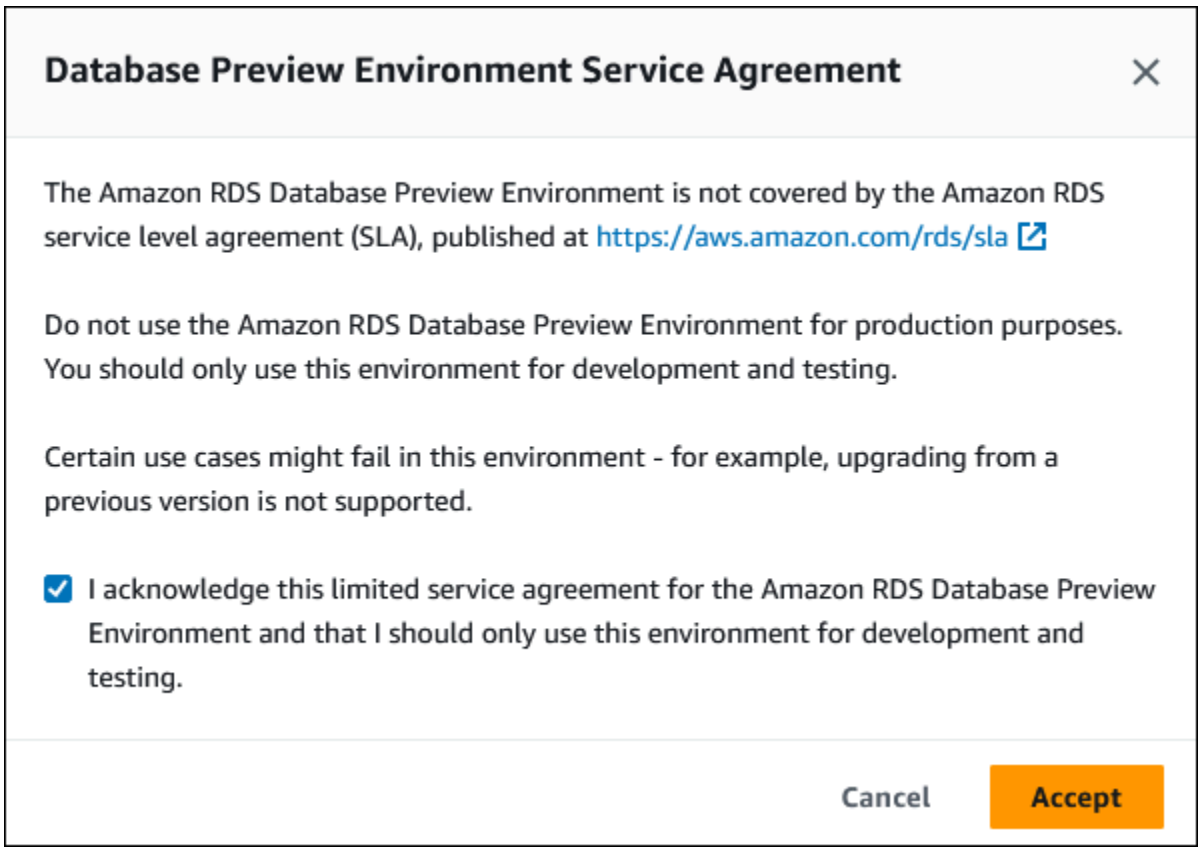
コンソール

データベースプレビュー環境で新しい DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[ダッシュボード] を選択します。
3. [ダッシュボード] ページで、次の図に示すように、[データベースプレビュー環境] セクションを見つけます。



また、[データベースプレビュー環境] に直接移動することもできます。先に進む前に、制限事項を確認して同意する必要があります。



4. RDS for MariaDB DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の コンソール 手順を参照してください。

AWS CLI

AWS CLI を使用してデータベースプレビュー環境でインスタンスを作成するには、次のエンドポイントを使用します。

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MariaDB DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の AWS CLI 手順を参照してください。

RDS API

RDS API を使用してデータベースプレビュー環境でインスタンスを作成するには、次のエンドポイントを使用します。

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MariaDB DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の RDS API 手順を参照してください。

データベースプレビュー環境の MariaDB バージョン 12.3.1

MariaDB バージョン 12.3.1 が Amazon RDS データベースプレビュー環境で利用可能になりました。MariaDB バージョン 12.3.1 には、「Changes and Improvements in MariaDB 12.3」に記載されているいくつかの改善点が含まれています。

データベースプレビュー環境を使用して、本番稼働ワークロードのすべての AWS リージョン で利用できる前に、このリリースに対してワークロードをテストできます。データベースプレビュー環境の詳細については、「the section called “データベースプレビュー環境”」を参照してください。コンソールからプレビュー環境にアクセスするには、rds-preview/ を選択します。

Amazon RDS for MariaDB の非推奨バージョン

Amazon RDS for MariaDB バージョン 10.0、10.1、10.2、10.3、10.4 および 10.5 は非推奨です。

MariaDB の Amazon RDS 廃止ポリシーについては、「Amazon RDS についてのよくある質問」を参照してください。

MariaDB DB インスタンスへの接続

Amazon RDS によって DB インスタンスがプロビジョニングされたら、スタンダードの MariaDB クライアントアプリケーションまたはユーティリティを使用してインスタンスに接続できます。接続文字列で、DB インスタンスのエンドポイントからのドメインネームシステム (DNS) アドレスをホストパラメータとして指定します。また、ポートパラメータとして、DB インスタンスのエンドポイントのポート番号も指定します。

MySQL コマンドラインクライアントなどのツールを使用して、DB インスタンスの Amazon RDS for MariaDB に接続できます。MySQL コマンドラインクライアントの使用についての詳細は、MariaDB ドキュメントの「mysql コマンドラインクライアント」を参照してください。接続に使用できる GUI ベースのアプリケーションの 1 つは Heidi です。詳細については、「HeidiSQL のダウンロード」ページを参照してください。MySQL のインストール (MySQL コマンドラインクライアントを含む) については、「MySQL のインストールと更新」を参照してください。

ほとんどの Linux ディストリビューションには、Oracle MySQL クライアントではなく MariaDB クライアントが含まれています。MariaDB の MySQL コマンドラインクライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install mariadb105
```

MariaDB の MySQL コマンドラインクライアントを Amazon Linux 2 にインストールするには、次のコマンドを実行します。

```
sudo yum install mariadb
```

ほとんどの DEB ベースの Linux ディストリビューションに MySQL コマンドラインクライアントをインストールするには、次のコマンドを実行します。

```
apt-get install mariadb-client
```

MySQL コマンドラインクライアントのバージョンを確認するには、次のコマンドを実行します。

```
mysql --version
```

現在のクライアントバージョン用の MySQL ドキュメントを表示するには、次のコマンドを実行します。

```
man mysql
```

Virtual Private Cloud (VPC) の外部から Amazon VPC に基づいて DB インスタンスに接続するには、DB インスタンスがパブリックにアクセスできる必要があります。また、DB インスタンスのセキュリティグループのインバウンドルールを使用してアクセスを許可し、その他の要件を満たしている必要があります。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

MariaDB DB インスタンスへの接続には、SSL 暗号化を使用できます。詳細については、Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート を参照してください。

RDS for MariaDB DB インスタンスを検索して接続するには、次のトピックを参照してください。

トピック

- MariaDB DB インスタンスの接続情報の検索
- RDS for MariaDB の MySQL コマンドラインクライアントからの接続 (非暗号化)
- AWS JDBC ドライバーと AWS Python ドライバーを使用した RDS for MariaDB への接続
- MariaDB DB インスタンスへの接続のトラブルシューティング

MariaDB DB インスタンスの接続情報の検索

DB インスタンスの接続情報には、エンドポイント、ポート、およびマスターユーザーなどの有効なデータベースユーザーが含まれます。例えば、エンドポイントの値が `mydb.123456789012.us-east-1.rds.amazonaws.com` であるとします。この場合、ポート値は `3306` であり、データベースユーザーは `admin` です。この情報を考慮して、接続文字列に次の値を指定します。

- ホスト、ホスト名または DNS 名には、`mydb.123456789012.us-east-1.rds.amazonaws.com` を指定します。
- ポートで、`3306` を指定します。
- ユーザーには、`admin` を指定します。

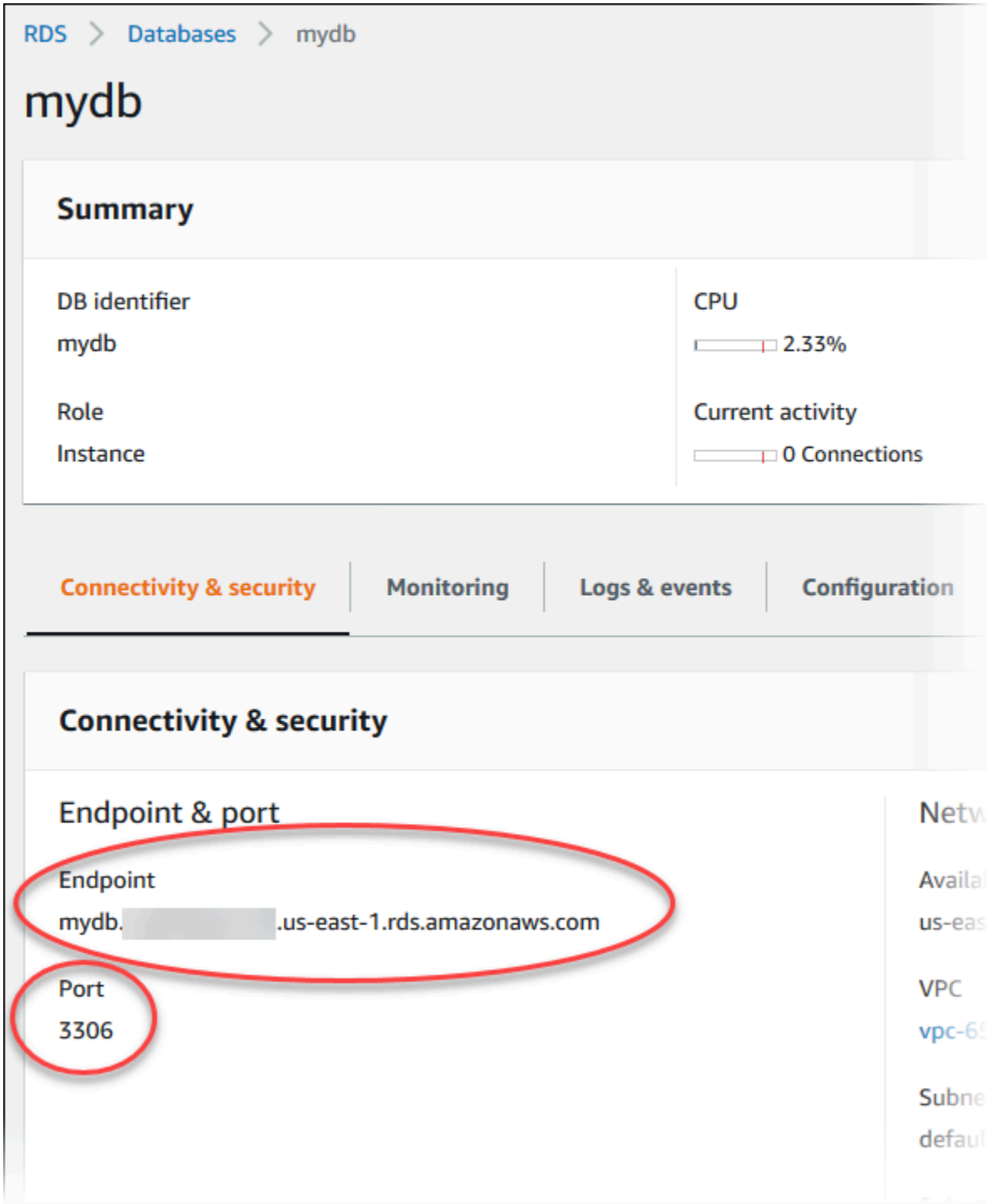
DB インスタンスに接続するには、MariaDB DB エンジンの任意のクライアントを使用します。例えば、MySQL コマンドラインクライアントまたは MySQL ワークベンチを使用できます。

DB インスタンスの接続情報を検索するには、AWS マネジメントコンソール、AWS Command Line Interface (AWS CLI) describe-db-instances コマンド、または Amazon RDS API DescribeDBInstances オペレーションを使用して、詳細を一覧表示できます。

コンソール

AWS マネジメントコンソール で DB インスタンスの接続情報を探すには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択して DB インスタンスのリストを表示します。
3. MariaDB DB インスタンスの名前を選択して、その詳細を表示します。
4. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



5. マスターユーザーネームを見つける必要がある場合は、[設定] タブを選択し、[マスターユーザーネーム] の値を表示します。

AWS CLI

AWS CLI を使用して MariaDB DB インスタンスの接続情報を検索するには、describe-db-instances コマンドを実行します。呼び出しで、DB インスタンス ID、エンドポイント、ポート、マスターユーザー名をクエリします。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
  --filters "Name=engine,Values=mariadb" \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows の場合:

```
aws rds describe-db-instances ^
  --filters "Name=engine,Values=mariadb" ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

出力は次のようになります。

```
[
    [
        "mydb1",
        "mydb1.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "mydb2",
        "mydb2.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ]
]
```

RDS API

Amazon RDS API を使用して DB インスタンスの接続情報を検索するには、DescribeDBInstances オペレーションを呼び出します。出力で、エンドポイントアドレス、エンドポイントポート、およびマスターユーザー名の値を検索します。

RDS for MariaDB の MySQL コマンドラインクライアントからの接続 (非暗号化)

> **Important**
>
> クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない MySQL 接続を使用します。暗号化された接続の使用については、「MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MariaDB DB インスタンスへの接続 (暗号化)」を参照してください。

MySQL コマンドラインクライアントを使用して DB インスタンスに接続するには、クライアントコンピュータのコマンドプロンプトで次のコマンドを入力します。これにより、MariaDB DB インスタンスのデータベースに接続されます。*<endpoint>* の DB インスタンスの DNS 名 (エンドポイント) を、*<mymasteruser>* で使用したマスターユーザー名に置き換えます。パスワードの入力を求められたときに使用したマスターパスワードを入力します。

```
mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

ユーザーのパスワードを入力すると、次のような出力が表示されます。

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.10-MariaDB-log Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

AWS JDBC ドライバーと AWS Python ドライバーを使用した RDS for MariaDB への接続

AWS JDBC ドライバーと AWS Python ドライバーを使用して RDS for MariaDB DB インスタンスに接続します。詳細については、以下のトピックを参照してください。

トピック

- Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for MariaDB への接続
- Amazon Web Services (AWS) Python ドライバーを使用した RDS for MariaDB への接続

Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for MariaDB への接続

Amazon Web Services (AWS) JDBC ドライバーは、高度な JDBC ラッパーとして設計されています。このラッパーは、既存の JDBC ドライバーの機能を補完し、拡張します。ドライバーには、コミュニティ MySQL Connector/J ドライバーおよびコミュニティ MariaDB Connector/J ドライバーとドロップイン互換性があります。

AWS JDBC ドライバーをインストールするには、AWS JDBC ドライバーの .jar ファイル (CLASSPATH アプリケーション内) を追加して、それぞれのコミュニティドライバーへの参照を保持します。対応する接続 URL プレフィックスを次のように更新します。

- `jdbc:mysql://` ~ `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` ~ `jdbc:aws-wrapper:mariadb://`

AWS JDBC ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) JDBC ドライバー GitHub リポジトリ」を参照してください。

Amazon Web Services (AWS) Python ドライバーを使用した RDS for MariaDB への接続

Amazon Web Services (AWS) Python ドライバーは、高度な Python ラッパーとして設計されています。このラッパーは、オープンソースの Psycopg ドライバーの機能を補完し、拡張します。AWS Python ドライバーは Python バージョン 3.8 以降をサポートしています。`aws-advanced-python-wrapper` パッケージは、`pip` コマンドと `psycopg` オープンソースパッケージを使用してインストールできます。

AWS Python ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) Python Driver GitHub repository」を参照してください。

MariaDB DB インスタンスへの接続のトラブルシューティング

新しい DB インスタンスへの接続に失敗する一般的な原因には、次の 2 つがあります。

- MariaDB アプリケーションまたはユーティリティが実行されているデバイスまたは Amazon EC2 インスタンスからの接続を許可しないセキュリティグループを使用して DB インスタンスが作成されました。DB インスタンスには、接続を許可する VPC セキュリティグループが必要です。詳しくは、「Amazon VPC と Amazon RDS」を参照してください。

  セキュリティグループでインバウンドのルールを追加または編集できます。[ソース] には [マイ IP] を選択します。これにより、ブラウザで検出された IP アドレスから DB インスタンスへのアクセスが許可されます。
- DB インスタンスが、デフォルトポートの 3306 を使用して作成されたが、会社のファイアウォールルールで、社内ネットワークのデバイスからそのポートへの接続がブロックされています。この問題を解決するには、別のポートでインスタンスを再起動します。

接続の問題の詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

MariaDB DB インスタンス接続の保護

MariaDB DB インスタンスのセキュリティを管理できます。

トピック

- Amazon RDS での MariaDB のセキュリティ
- RDS for MariaDB のパスワード検証プラグインの使用
- SSL/TLS を使用した Amazon RDS 上の MariaDB DB インスタンスへのクライアント接続の暗号化
- 新しい SSL/TLS 証明書を使用して MariaDB インスタンスに接続するようにアプリケーションを更新する

Amazon RDS での MariaDB のセキュリティ

MariaDB DB インスタンスのセキュリティは以下の 3 つのレベルで管理されます。

- AWS Identity and Access Management では、どのユーザーが DB インスタンスに対して Amazon RDS の管理アクションを実行できるかを制御します。IAM 認証情報を使用して AWS に接続している場合、IAM アカウントには、Amazon RDS の管理オペレーションを実行するためのアクセス許可を付与する IAM ポリシーが必要です。(詳しくは、「Amazon RDS での Identity and Access Management」を参照してください。)
- DB インスタンスを作成するときは、VPC セキュリティグループを使用して、どのデバイスまたは Amazon EC2 インスタンスが DB インスタンスのエンドポイントとポートへの接続を開くことができるかを制御します。これらの接続は、Secure Socket Layer (SSL) と Transport Layer Security (TLS) を使用して行います。さらに、会社のファイアウォールルールでも、社内のどのデバイスが DB インスタンスへの接続を開くことができるかを制御できます。
- MariaDB DB インスタンスへの接続が開かれたら、ログイン認証とアクセス許可は MariaDB のスタンドアロンインスタンスの場合と同じ方法で適用されます。`CREATE USER`、`RENAME USER`、`GRANT`、`REVOKE`、`SET PASSWORD` などのコマンドは、スタンドアロンデータベースの場合と同じ方法で、データベーススキーマテーブルを直接変更します。

Amazon RDS DB インスタンスを作成すると、マスターユーザーには以下のデフォルト権限が付与されます。

- `alter`
- `alter routine`

- create
- create routine
- create temporary tables
- create user
- create view
- delete
- drop
- event
- execute
- grant option
- index
- insert
- lock tables
- process
- references
- reload

  この権限は MariaDB DB インスタンスに限定されます。FLUSH LOGS オペレーションまたは FLUSH TABLES WITH READ LOCK オペレーションにはアクセスを許可しません。
- replication client
- replication slave
- select
- show create routine

  この権限は、バージョン 11.4 以降を実行している MariaDB DB インスタンスに限定されます。
- show databases
- show view
- trigger
- update

これらの権限の詳細については、MariaDB ドキュメントの「User Account Management」を参照してください。

> **Note**
>
> DB インスタンスのマスターユーザーを削除することはできますが、お勧めはしません。マスターユーザーを再作成するには、ModifyDBInstance API または modify-db-instance AWS CLI を使用して、新しいマスターユーザーのパスワードを該当するパラメータで指定します。インスタンスに既存のマスターユーザーがない場合、指定したパスワードを使用してマスターユーザーが作成されます。

各 DB インスタンスに管理サービスを提供するために、DB インスタンスの作成時に rdsadmin ユーザーが作成されます。rdsadmin アカウントの権限をドロップ、名前変更、パスワード変更、または変更しようとするとエラーになります。

RDS for MariaDB バージョン 12.3 以降は、rdsproxyadmin も予約済みユーザーです。Amazon RDS は、DB インスタンスをプロキシのターゲットとして初めて登録するときに、このユーザーを作成します。任意のホストで rdsproxyadmin アカウントを作成、削除、名前変更、または変更しようとすると、次のようなエラーが発生します。

```
mysql> DROP USER 'rdsproxyadmin'@'%';
ERROR 1396 (HY000): Operation DROP USER failed for 'rdsproxyadmin'@'%'

mysql> DROP USER 'rdsproxyadmin'@'host';
ERROR 1396 (HY000): Operation DROP USER failed for 'rdsproxyadmin'@'host'
```

RDS Proxy モニタリングユーザーの詳細については、「Amazon RDS Proxy」を参照してください。

Amazon RDS がレプリケーションに使用する rdsrepladmin ユーザーも、予約済みユーザーです。RDS for MariaDB バージョン 12.3 以降では、ホストにかかわらずこのアカウントを削除することはできません。

```
mysql> DROP USER 'rdsrepladmin'@'host';
ERROR 1396 (HY000): Operation DROP USER failed for 'rdsrepladmin'@'host'
```

DB インスタンスの管理を可能にするために、スタンダード的なコマンド kill と kill_query の使用は制限されています。Amazon RDS コマンド mysql.rds_kill、mysql.rds_kill_query、mysql.rds_kill_query_id が MariaDB と MySQL 用に用意されており、DB インスタンスのユーザーセッションやクエリを終了できます。

RDS for MariaDB のパスワード検証プラグインの使用

RDS for MariaDB バージョン 11.4 以降では、以下のパスワード検証プラグインを使用して、データベース接続のセキュリティを強化できます。

- simple_password_check – パスワードに特定のタイプの文字が少なくとも特定の数だけ含まれているかどうかを確認します。
- cracklib_password_check – パスワードが CrackLib ライブラリの辞書ファイルに表示されるかどうかを確認します。

これらのプラグインを有効にするには、DB インスタンスに関連付けられた DB パラメータグループでパラメータ `simple_password_check` または `cracklib_password_check` の値を `FORCE_PLUS_PERMANENT` に設定します。この値を設定すると、ランタイムに `UNINSTALL PLUGIN` ステートメントを使用してプラグインをアンインストールすることはできません。

これらのプラグインを無効にするには、DB インスタンスに関連付けられた DB パラメータグループでパラメータ `simple_password_check` または `cracklib_password_check` の値を `OFF` に設定します。この値を設定すると、プラグイン検証ルールは新しいパスワードには適用されません。

パラメータグループ内のパラメータの値の設定については、「the section called “DB パラメータグループのパラメータの変更”」を参照してください。

プラグインを有効にしたら、新しい検証ポリシーに準拠するように、既存のパスワードをリセットします。

MariaDB DB インスタンスは、Amazon RDS のパスワード検証を処理します。パスワードを変更するには、まず AWS マネジメントコンソール、`modify-db-instance` AWS CLI コマンド、または `ModifyDBInstance` API オペレーションを使用してパスワード更新リクエストを送信します。パスワードがポリシーを満たしていなくても、Amazon RDS は最初にリクエストを受け入れます。その後、Amazon RDS はリクエストを非同期的に処理します。MariaDB DB インスタンスのパスワードは、パスワードが定義されたポリシーを満たしている場合にのみ更新されます。パスワードがこれらのポリシーを満たさない場合、Amazon RDS は既存のパスワードを保持し、エラーイベントをログに記録します。

```
   Unable to reset your password. Error information: Password failed to meet
validation rules.
```

Amazon RDS イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。

SSL/TLS を使用した Amazon RDS 上の MariaDB DB インスタンスへのクライアント接続の暗号化

Secure Sockets Layer (SSL) は、クライアントとサーバー間のネットワーク接続を安全に保つための業界標準のプロトコルです。SSL バージョン 3.0 以降では、名前が Transport Layer Security (TLS) に変更されています。Amazon RDS は、MariaDB DB インスタンス向けに SSL/TLS での暗号化をサポートしています。SSL/TLS を使用して、アプリケーションクライアントと MariaDB インスタンス間の接続を暗号化できます。SSL/TLS サポートは、すべての AWS リージョン で提供されています。

Amazon RDS を使用すると、SSL/TLS を使用して MariaDB DB インスタンスへのクライアント接続を暗号化し、MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求し、MySQL コマンドラインクライアントから SSL/TLS (暗号化) を使用して接続することで、転送中のデータを保護できます。以下のセクションでは、Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS 暗号化の設定と使用に関するガイダンスを提供します。

トピック

- Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート
- Amazon RDS の MariaDB DB インスタンスへの特定のユーザーアカウントに SSL/TLS を要求する
- Amazon RDS 上の MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求する
- MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MariaDB DB インスタンスへの接続 (暗号化)

Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート

Amazon RDS によって、Amazon RDS によるインスタンスのプロビジョニング時、SSL/TLS 証明書が作成され、DB インスタンスにインストールされます。これらの証明書は認証局によって署名されます。SSL/TLS 証明書には、なりすまし攻撃から保護するために、SSL/TLS 証明書の共通名 (CN) として DB インスタンスのエンドポイントが含まれています。

Amazon RDS によって作成された SSL/TLS 証明書は信頼されたルートエンティティであり、ほとんどの場合は使用できますが、アプリケーションが証明書チェーンを受け入れていない場合は使用できない可能性があります。アプリケーションが証明書チェーンを受け入れていない場合は、中間

証明書を使用して AWS リージョンに接続してみてください。例えば、SSL/TLS を使用して AWS GovCloud (US) リージョンに接続するには、中間証明書を使用する必要があります。

証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。MySQL での SSL/TLS の使用の詳細については、「新しい SSL/TLS 証明書を使用して MariaDB インスタンスに接続するようにアプリケーションを更新する」を参照してください。

Amazon RDS for MariaDB は、Transport Layer Security (TLS) バージョン 1.3、1.2、1.1、および 1.0 をサポートしています。TLS のサポートは MariaDB のバージョンによって異なります。次の表は、MariaDB バージョンの TLS サポートを示しています。

TLS バージョン	MariaDB 10.11、11.4、11.8、12.3	MariaDB 10.6	MariaDB 10.5	MariaDB 10.4
TLS 1.3	すべてのマイナーバージョン	すべてのマイナーバージョン	すべてのマイナーバージョン	すべてのマイナーバージョン
TLS 1.2	すべてのマイナーバージョン	すべてのマイナーバージョン	すべてのマイナーバージョン	すべてのマイナーバージョン
TLS 1.1	サポートされていません	10.6.16 以前	10.5.23 以前	10.4.32 以前
TLS 1.0	サポートされていません	10.6.16 以前	10.5.23 以前	10.4.32 以前

Amazon RDS の MariaDB DB インスタンスへの特定のユーザーアカウントに SSL/TLS を要求する

Amazon RDS 上の MariaDB DB インスタンスへの特定のユーザーアカウント接続に SSL/TLS 暗号化を要求できます。機密情報を不正アクセスや傍受から保護することは、データの機密性が懸念される場合にセキュリティポリシーを適用する上で非常に重要です。

特定のユーザーのアカウントに SSL/TLS 接続を要求するには、MySQL のバージョンに応じて、次のいずれかのステートメントを使用して、ユーザーアカウントの `encrypted_user` に SSL/TLS 接続を要求します。

これを行うには、以下のステートメントを実行します。

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

MariaDB での SSL/TLS 接続の詳細については、MariaDB ドキュメントの「Securing Connections for Client and Server」(クライアントとサーバーの接続の保護) を参照してください。

Amazon RDS 上の MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求する

MariaDB DB インスタンスへのすべてのユーザー接続が SSL/TLS を使用することを、require_secure_transport パラメータを使用して要求します。バージョン 11.4 以前では、require_secure_transport パラメータはデフォルトで OFF に設定されています。11.8 以降のバージョンでは、デフォルト値は ON に設定され、DB インスタンスへの接続に SSL/TLS が適用されます。非セキュア接続が必要な場合は、require_secure_transport パラメータを OFF に変更できます。

> **Note**
>
> require_secure_transport パラメーターは、MariaDB バージョン 10.5 以降でのみサポートされます。

require_secure_transport パラメータの値は、DB インスタンスの DB パラメータグループを更新することで設定できます。変更を有効にするために、DB インスタンスを再起動する必要はありません。

DB インスタンスに対して require_secure_transport パラメータが ON に設定されている場合、データベースクライアントが暗号化された接続を確立できれば、データベースクライアントはそのクラスターに接続できます。それ以外の場合は、次のような エラーメッセージがクライアントに返されます。

```
ERROR 1045 (28000): Access denied for user 'USER'@'localhost' (using password: YES |
 NO)
```

パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

require_secure_transport パラメータの詳細については、MMariaDB のドキュメントを参照してください。

MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MariaDB DB インスタンスへの接続 (暗号化)

MySQL 5.7 バージョン、MySQL 8.0 バージョン、または MariaDB バージョンを使用している場合は、mysql クライアントプログラムのパラメータが若干異なります。

使用しているバージョンを確認するには、--version オプションを指定しながら mysql コマンドを実行します。次の例の出力では、MariaDB のクライアントプログラムが使用されていることが確認できます。

```
$ mysql --version
mysql  Ver 15.1 Distrib 10.5.15-MariaDB, for osx10.15 (x86_64) using readline 5.1
```

Amazon Linux、CentOS、SUSE、Debianなど、ほとんどのLinux ディストリビューションでは、MySQL をMariaDB に置き換えており、mysql バージョンは MariaDB と同じものを使用しています。

SSL/TLS を使用して DB インスタンスに接続するには、以下のステップを実行します。

MySQL コマンドラインクライアントを使用して SSL/TLS で DB インスタンスに接続するには

1. すべての AWS リージョン で使用できるルート証明書をダウンロードします。

   証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

2. MySQL コマンドラインクライアントを使用して、SSL/TLS 暗号化を使用して DB インスタンスに接続します。-h パラメータは、DB インスタンスの DNS 名 (エンドポイント) に置き換えます。--ssl-ca パラメータは、必要に応じて SSL/TLS 証明書のファイル名に置き換えます。-P パラメータは、使用中の DB インスタンスのポートに置き換えます。-u パラメータでは、マスターユーザーなどの有効なデータベースユーザーのユーザー名に置き換えます。プロンプトが表示されたら、マスターユーザーパスワードを入力します。

   次の例は、MariaDB で --ssl-ca パラメータを使用しながら、クライアントを起動する方法を示しています。

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl -P 3306 -u myadmin -p
```

SSL/TLS 接続で DB インスタンスのエンドポイントを SSL/TLS 証明書のエンドポイントと照合することを義務付けるには、次のコマンドを入力します。

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-verify-server-cert -P 3306 -u myadmin -p
```

次の例は、MySQL 5.7 以降で `--ssl-ca` パラメータを使用しながら、クライアントを起動する方法を示しています。

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=REQUIRED -P 3306 -u myadmin -p
```

3. プロンプトが表示されたら、マスターユーザーパスワードを入力します。

次のような出力が表示されます。

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.10-MariaDB-log Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

新しい SSL/TLS 証明書を使用して MariaDB インスタンスに接続するようにアプリケーションを更新する

2023 年 1 月 13 日に Amazon RDS は、Secure Socket Layer または Transport Layer Security (SSL/TLS) を使用して RDS DB インスタンスに接続するための新しい認証局 (CA) 証明書を公開しました。ここでは、新しい証明書を使用するためのアプリケーションの更新について説明します。

このトピックでは、アプリケーションが DB インスタンスに接続するために証明書認証が必要かどうかを判断できます。

> **Note**
>
> 一部のアプリケーションは、サーバー上の証明書を正常に検証できる場合にのみ、MariaDB に接続されるように設定されています。そのようなアプリケーションの場合は、新しい CA 証明書を含むように、クライアントアプリケーションの信頼ストアを更新する必要があります。
>
> `disabled`、`preferred`、および `required` の SSL モードを指定できます。`preferred` SSL モードを使用し、CA 証明書が存在しないか、最新でない場合、接続は SSL を使用しない状態にフォールバックし、正常に接続します。
>
> `preferred` モードの使用を避けることをお勧めします。`preferred` モードでは、接続時に無効な証明書が検出されると、暗号化の使用が停止し、暗号化せずに続行されます。

クライアントアプリケーションの信頼ストアで CA 証明書を更新した後、DB インスタンスで証明書をローテーションできます。これらの手順を開発環境またはステージング環境でテストしてから、本番環境で実装することを強くお勧めします。

証明書のローテーションの詳細については、「SSL/TLS 証明書のローテーション」を参照してください。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。MariaDB DB インスタンスで SSL/TLS を使用する方法については、「Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート」を参照してください。

トピック

- クライアントが接続するために証明書の検証を必要とするかどうかの確認
- アプリケーション信頼ストアの更新
- SSL 接続を確立するための Java コードの例

クライアントが接続するために証明書の検証を必要とするかどうかの確認

JDBC クライアントおよび MySQL クライアントを接続するために証明書の検証が必要かどうかを確認できます。

JDBC

MySQL Connector/J 8.0 を使用した次の例では、アプリケーションの JDBC 接続プロパティをチェックして、正常な接続に有効な証明書が必要かどうかを確認する 1 つの方法を示しま

す。MySQL のすべての JDBC 接続オプションの詳細については、MySQL ドキュメントの「設定プロパティ」を参照してください。

MySQL Connector/J 8.0 を使用するとき、次の例のように、接続プロパティで sslMode が VERIFY_CA または VERIFY_IDENTITY に設定されている場合、SSL 接続にはサーバー CA 証明書に対する検証が必要です。

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> MySQL Java Connector v5.1.38 以降または MySQL Java Connector v8.0.9 以降を使用してデータベースに接続する場合、データベースへの接続時に SSL/TLS を使用するようにアプリケーションを明示的に設定しなくても、これらのクライアントドライバーはデフォルトで SSL/TLS を使用します。また、SSL/TLS の使用時に証明書検証が部分的に実行され、データベースサーバー証明書の有効期限が切れていると、接続に失敗します。
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

MySQL

MySQL クライアントの以下の例では、スクリプトの MySQL 接続をチェックして、正常な接続に有効な証明書が必要かどうかを確認する 2 つの方法を示します。MySQL クライアントで使用するすべての接続オプションの詳細については、MySQL ドキュメントの「Client-Side Configuration for Encrypted Connections」を参照してください。

MySQL 5.7 または MySQL 8.0 クライアントを使用するとき、次の例のように、--ssl-mode オプションに VERIFY_CA または VERIFY_IDENTITY を指定する場合、SSL 接続にはサーバー CA 証明書に対する検証が必要です。

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

MySQL 5.6 クライアントを使用するとき、次の例のように、--ssl-verify-server-cert オプションを指定する場合、SSL 接続にはサーバー CA 証明書に対する検証が必要です。

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

アプリケーション信頼ストアの更新

MySQL アプリケーションの信頼ストアの更新の詳細については、MariaDB ドキュメントの「Using TLS/SSL with MariaDB Connector/Jを参照してください。

ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

証明書をインポートするサンプルスクリプトについては、証明書を信頼ストアにインポートするためのサンプルスクリプト を参照してください。

> **Note**
>
> 信頼ストアを更新するとき、新しい証明書を追加できるだけでなく、古い証明書を保持できます。

アプリケーションで MariaDB コネクタ/J JDBC ドライバーを使用している場合は、アプリケーションで以下のプロパティを設定します。

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

アプリケーションを起動するとき、以下のプロパティを設定します。

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに表示されているプロンプト以外のパスワードを指定してください。

SSL 接続を確立するための Java コードの例

次のコード例は、JDBC を使用する SSL 接続のセットアップ方法を示します。

```
private static final String DB_USER = "admin";

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void main(String[] args) throws Exception {
        Class.forName("org.mariadb.jdbc.Driver");

        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);


        Connection connection = DriverManager.getConnection("jdbc:mysql://ssl-mariadb-
public.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306?useSSL=true",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
```

```
}
```

Important

データベース接続で SSL/TLS を使用することを決定し、アプリケーションの信頼ストアを更新したら、rds-ca-rsa2048-g1 証明書を使用するようにデータベースを更新できます。ステップについては、「DB インスタンスまたはクラスターの変更による CA 証明書の更新」のステップ 3 を参照してください。
セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上

Amazon RDS Optimized Reads で RDS for MariaDB のクエリ処理を高速化することができます。RDS Optimized Reads を使用する RDS for MariaDB DB インスタンスは、これを使用しない DB インスタンスに比べて、クエリ処理を最大 2 倍高速化できます。

トピック

- RDS Optimized Reads の概要
- RDS Optimized Reads のユースケース
- RDS Optimized Reads のベストプラクティス
- RDS Optimized Reads の使用
- RDS Optimized Reads を使用する DB インスタンスのモニタリング
- RDS Optimized Reads についての制限事項

RDS Optimized Reads の概要

RDS Optimized Reads が有効になっている RDS for MariaDB DB インスタンスを使用する場合、DB インスタンスはインスタンスストアを使用することで、より高速なクエリパフォーマンスを達成します。インスタンスストアは、DB インスタンスに一時ブロックレベルのストレージを提供します。ストレージは、ホストサーバーに物理的にアタッチされた不揮発性メモリエクスプレス (NVMe) によるソリッドステートドライブ (SSD) にあります。このストレージは、低レイテンシー、優れたランダム I/O パフォーマンス、高いシーケンシャル読み取りスループットを実現するために最適化されています。

DB インスタンスが db.m5d や db.m6gd などのインスタンスストアを含む DB インスタンスクラスを使用する場合、RDS Optimized Reads はデフォルトで有効になっています。RDS Optimized Reads では、一部の一時オブジェクトがインスタンスストアに保存されます。これらの一時オブジェクトには、内部一時ファイル、内部オンディスク一時テーブル、メモリマップファイル、バイナリログ (binlog) キャッシュファイルが含まれます。インスタンスストアの詳細については、Linux インスタンス向け Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 インスタンスストア」を参照してください。

MariaDB でクエリ処理用の一時オブジェクトを生成するワークロードは、インスタンスストアを利用してクエリ処理を高速化できます。このタイプのワークロードには、ソート、ハッシュ集約、高負荷結合、共通テーブル式 (CTE)、インデックス付けされていない列に対するクエリが含まれます。

これらのインスタンスストアボリュームによって、永続的な Amazon EBS ストレージに使用されるストレージ設定に関係なく、より高い IOPS とパフォーマンスを提供します。RDS Optimized Reads は一時オブジェクトのオペレーションをインスタンスストアにオフロードするため、永続的ストレージ (Amazon EBS) の 1 秒あたりの入出力オペレーション (IOPS) またはスループットを、永続オブジェクトのオペレーションに使用できるようになりました。これらのオペレーションには、通常のデータファイルの読み取りと書き込み、フラッシュやバッファ挿入のマージなどのバックグラウンドエンジンオペレーションが含まれます。

> **Note**
>
> 手動の RDS スナップショットと自動の RDS スナップショットの両方に含まれているのは、永続オブジェクトのエンジンファイルのみです。インスタンスストアで作成された一時オブジェクトは RDS スナップショットには含まれません。

RDS Optimized Reads のユースケース

クエリの実行を内部テーブルやファイルなどの一時オブジェクトに大きく依存するワークロードがある場合は、RDS Optimized Reads を有効にすると便利です。RDS Optimized Reads の候補となるユースケースは次のとおりです。

- 複雑なテーブル共通式 (CTE)、派生テーブル、グループ化オペレーションを使用して分析クエリを実行するアプリケーション
- 最適化されていないクエリで大量の読み取りトラフィックを処理するリードレプリカ
- `GROUP BY` 句や `ORDER BY` 句を含むクエリなど、複雑なオペレーションを伴うオンデマンドまたは動的レポートクエリを実行するアプリケーション
- クエリ処理に内部テンポラリテーブルを使用するワークロード

  エンジンステータス変数 `created_tmp_disk_tables` をモニタリングすることで、DB インスタンスに作成されたディスクベースの一時テーブルの数を判断できます。
- 中間結果を格納するために、直接またはプロシージャ内で大規模な一時テーブルを作成するアプリケーション
- インデックス付けされていない列をグループ化または順序付けするデータベースクエリ

RDS Optimized Reads のベストプラクティス

RDS Optimized Reads を使用するベストプラクティスは次のとおりです。

- インスタンスストアが実行中にストレージ不足によって失敗した場合に備えて、読み取り専用クエリの再試行ロジックを追加します。
- CloudWatch メトリクスの `FreeLocalStorage` を使用して、インスタンスストアで使用可能なストレージ容量をモニタリングします。DB インスタンスのワークロードが原因でインスタンスストアが上限に達している場合は、より大きな DB インスタンスクラスを使用するように DB インスタンスを変更します。
- DB インスタンスに十分なメモリがあるのにインスタンスストアのストレージ制限に達している場合は、`binlog_cache_size` 値を増やしてセッション固有のバイナリログエントリをメモリに保持します。この設定により、ディスク上の一時バイナリログキャッシュファイルにバイナリログエントリの書き込みを防止します。

  `binlog_cache_size` パラメータはセッション固有です。この値は、新しいセッションごとに変更できます。このパラメータを設定すると、ピーク負荷時に DB インスタンスのメモリ使用量が増加する可能性があります。そのため、アプリケーションのワークロードのパターンと DB インスタンスで使用可能なメモリに基づいてパラメータ値を増やすことを検討してください。
- `binlog_format` には `MIXED` のデフォルト値を使用します。トランザクションのサイズによっては、`binlog_format` を `ROW` に設定すると、インスタンスストアのバイナリログキャッシュファイルが大きくなる可能性があります。
- 1 つのトランザクションで大量の変更を実行することは避けてください。このような種類のトランザクションでは、インスタンスストアに大きなバイナリログキャッシュファイルが生成され、インスタンスストアが満杯になると問題が発生する可能性があります。バイナリログキャッシュファイルのストレージ使用量を最小限に抑えるために、書き込みを複数の小さなトランザクションに分割することを検討してください。

RDS Optimized Reads の使用

シングル AZ DB インスタンスデプロイまたはマルチ AZ DB インスタンスデプロイで、次の DB インスタンスクラスのいずれかを使用して RDS for MariaDB DB インスタンスをプロビジョニングすると、DB インスタンスは自動的に RDS Optimized Reads を使用します。

RDS Optimized Reads をオンにするには、次のいずれかの操作を行います。

- これらの DB インスタンスクラスの 1 つを使用して、RDS for MariaDB DB インスタンスを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

- これらの DB インスタンスクラスの 1 つを使用して、既存の RDS for MariaDB DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS Optimized Reads は、ローカル NVMe SSD ストレージのある DB インスタンスクラスの 1 つ以上がサポートされているすべての AWS リージョン で使用できます。DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

DB インスタンスクラスの可用性は AWS リージョン によって異なります。DB インスタンスクラスが特定の AWS リージョン でサポートされているかどうかを判断するには、「the section called “AWS リージョン での DB インスタンスクラスのサポートを決定する”」を参照してください。

RDS Optimized Reads を使用しない場合は、この機能をサポートする DB インスタンスクラスを使用しないように DB インスタンスを変更してください。

RDS Optimized Reads を使用する DB インスタンスのモニタリング

RDS Optimized Reads を使用する DB インスタンスは、次の CloudWatch メトリクスでモニタリングできます。

- `FreeLocalStorage`
- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`
- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

これらのメトリクスでは、利用可能なインスタンスストアストレージ、IOPS、スループットに関するデータを提供します。これらのメトリクスの詳細については、「」を参照してください。Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス

RDS Optimized Reads についての制限事項

RDS Optimized Reads には次の制限事項が適用されます。

- RDS Optimized Reads は、以下の RDS for MariaDB バージョンでサポートされています。

- 11.4 以降のメジャーリリースで利用可能なすべてのマイナーバージョン
- 10.11.4 以降の 10.11 バージョン
- 10.6.7 以上の 10.6 バージョン
- 10.5.16 以上の 10.5 バージョン
- 10.4.25 以上の 10.4 バージョン

RDS for MariaDB のバージョンの詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。

- RDS Optimized Reads をサポートする DB インスタンスクラスでは、一時オブジェクトの場所を永続ストレージ (Amazon EBS) に変更することはできません。
- DB インスタンスでバイナリロギングが有効になっている場合、最大トランザクションサイズはインスタンスストアのサイズによって制限されます。MariaDB では、`binlog_cache_size` の値よりも多くのストレージを必要とするセッションでは、インスタンスストアに作成される一時的なバイナリログキャッシュファイルにトランザクションの変更が書き込まれます。
- トランザクションは、インスタンスストアが満杯になるとエラーになる可能性があります。

Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上

RDS Optimized Writes for MariaDB によって、書き込みトランザクションのパフォーマンスを向上させることができます。RDS for MariaDB データベースで RDS Optimized Writes を使用すると、書き込みトランザクションのスループットが最大 2 倍向上します。

トピック

- RDS Optimized Writes の概要
- RDS Optimized Writes の使用
- 既存のデータベースで RDS Optimized Writes を有効にする
- RDS Optimized Writes についての制限事項

RDS Optimized Writes の概要

RDS Optimized Writes を有効にすると、RDS for MariaDB データベースは、データを耐久性のあるストレージにフラッシュするときに、二重書き込みバッファを必要とせずに一度だけ書き込みます。データベースは、ACID プロパティ保護機能の提供を継続することで、信頼性の高いデータベーストランザクションを実現するとともに、パフォーマンスを向上させます。

MariaDB のようなリレーショナルデータベースには、原子性、一貫性、分離性、耐久性という ACID 特性があり、信頼性の高いデータベーストランザクションを実現できます。これらの特性を発揮するために、MariaDB は部分的なページ書き込みエラーを防ぐ二重書き込みバッファと呼ばれるデータストレージ領域を使用しています。これらのエラーは、データベースがページを更新中に、停電時などのハードウェア障害が発生した場合に発生します。MariaDB データベースは、部分的なページ書き込みを検出し、二重書き込みバッファ内のページのコピーによって回復できます。この手法で保護が可能ですが、余分な書き込みオペレーションも必要になります。MariaDB の二重書き込みバッファの詳細については、MariaDB ドキュメントの「InnoDB Doublewrite Buffer」(InnoDB 二重書き込みバッファ) を参照してください。

RDS Optimized Writes を有効にすると、RDS for MariaDB データベースは、データを耐久性のあるストレージにフラッシュするときに、二重書き込みバッファを使用せずに一度だけ書き込みます。RDS Optimized Writes は、RDS for MariaDB データベースで書き込みの多いワークロードを実行する場合に便利です。書き込みワークロードの高いデータベースの例としては、デジタル決済、金融取引、ゲームアプリケーションに対応したデータベースなどがあります。

これらのデータベースは、AWS Nitro System を使用する DB インスタンスクラスで実行されます。これらのシステムのハードウェア構成により、データベースは 16 KiB ページを 1 ステップで確実かつ永続的にデータファイルに直接書き込むことができます。AWS Nitro System により RDS Optimized Writes が可能になります。

新しいデータベースパラメータ `rds.optimized_writes` を設定して、RDS for MariaDB データベースの RDS Optimized Writes 機能を制御できます。RDS for MariaDB の場合、次のバージョンの DB パラメータグループのこのパラメータにアクセスします。

- 11.8 以降のメジャーリリースで利用可能なすべてのマイナーバージョン
- 14.4.3 以降の 11.4 バージョン
- 10.11.4 以降の 10.11 バージョン
- 10.6.10 以降の 10.6 バージョン

次の値を使用してパラメータを設定します。

- `AUTO` – データベースが RDS Optimized Writes をサポートしている場合は有効にします。データベースが RDS Optimized Writes をサポートしていない場合は無効にします。この設定はデフォルトです。
- `OFF` – データベースが RDS Optimized Writes をサポートしている場合でも無効にします。

RDS Optimized Writes を使用するように設定されている RDS for MariaDB データベースを、この機能をサポートしていない DB インスタンスクラスに移行した場合、RDS はそのデータベースの RDS Optimized Writes を自動的に無効にします。

RDS Optimized Writes を無効にすると、データベースは MariaDB の二重書き込みバッファを使用します。

RDS for MariaDB データベースが RDS Optimized Writes を使用しているかどうかを判断するには、データベースの `innodb_doublewrite` パラメータの現在の値を表示します。データベースが RDS Optimized Writes を使用している場合、このパラメータは `FALSE` (`0`) に設定されています。

RDS Optimized Writes の使用

RDS コンソール、AWS CLI、または RDS API を使用して RDS for MariaDB データベースを作成すると、RDS Optimized Writes を有効にできます。RDS Optimized Writes は、データベース作成時に、次の条件に両方当てはまる場合、自動的に有効になります。

- RDS Optimized Writes をサポートする DB エンジンのバージョンと DB インスタンスクラスを指定します。
  - RDS Optimized Writes は、以下の RDS for MariaDB バージョンでサポートされています。
    - 11.8 以降のメジャーリリースで利用可能なすべてのマイナーバージョン
    - 14.4.3 以降の 11.4 バージョン
    - 10.11.4 以降の 10.11 バージョン
    - 10.6.10 以降の 10.6 バージョン

    RDS for MariaDB のバージョンの詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。
  - RDS Optimized Writes は、次の DB インスタンスクラスを使用する RDS for MariaDB データベースでサポートされています。
    - db.m7i
    - db.m7g
    - db.m6g
    - db.m6gd
    - db.m6i
    - db.m5
    - db.m5d
    - db.r7i
    - db.r7g
    - db.r6g
    - db.r6gd
    - db.r6i
    - db.r5
    - db.r5b
    - db.r5d
    - db.x2idn
    - db.x2iedn

    DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

DB インスタンスクラスの可用性は AWS リージョン によって異なります。DB インスタンスクラスが特定の AWS リージョン でサポートされているかどうかを判断するには、「the section called “AWS リージョン での DB インスタンスクラスのサポートを決定する”」を参照してください。

- データベースに関連付けられたパラメータグループでは、`rds.optimized_writes` パラメータが `AUTO` に設定されます。デフォルトのパラメータグループでは、このパラメータは常に `AUTO` に設定されます。

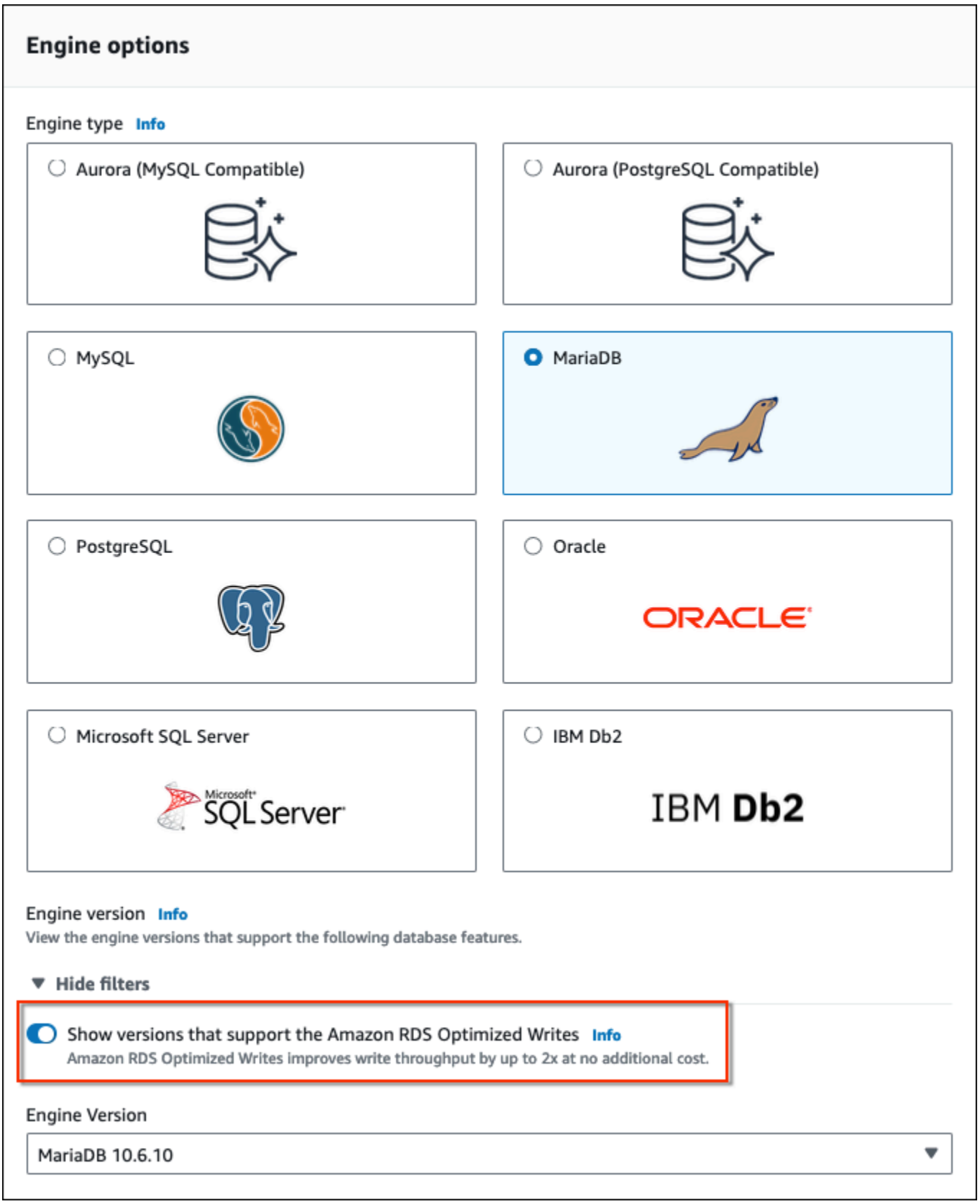
RDS Optimized Writes をサポートする DB エンジンバージョンと DB インスタンスクラスを使用し、この機能は使用しない場合は、データベースの作成時に、カスタムパラメータグループを指定します。このパラメータグループで、`rds.optimized_writes` パラメータを `OFF` に設定します。その後、データベースで RDS Optimized Writes を使用するには、パラメータを `AUTO` に設定して有効にすることができます。カスタムパラメータグループの作成とパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

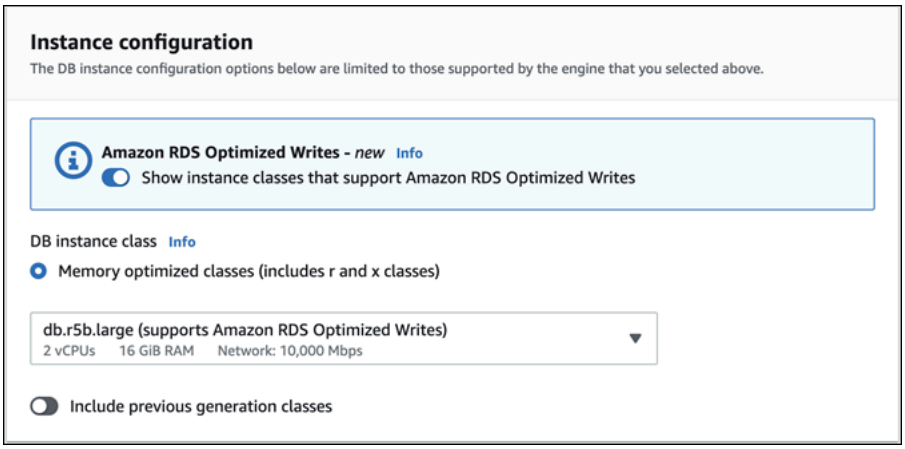
コンソール

RDS コンソールを使用して RDS for MariaDB データベースを作成すると、RDS Optimized Writes をサポートする DB エンジンバージョンと DB インスタンスクラスをフィルタリングできます。フィルタリングを有効にすると、利用可能な DB エンジンのバージョンと DB インスタンスクラスから選択できるようになります。

RDS Optimized Writes をサポートする DB エンジンバージョンを選択するには、エンジンバージョンをサポートする RDS for MariaDB DB のエンジンバージョンをフィルタリングして、バージョンを選択します。


Engine options
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
ORACLE
Microsoft SQL Server
Microsoft SQL Server
IBM Db2
IBM Db2
Engine version Info
View the engine versions that support the following database features.
Hide filters
Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Engine Version
MariaDB 10.6.10

[インスタンスの設定] セクションで、RDS Optimized Writes をサポートする DB インスタンスクラスをフィルタリングして、DB インスタンスクラスを選択します。



これらの選択を行った後、要件を満たす他の設定を選択し、コンソールで RDS for MariaDB データベースの作成を完了します。

AWS CLI

AWS CLI を使用して DB インスタンスを作成するには、create-db-instance コマンドを実行します。`--engine-version` と `--db-instance-class` の値が RDS Optimized Writes をサポートしていることを確認してください。また、DB インスタンスに関連付けられたパラメータグループでは、`rds.optimized_writes` パラメータが`AUTO` に設定されていることを確認します。この例では、デフォルトパラメータグループを DB インスタンスに関連付けます。

Example RDS Optimized Writes を使用する DB インスタンスの作成

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mariadb \
    --engine-version 10.6.10 \
    --db-instance-class db.r5b.large \
    --manage-master-user-password \
    --master-username admin \
    --allocated-storage 200
```

Windows の場合:

```
aws rds create-db-instance ^
```

```
    --db-instance-identifier mydbinstance ^
    --engine mariadb ^
    --engine-version 10.6.10 ^
    --db-instance-class db.r5b.large ^
    --manage-master-user-password ^
    --master-username admin ^
    --allocated-storage 200
```

RDS API

CreateDBInstance オペレーションを使用して、DB インスタンスを作成できます。このオペレーションを使用する際は、EngineVersion と DBInstanceClass の値が RDS Optimized Writes をサポートしていることを確認してください。また、DB インスタンスに関連付けられたパラメータグループでは、rds.optimized_writes パラメータがAUTO0 に設定されていることを確認します。

既存のデータベースで RDS Optimized Writes を有効にする

RDS Optimized Writes を有効にするために既存の RDS for MariaDB データベースを変更するには、サポート対象の DB エンジンバージョンと DB インスタンスクラスでデータベースを作成する必要があります。さらに、必要な基盤となるファイルシステム設定は、リリース前に作成されたデータベースの設定と互換性がないため、データベースは 2023 年 3 月 7 日に RDS Optimized Writes がリリースされた後に作成されたものでなければなりません。これらの条件が満たされている場合は、rds.optimized_writes パラメータを AUTO0 に設定して RDS Optimized Writes を有効にできます。

データベースが、サポートされているエンジンバージョン、インスタンスクラス、またはファイルシステム設定で作成されていない場合、RDS ブルー/グリーンデプロイを使用して、サポートされている設定に移行できます。ブルー/グリーンデプロイを作成して、以下の操作を行います。

- [グリーンデータベースで Optimized Writes を有効にする] を選択し、RDS Optimized Writes をサポートするエンジンバージョンと DB インスタンスクラスを指定します。サポートされているエンジンバージョンとインスタンスクラスのリストについては、「the section called “新しいデータベースでの使用”」を参照してください。
- [ストレージ] で、[ストレージファイルシステム設定のアップグレード] を選択します。このオプションは、データベースを互換性のある基本ファイルシステム設定にアップグレードします。

ブルー/グリーンデプロイを作成すると、rds.optimized_writes パラメータが AUTO0 に設定されている場合、RDS Optimized Writes がグリーン環境で自動的に有効になります。その後、ブルー/グリーンデプロイを切り替えて、グリーン環境を新しい本番環境に昇格できます。

詳細については、「the section called “ブルー/グリーンデプロイの作成”」を参照してください。

RDS Optimized Writes についての制限事項

RDS for MariaDB データベースをスナップショットから復元する場合、次の条件がすべて当てはまる場合のみ、データベースの RDS Optimized Writes を有効にできます。

- スナップショットが RDS Optimized Writes をサポートするデータベースから作成された。
- スナップショットは、RDS Optimized Writes のリリース後に作成されたデータベースから作成されました。
- スナップショットが RDS Optimized Writes をサポートするデータベースに復元された。
- 復元されたデータベースは、AUTO に設定された `rds.optimized_writes` パラメータに関連付けられています。

MariaDB DB エンジンのアップグレード

新しいバージョンのデータベースエンジンが Amazon RDS でサポートされている場合は、DB インスタンスをその新しいバージョンにアップグレードできます。MariaDB DB インスタンスのアップグレードには、メジャーバージョンのアップグレードとマイナーバージョンのアップグレードの 2 種類あります。

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、DB インスタンスのメジャーバージョンアップグレードは手動で実行する必要があります。メジャーバージョンアップグレードをスタートするには、DB インスタンスを変更します。ただし、メジャーバージョンのアップグレードを行う前に、「RDS for MariaDB のメジャーバージョンアップグレード」の手順を実行することをお勧めします。

それに対して、マイナーバージョンのアップグレードに含まれるのは、既存のアプリケーションとの下位互換性がある変更のみです。マイナーバージョンのアップグレードを手動でスタートするには、DB インスタンスを変更します。または、DB インスタンスの作成時または変更時に、[Auto minor version upgrade (マイナーバージョン自動アップグレード)] を有効にすることができます。これにより、Amazon RDS によって新しいバージョンがテストおよび承認されると、DB インスタンスが自動的にアップグレードされます。アップグレードの実行については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

ご使用の MariaDB DB インスタンスでリードレプリケーションを使用している場合は、ソースインスタンスのアップグレード前に、すべてのリードレプリカをアップグレードする必要があります。DB インスタンスがマルチ AZ 配置にある場合、プライマリとスタンバイのレプリカの両方がアップグレードされます。アップグレードが完了するまで、DB インスタンスは使用できない場合があります。

MariaDB でサポートされるバージョンおよびバージョン管理については、「Amazon RDS の MariaDB のバージョン」を参照してください。

データベースエンジンをアップグレードするには、ダウンタイムが必要です。ダウンタイムの時間は、DB インスタンスのサイズによって異なります。

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

> **Tip**
>
> ブルー/グリーンデプロイを使用することで、DB インスタンスのアップグレードに必要なダウンタイムを最小限に抑えることができます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

トピック

- MariaDB のアップグレードに関する考慮事項
- 有効なアップグレードターゲットの検索
- MariaDB のバージョン番号
- RDS for MariaDB の RDS バージョン番号
- RDS for MariaDB のメジャーバージョンアップグレード
- MariaDB DB インスタンスのアップグレード
- RDS for MariaDB のマイナーバージョンの自動アップグレード
- RDS for MariaDB データベースをアップグレードするときにリードレプリカを使用してダウンタイムを短縮する
- イベントによる RDS for MariaDB DB エンジンのアップグレードのモニタリング

MariaDB のアップグレードに関する考慮事項

Amazon RDS によってアップグレードプロセス中に 2 つ以上の DB スナップショットが作成されます。Amazon RDS は、アップグレードを変更する前に DB インスタンスのスナップショットを最大 2 つ作成します。アップグレードがデータベースに対して機能しない場合は、これらのスナップショットの 1 つを復元して、以前のバージョンを実行する DB インスタンスを作成できます。Amazon RDS は、アップグレードが完了すると、DB インスタンスのもう 1 つのスナップショットを作成します。Amazon RDS は、AWS Backup が DB インスタンスのバックアップを管理するかどうかにかかわらず、これらのスナップショットを作成します。

> **Note**
>
> DB インスタンスのバックアップ保持期間を 0 より大きく設定した場合にのみ、Amazon RDS は DB スナップショットを作成します。バックアップ保持期間を変更するには、「Amazon RDS DB インスタンスを変更する」を参照してください。

アップグレードが完了したら、データベースエンジンの前のバージョンに戻すことはできません。前のバージョンに戻す必要がある場合は、作成された初期の DB スナップショットを復元して、新しい DB インスタンスを作成します。

Amazon RDS でサポートされている新しいバージョンに DB インスタンスをアップグレードするタイミングを制御します。このレベルの管理によって、特定のデータベースのバージョンとの互換性を維持しながら、新しいバージョンを本稼働環境にデプロイする前にアプリケーションに対してテストできます。準備が整ったら、自分のスケジュールに最適なタイミングでバージョンアップグレードを実行できます。

ご使用の DB インスタンスでリードレプリカを使用している場合は、ソースインスタンスのアップグレード前に、すべてのリードレプリカをアップグレードする必要があります。

DB インスタンスがマルチ AZ 配置にある場合は、プライマリとスタンバイの両方の DB インスタンスがアップグレードされます。プライマリとスタンバイの DB インスタンスは同時にアップグレードされ、アップグレードが完了するまで停止します。停止時間は、データベースエンジン、エンジンのバージョン、DB インスタンスのサイズによって異なります。

有効なアップグレードターゲットの検索

AWS マネジメントコンソール を使用して DB インスタンスをアップグレードする場合、DB インスタンスの有効なアップグレードターゲットが表示されます。次の AWS CLI コマンドを実行して、DB インスタンスの有効なアップグレードターゲットを特定することもできます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
  --engine mariadb \
  --engine-version version_number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine mariadb ^
  --engine-version version_number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

例えば、MariaDB バージョン 10.5.15 DB インスタンスの有効なアップグレードターゲットを特定するには、次の AWS CLI コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
  --engine mariadb \
  --engine-version 10.5.17 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine mariadb ^
  --engine-version 10.5.17 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

MariaDB のバージョン番号

RDS for MariaDB データベースエンジンのバージョン番号付けシーケンスは、major.minor.patch.YYYYMMDD または major.minor.patch の形式です。例えば、10.11.5.R2.20231201 または 10.4.30 です。使用される形式は、MariaDB エンジンのバージョンによって異なります。

major

メジャーバージョン番号は、10.11 など、バージョン番号の整数部分と 1 つ目の小数部分の両方です。メジャーバージョンのアップグレードでは、バージョン番号の主要な部分が大きくなります。例えば、10.5.20 から 10.6.12 へのアップグレードはメジャーバージョンのアップグレードであり、10.5 と 10.6 はメジャーバージョン番号です。

minor

マイナーバージョン番号は、バージョン番号の 3 番目と 4 番目の部分です。例えば、10.11.5 の 5 です。

patch

パッチはバージョン番号の 4 番目の部分です。例えば、10.11.5.R2 の R2 です。RDS パッチバージョンには、リリース後にマイナーバージョンに追加された重要なバグ修正が含まれています。

YYYYMMDD

日付はバージョン番号の 5 番目の部分です。例えば、10.11.5.R2.20231201 の 20231201 です。RDS 日付バージョンは、リリース後にマイナーバージョンに追加された重要なセキュリティ修正を含むセキュリティパッチです。エンジンの動作を変更する可能性のある修正は含まれていません。

次の表に、RDS for MariaDB バージョン 10.11 の命名スキームを示します。

10.11 マイナーバージョン	命名スキーム
≥5	新しい DB インスタンスでは、major.minor.patch.YYYMMDD を使用します。例えば、10.11.5.R2.20231201。 既存の DB インスタンスは、次のメジャーバージョンまたはマイナーバージョンのアップグレードまで、10.11.5.R2 など、major.minor.patch を使用する場合があります。
< 5	既存の DB インスタンスは、major.minor.patch を使用します。例えば、10.11.4.R2。

次の表に、RDS for MariaDB バージョン 10.6 の命名スキームを示します。

10.6 マイナーバージョン	命名スキーム
≥ 14	新しい DB インスタンスは、major.minor.patch.YYYMMDD を使用します。例えば、10.6.14.R2.20231201。 既存の DB インスタンスは、次のメジャーまたはマイナーバージョンのアップグレードまで、10.6.14.R2 など、major.minor.patch を使用する場合があります。
< 14	既存の DB インスタンスは、major.minor.patch を使用します。例えば、10.6.13.R2。

次の表に、RDS for MariaDB バージョン 10.5 の命名スキームを示します。

10.5 マイナーバージョン	命名スキーム
≥ 21	新しい DB インスタンスは、major.minor.patch.YYYMMDD を使用します。例えば、10.5.21.R2.20231201。 既存の DB インスタンスは、次のメジャーまたはマイナーバージョンのアップグレードまで、10.5.21.R2 など、major.minor.patch を使用する場合があります。
< 21	既存の DB インスタンスは、major.minor.patch を使用します。例えば、10.5.20.R2。

次の表に、RDS for MariaDB バージョン 10.4 の命名スキームを示します。

10.4 マイナーバージョン	命名スキーム
≥ 30	新しい DB インスタンスは、major.minor.patch.YYMMDD を使用します。例えば、10.4.30.R2.20231201。 既存の DB インスタンスは、次のメジャーまたはマイナーバージョンのアップグレードまで、10.4.30.R2 など、major.minor.patch を使用する場合があります。
< 30	既存の DB インスタンスは、major.minor.patch を使用します。例えば、10.4.29.R2。

RDS for MariaDB の RDS バージョン番号

RDS バージョン番号は *major*.*minor*.*patch* または *major*.*minor*.*patch*.*YYYYMMDD* 命名規則を使用します。RDS パッチバージョンには、リリース後にマイナーバージョンに追加された重要なバグ修正が含まれています。RDS 日付バージョン (*YYMMDD*) はセキュリティパッチです。セキュリティパッチには、エンジンの動作を変更する可能性のある修正は含まれていません。

データベースの Amazon RDS バージョン番号を識別するには、まず次のコマンドを使用して `rds_tools` 拡張機能を作成する必要があります。

```
CREATE EXTENSION rds_tools;
```

RDS for MariaDB データベースの RDS バージョン番号は、次の SQL クエリで確認できます。

```
mysql> select mysql.rds_version();
```

例えば、RDS for MariaDB 10.6.14 データベースの RDS をクエリすると、次が出力が返されます。

```
+---------------------+
| mysql.rds_version() |
+---------------------+
| 10.6.14.R2.20231201  |
+---------------------+
1 row in set (0.01 sec)
```

RDS for MariaDB のメジャーバージョンアップグレード

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、Amazon RDS では、メジャーバージョンアップグレードは自動的に適用されません。DB インスタンスを手動で変更する必要があります。本稼働インスタンスへの適用前に、いずれのアップグレードも徹底的にテストすることをお勧めします。

> **Note**
>
> MariaDB 11.8 では、`require_secure_transport` のデフォルト値は `1` になり、安全な SSL/TLS 接続が必要です。非セキュア接続が必要な場合は、`0` に設定します。

Amazon RDS では、MariaDB データベースエンジンのメジャーバージョンに対して、以下のインプレースアップグレードをサポートしています。

- MariaDB の任意バージョンから MariaDB 12.3 へ
- MariaDB の任意バージョンから MariaDB 11.8 へ
- MariaDB の任意バージョンから MariaDB 11.4 へ
- MariaDB の任意バージョンから MariaDB 10.11 へ
- MariaDB の任意バージョンから MariaDB 10.6 へ
- MariaDB 10.4 から MariaDB 10.5 へ

カスタムパラメータグループを使用しており、メジャーバージョンアップグレードを実行する場合には、新しい DB エンジンバージョンのデフォルトのパラメータグループを指定するか、新しい DB エ

ンジンバージョンの独自のカスタムパラメータグループを作成する必要があります。新しいパラメータグループを DB インスタンスと関連付けるには、アップグレードの完了後に、顧客主導型のデータベースの再起動が必要となります。パラメータグループの変更を適用するためにインスタンスを再起動する必要がある場合には、インスタンスのパラメータグループのステータスとして `pending-reboot` が表示されます。インスタンスのパラメータグループのステータスは、AWS マネジメントコンソール で表示するか、`describe-db-instances` などの「describe」呼び出しを実行して表示することができます。

MariaDB DB インスタンスのアップグレード

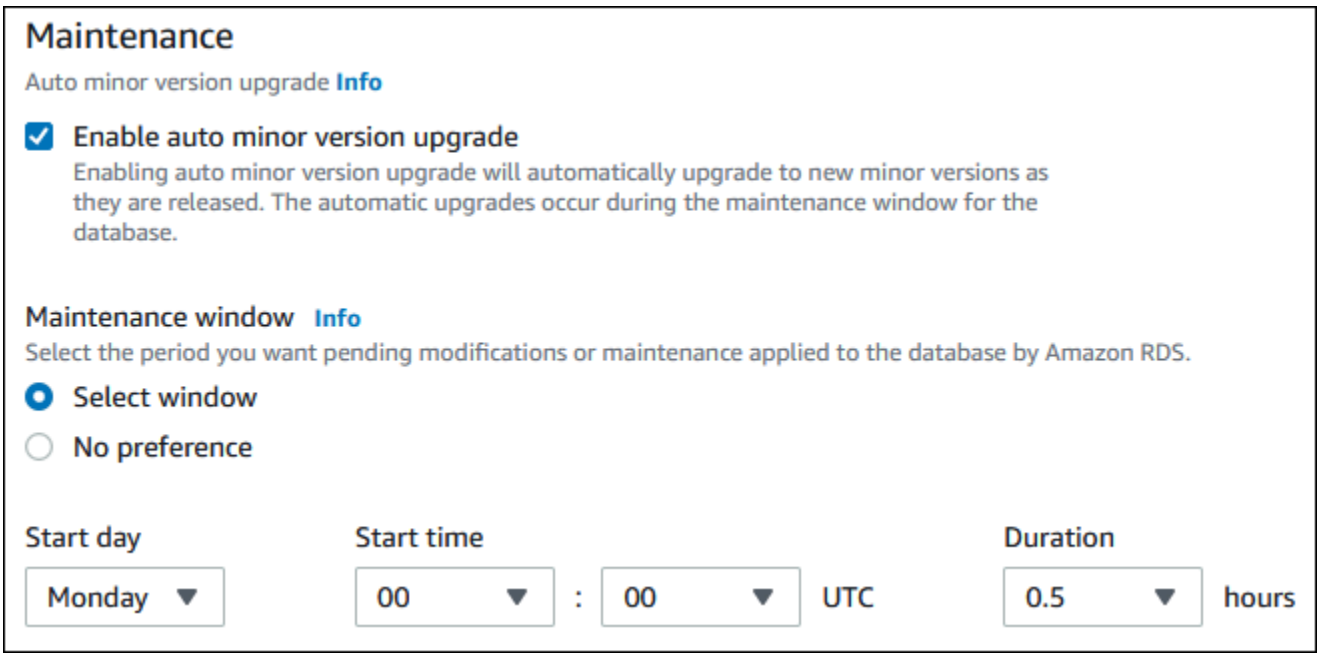
MariaDB DB インスタンスの手動または自動アップグレードについては、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

RDS for MariaDB のマイナーバージョンの自動アップグレード

DB インスタンスの作成または変更時に次の設定を指定すると、DB インスタンスを自動的にアップグレードできます。

- [マイナーバージョンの自動アップグレード] の設定は有効です。
- バックアップ保持期間の設定は 0 より大きいです。

AWS マネジメントコンソールの場合、これらの設定はAAdditional configuration (追加設定) の下にあります。下図は、Auto minor version upgrade (自動マイナーバージョンアップグレード) 設定を示しています。



これらの設定の詳細については、「DB インスタンスの設定」をご参照ください。

一部の AWS リージョン の RDS for MariaDB メジャーバージョンでは、RDS によって 1 つのマイナーバージョンが自動アップグレードバージョンとして指定されます。Amazon RDS でマイナーバージョンのテストと承認が完了すると、メンテナンスウィンドウの間にマイナーバージョンアップグレードが自動的に行われます。RDS では、新しくリリースされたマイナーバージョンが自動アップグレードバージョンとして自動的に設定されることはありません。RDS によって新しい自動アップグレードバージョンが指定される前に、以下のような複数の基準が考慮されます。

- 既知のセキュリティの問題
- MariaDB コミュニティバージョンのバグ
- マイナーバージョンがリリースされてからのフリート全体の安定性

> **Note**
>
> TLS バージョン 1.0 と 1.1 の使用のサポートは、MariaDB の特定のマイナーバージョン以降では終了しています。サポートしている MariaDB のマイナーバージョンの詳細については、「the section called “MariaDB の SSL/TLS サポート”」参照してください。

次の AWS CLI コマンドを実行して、特定の AWS リージョン で指定された MariaDB マイナーバージョンの現在の自動マイナーアップグレードターゲットバージョンを確認できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine mariadb \
--engine-version minor_version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine mariadb ^
--engine-version minor_version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

例えば、次の AWS CLI コマンドにより、米国東部 (オハイオ) AWS リージョン (us-east-2) の MariaDB マイナーバージョン 10.5.12 の自動マイナーアップグレードターゲットを特定できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine mariadb \
--engine-version 10.5.16 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine mariadb ^
--engine-version 10.5.16 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

以下のような出力が生成されます。

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  True        |  10.5.17        |
|  False       |  10.5.18        |
|  False       |  10.5.19        |
|  False       |  10.6.5         |
|  False       |  10.6.7         |
|  False       |  10.6.8         |
|  False       |  10.6.10        |
|  False       |  10.6.11        |
|  False       |  10.6.12        |
+--------------+-----------------+
```

この例では、AutoUpgrade 値は、MariaDB バージョン 10.5.17 の場合、True です。したがって、自動マイナーアップグレードターゲットは MariaDB バージョン 10.5.17 であり、出力で強調表示されています。

MariaDB DB インスタンスは、以下の基準を満たしている場合、メンテナンスウィンドウの間に自動的にアップグレードされます。

- [マイナーバージョンの自動アップグレード] の設定は有効です。
- バックアップ保持期間の設定は 0 より大きいです。
- DB インスタンスでは、現在の自動アップグレードマイナーバージョン未満の DB エンジンのマイナーバージョンが実行されています。

詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。

RDS for MariaDB データベースをアップグレードするときにリードレプリカを使用してダウンタイムを短縮する

ほとんどの場合、MariaDB DB インスタンスをアップグレードする際のダウンタイムを減らすには、ブルー/グリーンデプロイが最適なオプションです。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

ブルー/グリーンデプロイを使用できず、MariaDB DB インスタンスが現在本稼働アプリケーションで使用されている場合は、次の手順を使用して DB インスタンスのデータベースバージョンをアップグレードできます。この手順により、アプリケーションの停止時間を短縮できます。

リードレプリカを使用して、ほとんどのメンテナンスステップを事前に実行し、実際の停止中に必要な変更を最小限に抑えることができます。この方法で、既存の DB インスタンスを変更せずに、新しい DB インスタンスのテストおよび準備ができます。

次の手順は、MariaDB バージョン 10.5 から MariaDB バージョン 10.6 へのアップグレードの例を示しています。他のメジャーバージョンへのアップグレードに、この同じ一般的なステップを使用できます。

DB インスタンスの使用中に MariaDB データベースをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. MariaDB 10.5 DB インスタンスのリードレプリカを作成します。このプロセスにより、データベースのアップグレード可能なコピーが作成されます。DB インスタンスの他のリードレプリカも存在する場合があります。

   a. コンソールで [データベース] を選択し、アップグレードする DB インスタンスを選択します。

   b. [アクション] で [リードレプリカの作成] を選択します。

   c. リードレプリカの [DB instance identifier] (DB インスタンス識別子) の値を指定し、[DB instance class] (DB インスタンスクラス) とその他の設定がご使用の MariaDB 10.5 DB インスタンスのものと一致していることを確認します。

   d. [Create read replica] を選択します。
3. (オプション) リードレプリカが作成され、ステータスが [利用可能] になったら、リードレプリカをマルチ AZ 配置に変換し、バックアップを有効にします。

   デフォルトでは、リードレプリカは、バックアップが無効なシングル AZ デプロイとして作成されます。リードレプリカは最終的に本番 DB インスタンスになるため、ベストプラクティスは、マルチ AZ デプロイを設定し、すぐにバックアップを有効にすることです。

   a. コンソールで [データベース] を選択し、作成したばかりのリードレプリカを選択します。

   b. Modify を選択します。

   c. マルチ AZ 配置で、[スタンバイインスタンスの作成] を選択します。

d. [Backup Retention Period] (バックアップ保持期間) として、0 以外の正の値 (3 日など) を選択し、[Continue] (続行) を選択します。

e. [変更のスケジューリング] で、[すぐに適用] を選択します。

f. [DB インスタンスの変更] を選択します。

4. リードレプリカの [Status] (ステータス) が [Available] (使用可能) になったら、リードレプリカを MariaDB 10.6 にアップグレードします。

a. コンソールで [データベース] を選択し、作成したばかりのリードレプリカを選択します。

b. Modify を選択します。

c. [DB engine version] (DB エンジンバージョン) として、アップグレードする MariaDB 10.6 バージョンを選択し、[Continue] (続行) を選択します。

d. [変更のスケジューリング] で、[すぐに適用] を選択します。

e. [Modify DB instance] を選択してアップグレードをスタートします。

5. アップグレードが完了し、[Status] (ステータス) が [Available] (利用可能) になったら、アップグレードしたリードレプリカがソース MariaDB 10.5 DB インスタンスで最新であることを確認します。確認するには、リードレプリカに接続して、`SHOW REPLICA STATUS` コマンドを実行します。`Seconds_Behind_Master` フィールドが `0` である場合、レプリカは更新されています。

> **Note**
>
> MariaDB の旧バージョンは、`SHOW SLAVE STATUS` ではなく `SHOW REPLICA STATUS` を使用していました。10.6 より前の MariaDB バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

6. (オプション) リードレプリカのリードレプリカを作成します。

DB インスタンスをスタンドアロン DB インスタンスに昇格した後、リードレプリカを追加する場合、すぐにリードレプリカを作成できます。

a. コンソールで [データベース] を選択し、アップグレードしたばかりのリードレプリカを選択します。

b. [アクション] で [リードレプリカの作成] を選択します。

c. リードレプリカの [DB instance identifier] (DB インスタンス識別子) の値を指定し、[DB instance class] (DB インスタンスクラス) とその他の設定がご使用の MariaDB 10.5 DB インスタンスのものと一致していることを確認します。

d. [Create read replica] を選択します。

7. (オプション) リードレプリカのカスタム DB パラメータグループを設定します。

   DB インスタンスをスタンドアロン DB インスタンスに昇格した後、カスタムパラメータグループを使用する場合、すぐにDB パラメータグループを作成して、リードレプリカに関連付けられます。

   a. MariaDB 10.6 のカスタム DB パラメータグループを作成します。手順については、Amazon RDS での DB パラメータグループの作成 を参照してください。

   b. 作成したばかりの DB パラメータグループで変更するパラメータを変更します。手順については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

   c. コンソールで [データベース] を選択し、リードレプリカを選択します。

   d. Modify を選択します。

   e. [DB parameter group] (DB パラメータグループ) で、作成したばかりの MariaDB 10.6 DB パラメータグループを選択し、[Continue] (続行) を選択します。

   f. [変更のスケジューリング] で、[すぐに適用] を選択します。

   g. [Modify DB instance] を選択してアップグレードをスタートします。

8. MariaDB 10.6 リードレプリカをスタンドアロン DB インスタンスにします。

   > **Important**
   >
   > MariaDB 10.6 のリードレプリカをスタンドアロン DB インスタンスに昇格すると、MariaDB 10.5 DB インスタンスのレプリカではなくなります。MariaDB 10.6 のリードレプリカの昇格は、ソースの MariaDB 10.5 DB インスタンスが読み取り専用モードであり、すべての書き込みオペレーションが停止されているメンテナンスウィンドウ中に行うことをお勧めします。昇格が完了したら、アップグレードした MariaDB 10.6 DB インスタンスに書き込み操作を送信して、書き込み操作が失われないようにできます。
   > また、MariaDB 10.6 のリードレプリカを昇格する前に、必要なすべてのデータ定義言語 (DDL) のオペレーションを MariaDB 10.6 のリードレプリカに対して実行することをお勧めします。例えば、インデックスの作成があります。これにより、昇格後の MariaDB 10.6 のリードレプリカのパフォーマンスへの悪影響を避けることができます。リードレプリカを昇格させるには、次の手順に従います。

a. コンソールで [データベース] を選択し、アップグレードしたばかりのリードレプリカを選択します。
b. [アクション] で、[Promote (昇格)] を選択します。
c. [はい] を選択して、リードレプリカインスタンスの自動バックアップを有効にします。詳細については、「バックアップの概要」を参照してください。
d. [Continue] (続行) をクリックします。
e. [Promote Read Replica] を選択します。

9. これで、MariaDB データベースのアップグレードバージョンが作成されました。この時点で、アプリケーションを新しい MariaDB 10.6 DB インスタンスに送信できます。

イベントによる RDS for MariaDB DB エンジンのアップグレードのモニタリング

RDS for MariaDB データベースのエンジンバージョンをアップグレードすると、Amazon RDS はプロセスの各フェーズで特定のイベントを発行します。アップグレードの進行状況を追跡するには、これらのイベントを表示またはサブスクライブします。

RDS イベントの詳細については、「Amazon RDS イベントのモニタリング」を参照してください。

エンジンのアップグレード中に発生する特定の Amazon RDS イベントの詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

MariaDB DB スナップショットエンジンバージョンのアップグレード

Amazon RDS を使用すると、MariaDB DB インスタンスのストレージボリュームの DB スナップショットを作成できます。DB スナップショットを作成するとき、スナップショットは DB インスタンスが使用するエンジンバージョンに基づきます。DB スナップショットのエンジンバージョンをアップグレードできます。

RDS for MariaDB では、使用可能なすべてのエンジンバージョンにアップグレードできます。暗号化されている DB スナップショットまたは暗号化されていない DB スナップショットをアップグレードできます。

RDS for MariaDB DB スナップショットで使用可能なエンジンバージョンを表示するには、次の AWS CLI 例を使用します。

```
aws rds describe-db-engine-versions --engine mariadb --include-all --engine-
version example-engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text
```

スナップショットの結果が表示されない場合は、エンジンのバージョンが非推奨である可能性があります。エンジンのバージョンが非推奨の場合は、最新のメジャーバージョンのアップグレードターゲットにアップグレードするか、そのバージョンで利用可能な他のアップグレードターゲットにアップグレードすることをお勧めします。詳細については、「RDS for MariaDB でサポートされていないエンジンバージョンの DB スナップショットのアップグレードオプション」を参照してください。

新しいエンジンバージョンにアップグレードした DB スナップショットをリストアしたら、アップグレードに問題がないか必ずテストしてください。主なバージョンアップグレードの詳細は、the section called “MariaDB DB エンジンのアップグレード” を参照してください。DB スナップショットをリストアする方法については the section called “DB インスタンスへの復元” をご覧ください。

> **Note**
>
> 自動バックアッププロセス中に作成された自動 DB スナップショットをアップグレードすることはできません。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB スナップショットをアップグレードできます。

Console

AWS マネジメントコンソールを使用して DB スナップショットエンジンのバージョンをアップグレードするには、次の手順を使用します。

DB スナップショットをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. アップグレードしたいスナップショットを選択します。
4. [アクション] は、[スナップショットのアップグレード] を選択します。[スナップショットのアップグレード] ページが表示されます。
5. アップグレードする [新しいエンジンバージョン] を選択します。
6. [変更を保存] を選択してスナップショットをアップグレードします。

   アップグレード中は、この DB スナップショットに関するスナップショット操作が、すべて無効になります。また、DB スナップショットのステータスは、利用可能からアップグレード中に変わり、プロセスが完了するとアクティブに変わります。スナップショットの破損問題で DB スナップショットをアップグレードできない場合、ステータスは使用不可になります。この状態からスナップショットを復元することはできません。

   > **Note**
   >
   > DB スナップショットアップグレードが失敗した場合、スナップショットは元の状態にロールバックします。

AWS CLI

DB スナップショットを新しいバージョンのデータベースエンジンにアップグレードするには、AWS CLI の modify-db-snapshot コマンドを実行します。

オプション

- `--db-snapshot-identifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

- `--engine-version` – DB スナップショットをこのエンジンバージョンにアップグレードする。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot \

    --db-snapshot-identifier my_db_snapshot \
    --engine-version new_version
```

Windows の場合:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

Amazon RDS API

DB スナップショットを新しいデータベースエンジンバージョンにアップグレードするには、RDS API の ModifyDBSnapshot オペレーションを呼び出します。

パラメータ

- `DBSnapshotIdentifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。
- `EngineVersion` – DB スナップショットをこのエンジンバージョンにアップグレードする。

RDS for MariaDB でサポートされていないエンジンバージョンの DB スナップショットのアップグレードオプション

次の表は、RDS for MariaDB DB スナップショットでサポートされていないエンジンバージョンからアップグレードできるエンジンバージョンを示しています。

> **Note**
>
> 選択したエンジンバージョンにアップグレードするには、DB スナップショットを複数回アップグレードする必要がある場合があります。

DB スナップショットエンジンのバージョン	アップグレード可能なエンジンバージョン
10.0.17、10.0.24、10.0.28、10.0.31、10.0.32、10.0.34、10.0.35	10.0.38、10.1.48、10.6.19、10.6.20、10.11.9、10.11.10、11.4.3、11.4.4、11.8.3、12.3.2
10.1.14、10.1.16、10.1.19、10.1.23、10.1.26、10.1.31、10.1.34	10.1.48、10.2.44、10.6.19、10.6.20、10.11.9、10.11.10、11.4.3、11.4.4、11.8.3、12.3.2
10.2.11、10.2.12、10.2.15、10.2.21、10.2.32、10.2.37、10.2.39、10.2.40、10.2.41、10.2.43	10.2.44、10.3.39、10.6.19、10.6.20、10.6.21、10.11.9、10.11.10、10.11.11、11.4.3、11.4.4、11.8.3、12.3.2
10.3.8、10.3.13、10.3.20、10.3.23、10.3.28、10.3.31、10.3.32	10.3.39、10.4.34、10.6.19、10.6.20、10.6.21、10.11.9、10.11.10、10.11.11、11.4.3、11.4.4、11.8.3、12.3.2
10.4.8	10.4.34、10.5.16、10.5.17、10.5.18、10.5.20、10.5.21、10.5.22、10.5.23、10.5.24、10.5.25、10.5.26、10.5.27、10.5.28、10.6.8、10.6.10、10.6.11、10.6.13、10.6.14、10.6.15、10.6.16、10.6.18、10.6.19、10.6.20、10.6.21、10.11.4、10.6.17、10.11.5、10.11.6、10.11.7、10.11.8、10.11.9、10.11.10、10.11.11、11.4.3、11.4.4、11.8.3、12.3.2

Amazon RDS for MariaDB DB インスタンスへのデータのインポート

さまざまな手法を使用して、RDS for MariaDB DB インスタンスにデータをインポートできます。最善のアプローチは、いくつかの要因によって異なります。

- データのソース
- データ量
- 1 回限りのインポートか継続的なインポートか
- ダウンタイムの長さ

データと同時にアプリケーションも移行する場合は、ダウンタイムの長さを考慮することが重要です。

次の表は、RDS for MariaDB DB インスタンスにデータをインポートする手法の一覧です。

> **Note**
>
> Amazon RDS は、Amazon S3 for MariaDB の `mariadb-backup` またはインポートをサポートしていません。

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
オンプレミスまたは Amazon EC2 の既存の MariaDB	いずれか	継続的	Minimal	レプリケーション元として既存の MariaDB データベースを使用してレプリケーションを設定します。 MariaDB DB インスタンスへのレプリケーションを設定するには、外部インスタンスが MariaDB バージョン 10.0.24 以	外部のソースインスタンスを使用したバイナ

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
データベース				降の場合は MariaDB グローバルトランザクション識別子 (GTID) を使用し、10.0.24 より前のバージョンでは MariaDB インスタンスのバイナリログ座標を使用できます。MariaDB GTID の実装方法は、Amazon RDS ではサポートされていない MySQL GTID とは異なります。	リログファイル位置のレプリケーションの設定 ダウンタイムを短縮して Amazon RDS for MariaDB DB インスタンスにデータをインポートする

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存のデータベース	すべて	1 回または進行中	最小限	AWS Database Migration Service を使用してダウンタイムを最小限にしてデータベースを移行し、多くのデータベース DB エンジンで継続的なレプリケーションを継続します。	「AWS Database Migration Service ユーザーガイド」の「AWS Database Migration Service とは?」および「MySQL 互換データベースの AWS DMS のターゲットとしての使用」

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存の MariaDB DB インスタンス	いずれか	1 回または進行中	最小限	継続的なレプリケーション用のリードレプリカを作成します。新しい DB インスタンスの 1 回限りの作成用のリードレプリカを昇格させます。	DB インスタンスのリードレプリカの操作
既存の MariaDB データベース	スモール	1 回	ある程度	コマンドラインユーティリティを使用して、MariaDB DB インスタンスに直接データをコピーします。	外部の MariaDB データベースから Amazon RDS for MariaDB DB インスタンスにデータをインポートする

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存のデータベースに保存されないデータ	ミディアム	1 回	ある程度	フラットファイルを作成し、MariaDB の `LOAD DATA LOCAL INFILE` ステートメントを使用してインポートします。	任意のソースから Amazon RDS for MariaDB DB インスタンスにデータをインポートする

> **Note**
>
> `mysql` システムデータベースには、DB インスタンスへのログインとデータへのアクセスに必要な認証と認可の情報が含まれています。DB インスタンスにある `mysql` データベースのテーブル、データ、その他のコンテンツを削除、変更、名前変更、または切り捨てを行うとエラーが発生し、DB インスタンスやデータにアクセスできなくなる場合があります。この場合は、AWS CLI の restore-db-instance-from-db-snapshot コマンドを使用して、スナップショットから DB インスタンスを復元できます。DB インスタンスは、restore-db-instance-to-point-in-time コマンドを使用して復元できます。

MariaDB のデータインポートに関する考慮事項

以下のコンテンツには、MariaDB へのデータのロードに関連する技術情報が含まれています。このコンテンツは、MariaDB サーバーアーキテクチャに精通しているユーザーを対象としています。

バイナリログ記録

バイナリログ記録を有効にすると、データロードのパフォーマンスが低下し、ログを無効にした場合と比較して最大 4 倍の追加のディスク領域が必要になります。データのロードに使用されるトランザクションサイズは、システムのパフォーマンスとディスク容量のニーズに直接影響します。トランザクションが大きいほど、より多くのリソースが必要になります。

トランザクションサイズ

トランザクションサイズは、MariaDB データロードの以下の側面に影響します。

- リソース消費
- ディスクスペースの使用率
- プロセスの再開
- 復旧時間
- 入力形式 (フラットファイルまたは SQL)

このセクションでは、トランザクションサイズがバイナリログ作成に与える影響について説明し、大量のデータロード中にバイナリログ作成を無効にするケースを作成します。Amazon RDS 自動バックアップ保持期間を設定することで、バイナリログ作成を有効または無効にすることができます。値が 0 以外の場合はバイナリログ作成が有効になり、0 の場合は無効になります。詳細については、「バックアップの保存期間」を参照してください。

このセクションでは、大規模なトランザクションが InnoDB に与える影響と、トランザクションサイズを小さく保つことが重要な理由についても説明します。

小さいトランザクション

トランザクションが小さい場合、バイナリログ作成により、データのロードに必要なディスク書き込み数が 2 倍になります。この影響は、他のデータベースセッションのパフォーマンスを著しく低下させ、データのロードに必要な時間を長くします。劣化の程度は、以下の要因に一部依存します。

- アップロードレート
- ロード中に発生するその他のデータベースアクティビティ
- Amazon RDS DB インスタンスの容量

バイナリログは、バックアップおよび削除されるまでにロードされたデータの量とほぼ同じディスク容量を消費します。Amazon RDS は、バイナリログを頻繁にバックアップして削除することで、この容量を最小限に抑えます。

大きいトランザクション

大規模なトランザクションの場合、バイナリログによって、次の理由により IOPS とディスク使用量が 3 倍になります。

- バイナリログキャッシュは、トランザクションデータを一時的にディスクに保存します。
- このキャッシュはトランザクションのサイズに応じて大きくなり、ディスク領域を消費します。
- トランザクション (コミットまたはロールバック) が完了すると、システムはキャッシュをバイナリログにコピーします。

このプロセスでは、データのコピーが 3 つ作成されます。

- 元のデータ
- ディスク上のキャッシュ
- 最後のバイナリログエントリ

書き込みオペレーションごとに追加の IO が発生し、パフォーマンスにさらに影響を及ぼします。

このため、バイナリログ記録には、ログ記録を無効にした場合と比較して 3 倍のディスク容量が必要です。例えば、1 つのトランザクションとして 10 GiB のデータをロードすると、次の 3 つのコピーが作成されます。

- テーブルデータに 10 GiB
- バイナリログキャッシュに 10 GiB
- バイナリログファイルに 10 GiB

必要な一時ディスク容量の合計は 30 GiB です。

ディスク容量に関する重要な考慮事項:

- キャッシュファイルは、セッションが終了するか、新しいトランザクションによって別のキャッシュが作成されるまで保持されます。

- バイナリログはバックアップされるまで保持され、長期間にわたって 20 GiB (キャッシュとログ) が保持される可能性があります。

`LOAD DATA LOCAL INFILE` を使用してデータをロードする場合、ロード前に行われたバックアップからデータベースを復元する必要がある場合に備えて、データ復旧によって 4 番目のコピーが作成されます。復旧時に、MariaDB はバイナリログからフラットファイルにデータを抽出します。その後 MariaDB は `LOAD DATA LOCAL INFILE` を実行します。上記の例の場合、この復旧には合計 40 GiB、つまりテーブル、キャッシュ、ログ、ローカルファイルごとに 10 GiB の一時ディスク領域が必要です。少なくとも 40 GiB の空きディスク容量がないと、復旧は失敗します。

大規模なデータロードの最適化

大量のデータをロードする場合は、バイナリログを無効にして、オーバーヘッドとディスク領域の要件を削減します。バイナリログ記録を無効にするには、バックアップ保持期間を 0 に設定します。ロードが完了したら、バックアップ保持期間を適切なゼロ以外の値に復元します。詳細については、設定テーブルの「Amazon RDS DB インスタンスを変更する」および「バックアップ保持期間」を参照してください。

> **Note**
>
> DB インスタンスがリードレプリカのソース DB インスタンスである場合、バックアップ保持期間を 0 に設定することはできません。

データをロードする前に、DB スナップショットを作成することをお勧めします。詳細については、「手動バックアップの管理」を参照してください。

InnoDB

undo ログと復旧のオプションに関する以下の情報は、InnoDB トランザクションを小さくしてデータベースのパフォーマンスを最適化することをサポートしています。

InnoDB の undo ログについて

undo ログは、トランザクションのロールバックを有効にし、マルチバージョン同時実行制御 (MVCC) をサポートするログ記録メカニズムです。

MariaDB 10.11 以前のバージョンでは、undo ログは InnoDB システムテーブルスペース (通常は ibdata1) に保存され、パージスレッドによって削除されるまで保持されます。その結果、大規模な

データロードトランザクションによってシステムテーブルスペースが非常に大きくなり、データベースを再作成しない限り再利用できないディスク領域が消費される可能性があります。

すべての MariaDB バージョンでは、最も古いアクティブなトランザクションがコミットまたはロールバックされるまで、パージスレッドは undo ログの削除を待機する必要があります。データベースが、ロード中に他のトランザクションを処理する場合、そのトランザクションがコミットされ、他のトランザクションが MVCC の undo ログを必要としない場合でも、それらの undo ログも蓄積され、削除できません。この場合、読み取り専用トランザクションを含むすべてのトランザクションが遅くなります。このスローダウンは、すべてのトランザクションが、ロードトランザクションだけでなく、任意のトランザクションが変更するすべての行にアクセスするために発生します。実際には、トランザクションは、長時間実行されるロードトランザクションによって undo ログのクリーンアップ中に消去されなかった undo ログをスキャンする必要があります。これは、変更された行にアクセスするオペレーションのパフォーマンスに影響します。

InnoDB トランザクション復旧オプション

InnoDB はコミットオペレーションを最適化しますが、大規模なトランザクションのロールバックは遅くなります。復旧を高速化するには、ポイントインタイムリカバリを実行するか、DB スナップショットを復元します。詳細については、「ポイントインタイムリカバリ」および「DB インスタンスへの復元」を参照してください。

データインポート形式

MariaDB は、フラットファイルと SQL の 2 つのデータインポート形式をサポートしています。各形式に関する情報を確認して、ニーズに最適なオプションを決定します。

フラットファイル

小規模なトランザクションの場合は、`LOAD DATA LOCAL INFILE` を使用してフラットファイルをロードします。このデータインポート形式は、SQL を使用する場合に比べて次のようなメリットがあります。

- ネットワークトラフィックの削減
- データ転送コストの削減
- データベース処理オーバーヘッドの減少
- より高速な処理

`LOAD DATA LOCAL INFILE` はフラットファイル全体を 1 つのトランザクションとしてロードします。個々のファイルのサイズを小さく保つと、次の利点があります。

- 再開機能 - どのファイルがロードされたかを追跡できます。ロード中に問題が発生した場合、中断したところから再開できます。一部のデータを Amazon RDS に再送信する必要がある場合でも、ファイルが小さいと、再送信される量は最小限ですみます。
- 同時データロード – 単一ファイルのロードに十分な IOPS とネットワーク帯域幅がある場合、同時にロードすれば時間を節約できます。
- ロード速度制御 – データロードが他のプロセスに悪影響を及ぼす場合は、ファイル間の間隔を増やすことでロード速度を制御できます。

大規模なトランザクションでは、`LOAD DATA LOCAL INFILE` を使用してデータをインポートするメリットが減ります。大量のデータを小さなファイルに分割できない場合は、SQL の使用を検討してください。

SQL

SQL にはフラットファイルに比べて、トランザクションサイズを簡単に小さく保つことができるという大きなメリットがあります。ただし、SQL のロードにはフラットファイルよりもかなり時間がかかる場合があります。また、障害が発生した後、どこから再開するかを判断するのが難しい場合があります。mariadb-dump ファイルを再開することはできません。mariadb-dump ファイルのロード中にエラーが発生した場合、ロードを再開するにはファイルを変更するか置き換える必要があります。または、障害の原因を修正した後、ロード前の時点に復元してファイルを再開する必要があります。詳細については、「ポイントインタイムリカバリ」を参照してください。

データベースチェックポイントに Amazon RDS DB スナップショットを使用する

バイナリログ記録を使用せずに、数時間または数日など長期間にわたってデータをロードする場合は、DB スナップショットを使用して、データの安全性を確保するために定期的なチェックポイントを提供します。各 DB スナップショットは、システム障害やデータ破損イベント時の復旧ポイントとして機能するデータベースインスタンスの一貫したコピーを作成します。DB スナップショットは高速であるため、頻繁なチェックポイント作成によるロードパフォーマンスへの影響は最小限に抑えられます。データベースの耐久性や復旧機能に影響を与えることなく、以前の DB スナップショットを削除できます。DB スナップショットの詳細については、「手動バックアップの管理」を参照してください。

データベースのロード時間を短縮する

以下の項目は、ロード時間を短縮するための追加のヒントです。

- MariaDB データベースにデータをロードする前に、すべてのセカンダリインデックスを作成します。他のデータベースシステムとは異なり、MariaDB はセカンダリインデックスを追加または変

更するときにテーブル全体を再構築します。このプロセスでは、インデックスが変更された新しいテーブルが作成され、すべてのデータがコピーされ、元のテーブルが削除されます。

- プライマリキーの順序でデータをロードします。InnoDB テーブルの場合、これによりロード時間が 75%～80% 短縮され、データファイルサイズが 50% 短縮されます。
- `foreign_key_checks` を `0` に設定して、外部キーの制約を無効にします。これは多くの場合、`LOAD DATA LOCAL INFILE` を使用してロードされたフラットファイルで必要となります。どのロードでも、外部キーチェックを無効にすると、データのロードが高速化されます。ロードが完了したら、`foreign_key_checks` を `1` に設定して制約を再度有効にし、データを検証します。
- リソース制限に近づかない限り、データを同時にロードします。複数のテーブルセグメント間で同時ロードを有効にするには、必要に応じてパーティションテーブルを使用します。
- SQL 実行のオーバーヘッドを減らすには、複数の `INSERT` ステートメントを単一の複数値 `INSERT` オペレーションに結合します。`mariadb-dump` は、この最適化を自動的に実装します。
- `innodb_flush_log_at_trx_commit` を `0` に設定することで、InnoDB ログ IO オペレーションを減らします。ロードが完了したら、`innodb_flush_log_at_trx_commit` を `1` に戻します。

> **Warning**
>
> `innodb_flush_log_at_trx_commit` を `0` に設定すると、InnoDB はコミットごとではなく毎秒ログをフラッシュします。この設定はパフォーマンスを向上させますが、システム障害時にトランザクションが失われるリスクがあります。

- リードレプリカがない DB インスタンスにデータをロードする場合は、`sync_binlog` を `0` に設定します。ロードが完了したら、`sync_binlog parameter` を `1` に戻します。
- DB インスタンスをマルチ AZ 配置に変換する前に、シングル AZ DB インスタンスにデータをロードします。DB インスタンスが既にマルチ AZ 配置を使用している場合は、データロードのためにシングル AZ 配置に切り替えることはお勧めしません。これで実現できる改善はわずかだけです。

外部の MariaDB データベースから Amazon RDS for MariaDB DB インスタンスにデータをインポートする

既存の MariaDB データベースから RDS for MariaDB DB インスタンスに、データをインポートできます。これを行うには、mysqldump または mariadb-dump を使用してデータベースをコピーし、データベースを RDS for MariaDB DB インスタンスに直接パイプ処理します。`mysqldump` または `mariadb-dump` コマンドラインユーティリティは、データのバックアップや、MariaDB サーバー

から別の場所への転送によく使用されます。このユーティリティは、MariaDB クライアントソフトウェアに含まれています。

MariaDB 11.0.1 以降では、mariadb-dump を mysqldump の代わりに使用する必要があります。

外部のデータベースから Amazon RDS DB インスタンスにデータを移動するための一般的な mysqldump コマンドは、次の例のようになります。値を自分の情報に置き換えます。MariaDB 11.0.1 以降のバージョンでは、mysqldump を mariadb-dump に置き換えます。

```
mysqldump -u local_user \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mariadb -u RDS_user \
        --port=port_number \
        --host=host_name \
        -pRDS_password
```

> **Important**
>
> -p オプションと入力するパスワードの間にスペースを残していないことを確認します。セキュリティのベストプラクティスとして、この例に表示されているプロンプト以外の認証情報を指定してください。

次の推奨事項と考慮事項に注意してください。

- ダンプファイルから次のスキーマを除外します。
  - sys
  - performance_schema
  - information_schema

  mysqldump および mariadb-dump ユーティリティは、これらのスキーマをデフォルトで除外します。

- ユーザーや権限を移行する必要がある場合は、再作成するデータ制御言語 (DCL) を生成するツールの使用を検討します。例えば、pt-show-grants ユーティリティがあります。
- インポートを実行するには、そのユーザーに DB インスタンスへのアクセスが許可されていることを確認してください。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

使用するパラメータは次のとおりです。

- `-u local_user` - ユーザー名を指定します。このパラメータの初回の使用では、`--databases` パラメータによって識別されたローカル MariaDB データベースのユーザーアカウントの名前を指定します。
- `--databases database_name` — Amazon RDS にインポートするローカル MariaDB インスタンスのデータベースの名前を指定します。
- `--single-transaction` - ローカルデータベースからロードされたすべてのデータが、ある時点において一貫していることを確認します。`mysqldump` または `mariadb-dump` によるデータの読み取り中に他のプロセスがデータを変更する場合は、このパラメータを使用することでデータの整合性を維持できます。
- `--compress` - ローカルデータベースからのデータを、Amazon RDS に送信する前に圧縮することで、ネットワーク帯域幅の消費量を削減します。
- `--order-by-primary` - 各テーブルのデータをプライマリキーでソートすることで、ロード時間を短縮します。
- `--routines` – ストアドプロシージャや関数などのルーチンが、コピーするデータベースに存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にルーチンを除外します。その後、Amazon RDS データベースでルーチンを手動で再作成します。
- `--triggers` – コピーするデータベースにトリガーが存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にトリガーを除外します。その後、Amazon RDS データベースでトリガーを手動で再作成します。
- `--events` – コピーするデータベースにイベントが存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にイベントを除外します。その後、Amazon RDS データベースでイベントを手動で再作成します。
- `-plocal_password` - パスワードを指定します。このパラメータの初回の使用では、初期の `-u` パラメータにより識別されるユーザーアカウントのパスワードを指定します。

- -u *RDS_user* - ユーザー名を指定します。このパラメータの 2 回目の使用では、--host パラメータによって識別された MariaDB DB インスタンスのデフォルトデータベースのユーザーアカウントの名前を指定します。
- --port *port_number* — MariaDB DB インスタンスのポートを指定します。DB インスタンスの作成時に値を変更していない限り、デフォルトでは 3306 です。
- --host *host_name* — Amazon RDS DB インスタンスのエンドポイント、例えば からのドメインネームシステム (DNS) 名を指定します。myinstance.123456789012.us-east-1.rds.amazonaws.comエンドポイントの値は、Amazon RDS コンソールの DB インスタンスの詳細で確認できます。
- -p*RDS_password* - パスワードを指定します。このパラメータの 2 回目の使用では、2 回目の -u パラメータにより識別されるユーザーアカウントのパスワードを指定します。

Amazon RDS データベースで、ストアドプロシージャ、トリガー、関数、イベントを必ず手動で作成してください。これらのオブジェクトのいずれかがコピー対象のデータベースに含まれている場合は、mysqldump または mariadb-dump の実行時に除外します。これを行うには、mysqldump または mariadb-dump コマンドに次のパラメータを含めます。

- --routines=0
- --triggers=0
- --events=0

例

次の例では、ローカルホストにある world サンプルデータベースを、RDS for MariaDB DB インスタンスにコピーします。値を自分の情報に置き換えます。

Linux、macOS、Unix の場合:

```
sudo mariadb-dump -u local_user \
    --databases world \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mariadb -u rds_user \
```

```
        --port=3306 \
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com \
        -pRDS_password
```

Windows の場合:

Windows プログラムメニューの [コマンドプロンプト] を右クリックし、[管理者として実行] を選択して開いたコマンドプロンプトで次のコマンドを実行します。値を自分の情報に置き換えます。

```
mariadb-dump -u local_user ^
    --databases world ^
    --single-transaction ^
    --compress ^
    --order-by-primary  ^
    --routines=0 ^
    --triggers=0 ^
    --events=0 ^
    -plocal_password | mariadb -u RDS_user ^
        --port=3306 ^
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com ^
        -pRDS_password
```

> **Note**
>
> セキュリティのベストプラクティスとして、例に表示されているプロンプト以外の認証情報を指定します。

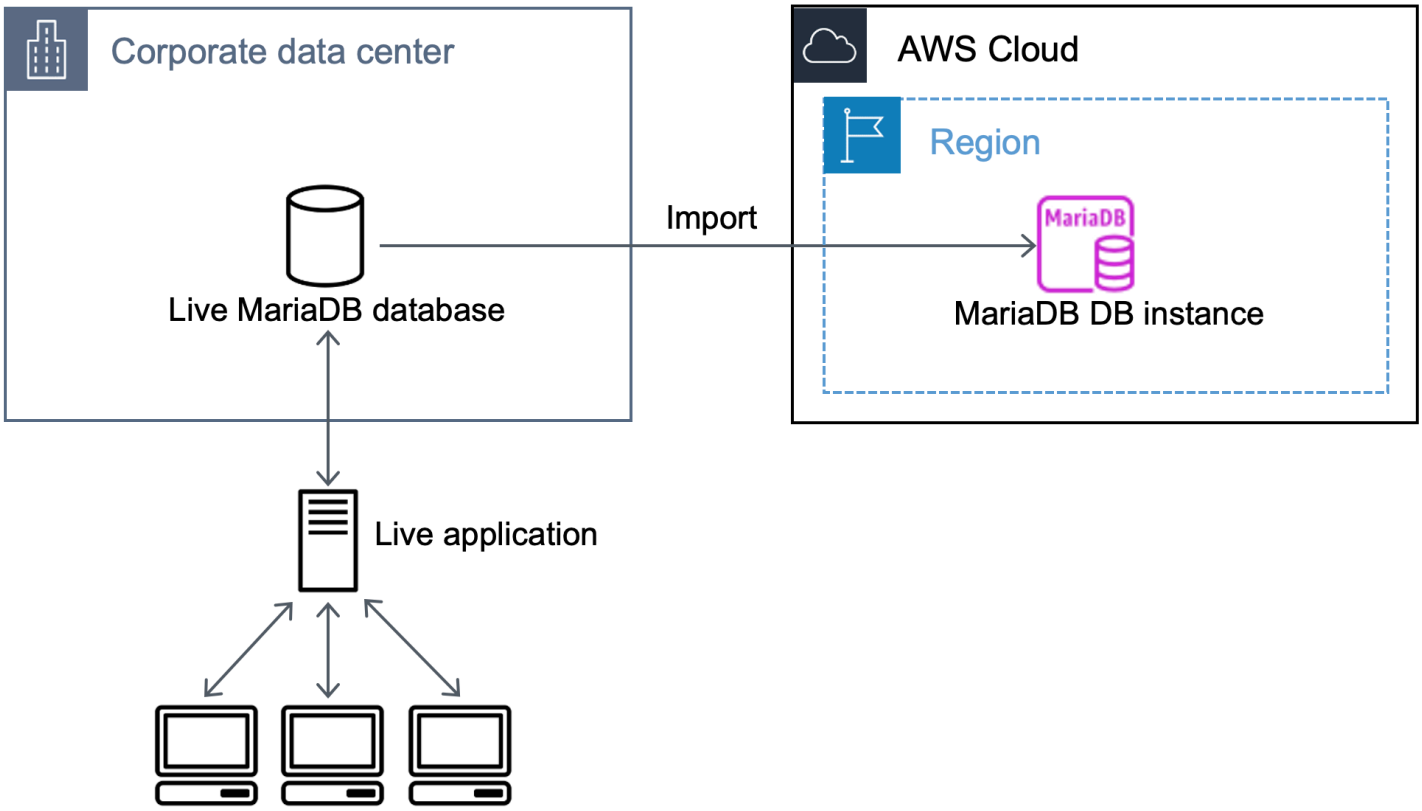
ダウンタイムを短縮して Amazon RDS for MariaDB DB インスタンスにデータをインポートする

場合によっては、ライブアプリケーションをサポートする外部 MariaDB データベースから RDS for MariaDB DB インスタンスにデータをインポートする必要がある場合があります。次の手順を使用して、アプリケーションの可用性への影響を最小限に抑えることができます。この手順は、巨大なデータベースを使用する場合にも役立ちます。この手順を使用すると、ネットワーク経由で AWS に渡されるデータ量を削減することで、インポートのコストを削減できます。

この手順では、データベースデータのコピーを Amazon EC2 インスタンスに送信し、そのデータを新しい Amazon RDS データベースにインポートします。次に、レプリケーションを使用し

て、Amazon RDS データベースをライブ外部インスタンスで最新の状態にした後、アプリケーションを Amazon RDS データベースにリダイレクトします。外部のインスタンスが MariaDB 10.0.24 以降であり、ターゲットインスタンスが RDS for MariaDB である場合は、グローバルトランザクション識別子 (GTID) に基づいて MariaDB のレプリケーションを設定します。それ以外の場合は、バイナリログの調整に基づいてレプリケーションを設定します。外部データベースがサポートしている場合は、GTID ベースのレプリケーションが推奨されます。GTID ベースのレプリケーションは信頼性の高い方法だからです。詳細については、MariaDB ドキュメントの「グローバルトランザクション ID」を参照してください。

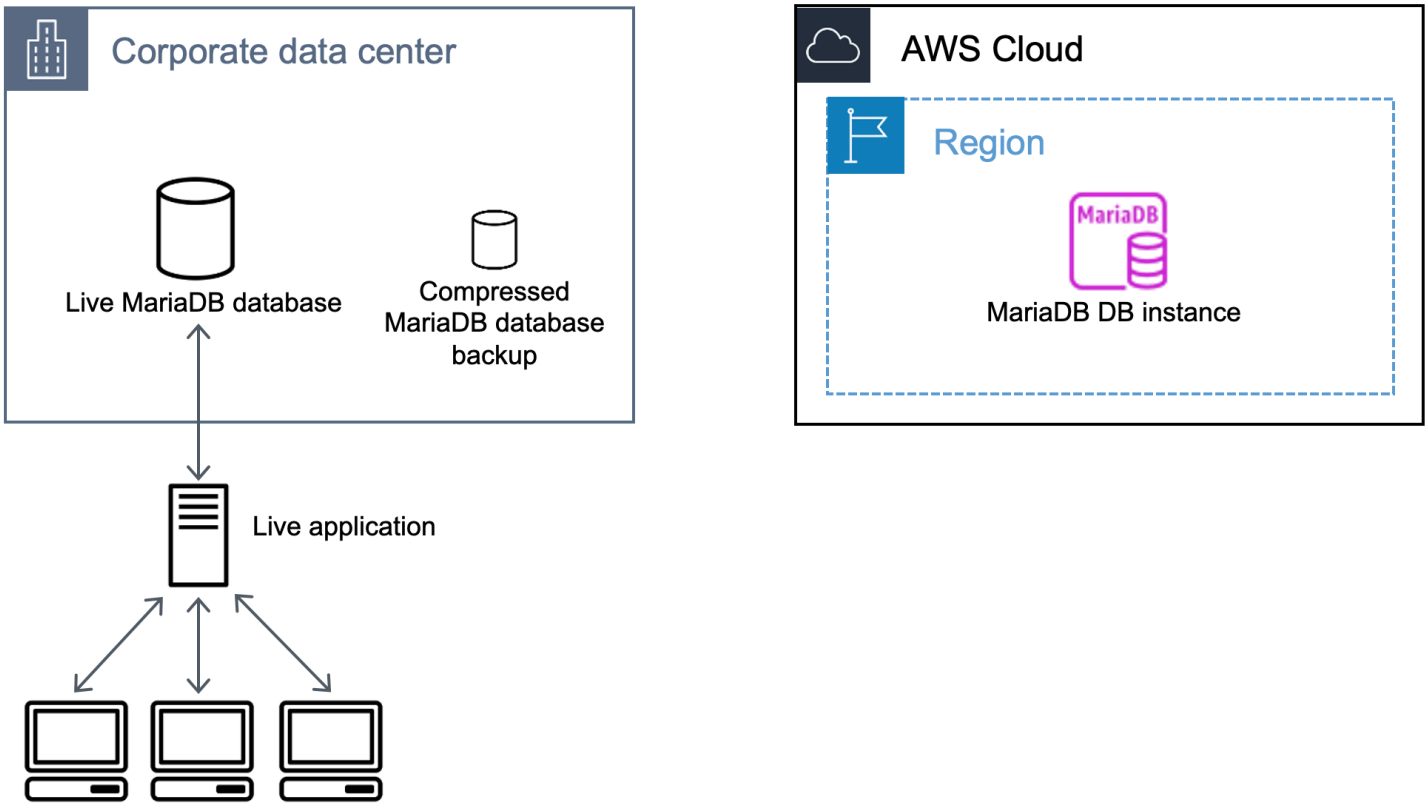
次の図は、外部 MariaDB データベースを Amazon RDS 上の MariaDB データベースにインポートする様子を示しています。



タスク 1: 既存のデータベースのコピーを作成する

最小限のダウンタイムで RDS for MariaDB データベースに大量のデータを移行するプロセスでは、最初のステップとしてソースデータのコピーを作成します。

次の図は、MariaDB データベースのバックアップの作成を示しています。



SQL 形式または区切り文字付きテキスト形式でデータベースのバックアップを作成するには、mysqldump または mariadb-dump ユーティリティを使用できます。MariaDB 10.5 では、クライアントは mariadb-dump と呼ばれます。MariaDB 11.0.1 以降では、mariadb-dump を mysqldump の代わりに使用する必要があります。非運用環境で各形式のテスト実行を行って、どちらの方法が mysqldump または mariadb-dump の実行時間が短いか確認することをお勧めします。

また、ロードに区切り文字付きテキスト形式を使用することでもたらされるメリットに対して、mysqldump または mariadb-dump のパフォーマンスの重み付けをすることをお勧めします。区切り文字付きテキスト形式を使用したバックアップでは、ダンプされる各テーブルについてタブ区切りテキストファイルを作成されます。データベースのインポートに必要な時間を短縮するため、LOAD DATA LOCAL INFILE コマンドを使用してこれらのファイルを同時にロードできます。詳細については、任意のソースからデータをインポートする手順の「ステップ 5: データをロードする」を参照してください。

バックアップ操作をスタートする前に、Amazon RDS にコピーする MariaDB データベースでレプリケーションのオプションを設定してください。レプリケーションのオプションには、バイナリログ記録の有効化や一意のサーバー ID の設定が含まれます。これらのオプションを設定すると、サーバーはデータベーストランザクションのログ作成をスタートし、このプロセスの後でこのサーバーをソースレプリケーションインスタンスにするために準備します。

次の推奨事項と考慮事項に注意してください。

- --single-transaction オプションは、データベースの一貫した状態をダンプするため、mysqldump または mariadb-dump と共に使用します。有効なダンプファイルを確保するために、mysqldump または mariadb-dump の実行中はデータ定義言語 (DDL) ステートメントを実行しないでください。これらのオペレーションに対してメンテナンスウィンドウをスケジュールできます。
- ダンプファイルから次のスキーマを除外します。
  - sys
  - performance_schema
  - information_schema

  mysqldump および mariadb-dump ユーティリティは、これらのスキーマをデフォルトで除外します。
- ユーザーや権限を移行する必要がある場合は、再作成するデータ制御言語 (DCL) を生成するツールの使用を検討します。例えば、pt-show-grants ユーティリティがあります。

レプリケーションオプションを設定するには

1. my.cnf ファイルを編集します。このファイルは /etc にあります。

   ```
   sudo vi /etc/my.cnf
   ```

   log_bin オプションと server_id オプションを [mysqld] に追加します。log_bin オプションは、バイナリログファイルのファイル名識別子を提供します。server_id オプションは、ソースとレプリカの関係のサーバーに一意の識別子を提供します。

   次の例は、my.cnf ファイルの更新された [mariadb] セクションを示しています。

   ```
   [mariadb]
   log-bin
   server-id=1
   log-basename=master1
   binlog-format=mixed
   ```

   詳細については、MariaDB ドキュメントの「Setting the Replication Source Configuration」を参照してください。

2. マルチ AZ DB クラスターでのレプリケーションでは、gtid_strict_mode を有効にします。詳細については、MariaDB のドキュメントの「gtid_strict_mode」を参照してください。

   DB インスタンスでのレプリケーションには、gtid_strict_mode を有効にする必要はありません。

3. mariadb サービスを再起動します。

```
sudo service mariadb restart
```

既存のデータベースのバックアップコピーを作成するには

1. mysqldump または mariadb-dump ユーティリティを使用し、SQL 形式または区切り文字付きテキスト形式を指定して、データのバックアップを作成します。

   パフォーマンスを向上させ、データ整合性を確保するために、mysqldump または mariadb-dump で --order-by-primary オプションと --single-transaction オプションを使用します。

   MySQL システムデータベースをバックアップに含めないためには、mysqldump または mariadb-dump で --all-databases オプションを使用しないでください。詳細については、MySQL ドキュメントの「Creating a Data Snapshot Using mysqldump」を参照してください。

   バックアップファイルが作成されるディレクトリを書き込み可能にするために、必要に応じて chmod を使用します。

> **Important**
>
> Windows で、管理者としてコマンドウィンドウを実行します。

- SQL 出力を作成するには、次のコマンドを使用します。MariaDB 10.11 以前のバージョンでは、mariadb-dump を mysqldump に置き換えます。

  Linux、macOS、Unix の場合:

```
sudo mariadb-dump \
    --databases database_name \
```

```
    --master-data=2  \
    --single-transaction \
    --order-by-primary \
    -r backup.sql \
    -u local_user \
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。

Windows の場合:

```
mariadb-dump ^
    --databases database_name ^
    --master-data=2  ^
    --single-transaction ^
    --order-by-primary ^
    -r backup.sql ^
    -u local_user ^
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。

- 区切り文字付きテキスト出力を作成するには、次のコマンドを使用します。MariaDB 11.01 以降のバージョンでは、`mysqldump` を `mariadb-dump` に置き換えます。

  Linux、macOS、Unix の場合:

```
sudo mysqldump \
    --tab=target_directory \
    --fields-terminated-by ',' \
    --fields-enclosed-by '"' \
    --lines-terminated-by 0x0d0a \
    database_name \
```

```
    --master-data=2 \
    --single-transaction \
    --order-by-primary \
    -ppassword
```

Windows の場合:

```
mysqldump ^
    --tab=target_directory ^
    --fields-terminated-by "," ^
    --fields-enclosed-by """ ^
    --lines-terminated-by 0x0d0a ^
    database_name ^
    --master-data=2 ^
    --single-transaction ^
    --order-by-primary ^
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。
> Amazon RDS データベースで、ストアドプロシージャ、トリガー、関数、イベントを必ず手動で作成してください。これらのオブジェクトのいずれかがコピー対象のデータベースに含まれている場合は、mysqldump または mariadb-dump の実行時に除外します。これを行うには、mysqldump または mariadb-dump コマンドで以下の引数を含めます。
>
> - --routines=0
> - --triggers=0
> - --events=0

mysqldump を実行して区切り文字付きテキスト形式を指定すると、CHANGE MASTER TO コメントが返されます。このコメントには、マスターログのファイル名と場所が含まれます。外部インスタンスが MariaDB 10.0.23 以前のバージョンの場合は、MASTER_LOG_FILE および MASTER_LOG_POS の値を書き留めてください。これらの値は、レプリケーションを設定するときに必要です。

MariaDB バージョンでは、次の出力が返されます。

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

2. 使用している外部インスタンスが MariaDB バージョン 10.0.24 以降である場合は、GTID ベースのレプリケーションを使用します。外部 MariaDB インスタンスで SHOW MASTER STATUS を実行してバイナリログファイル名と場所を取得し、外部 MariaDB インスタンス上で BINLOG_GTID_POS を実行してそれらを GTID に変換します。

```
SELECT BINLOG_GTID_POS('binary_log_file_name', binary_log_file_position);
```

返された GTID を書き留めます。これはレプリケーションを設定する際に必要となります。

3. Amazon RDS データベースにデータをコピーするために必要なネットワークリソースの量を減らすために、コピーされたデータを圧縮します。バックアップファイルのサイズをメモします。この情報は、作成する Amazon EC2 インスタンスの大きさを決定するときに必要です。作業が終了したら、GZIP または任意の圧縮ユーティリティを使用してバックアップファイルを圧縮します。

   - SQL 出力を圧縮するには、次のコマンドを使用します。

     ```
     gzip backup.sql
     ```

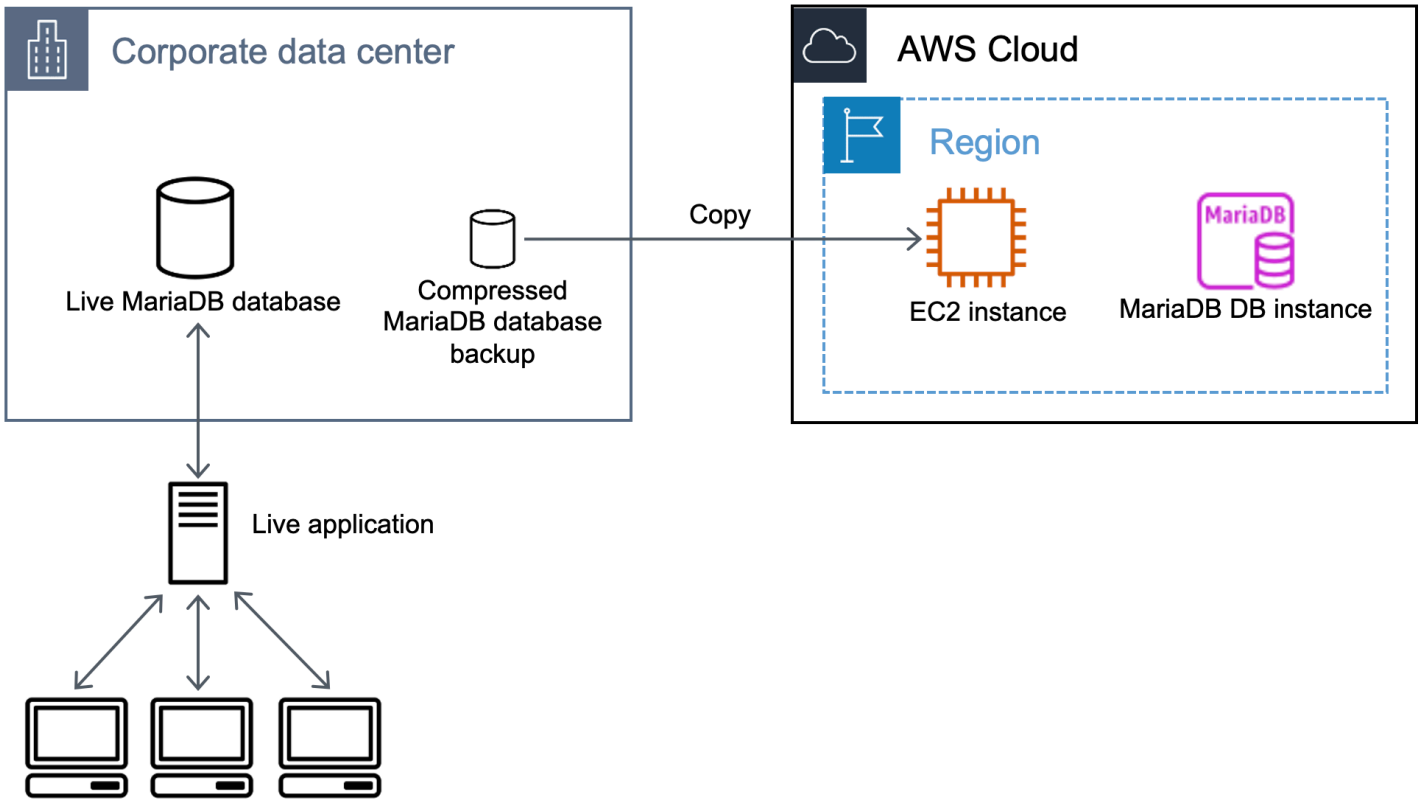
   - 区切り文字付きテキスト出力を圧縮するには、次のコマンドを使用します。

     ```
     tar -zcvf backup.tar.gz target_directory
     ```

タスク 2: Amazon EC2 インスタンスを作成し、圧縮したデータベースをコピーする

圧縮したデータベースのバックアップファイルを Amazon EC2 インスタンスにコピーする場合、データベースインスタンス間で非圧縮データを直接コピーするよりも必要なネットワークリソースは少なくなります。データを Amazon EC2 にコピーしたら、そこから MariaDB データベースに直接コピーできます。ネットワークリソースのコストを節約するには、Amazon EC2 インスタンスが Amazon RDS DB インスタンスと同じ AWS リージョンに存在している必要があります。Amazon EC2 インスタンスを Amazon RDS データベースと同じ AWS リージョンに配置することで、インポート時のネットワークレイテンシーも低減されます。

次の図は、Amazon EC2 インスタンスへのデータベースバックアップのコピーを示しています。



Amazon EC2 インスタンスを作成し、データをコピーするには

1. Amazon RDS データベースを作成する予定の AWS リージョンに、仮想プライベートクラウド (VPC)、VPC セキュリティグループ、および VPC サブネットを作成します。VPC セキュリティグループのインバウンドルールで、アプリケーションが AWS に接続するために必要な IP アドレスを許可していることを確認します。IP アドレスの範囲 (`203.0.113.0/24` など) や別の VPC セキュリティグループを指定できます。Amazon VPC コンソールを使用して、VPC、サブネット、セキュリティグループを作成および管理できます。詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「Amazon VPC の開始方法」を参照してください。
2. Amazon EC2 管理コンソールを開き、Amazon EC2 インスタンスと Amazon RDS データベースの両方が含まれる AWS リージョンを選択します。ステップ 1 で作成した VPC、サブネット、セキュリティグループを使用して Amazon EC2 インスタンスを起動します。非圧縮の場合のデータベースバックアップファイルに十分なストレージを備えたインスタンスタイプを選択していることを確認します。Amazon EC2 インスタンスの詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EC2 の使用を開始する」を参照してください。
3. Amazon EC2 インスタンスから Amazon RDS データベースに接続するには、VPC セキュリティグループを編集します。EC2 インスタンスのプライベート IP アドレスを指定するインバウンド

ルールを追加します。このプライベート IP アドレスは、EC2 コンソールウィンドウの [Instance] ペインの [Details] タブで確認できます。VPC セキュリティグループを編集してインバウンドルールを追加するには、EC2 コンソールのナビゲーションペインの [Security Groups] (セキュリティグループ) でセキュリティグループを選択してから、EC2 インスタンスのプライベート IP アドレスを指定して MySQL または Aurora のインバウンドルールを追加します。VPC セキュリティグループにインバウンドルールを追加する方法については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルール」を参照してください。

4. ローカルシステムから Amazon EC2 インスタンスに、圧縮されたデータベースバックアップファイルをコピーします。必要に応じて chmod を使用して、Amazon EC2 インスタンスのターゲットディレクトリに対する書き込みアクセス許可があることを確認します。scp または Secure Shell (SSH) クライアントを使用してファイルをコピーできます。以下は scp コマンドの例です。

```
scp -r -i key pair.pem backup.sql.gz ec2-user@EC2 DNS:/target_directory/backup.sql.gz
```

> **Important**
>
> 機密データをコピーするときは、必ず安全なネットワーク転送プロトコルを使用してください。

5. Amazon EC2 インスタンスに接続し、次のコマンドを使用して最新のアップデートと MariaDB クライアントツールをインストールします。

```
sudo yum update -y
sudo yum install mariadb1011-client-utils -y
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の Linux インスタンスの「インスタンスに接続する」および「MariaDB ドキュメント」の「MariaDB Connectors」を参照してください。

6. Amazon EC2 インスタンスに接続されている間に、データベースバックアップファイルを解凍します。次のコマンドはその例です。

   - SQL 出力を解凍するには、次のコマンドを使用します。

     ```
     gzip backup.sql.gz -d
     ```

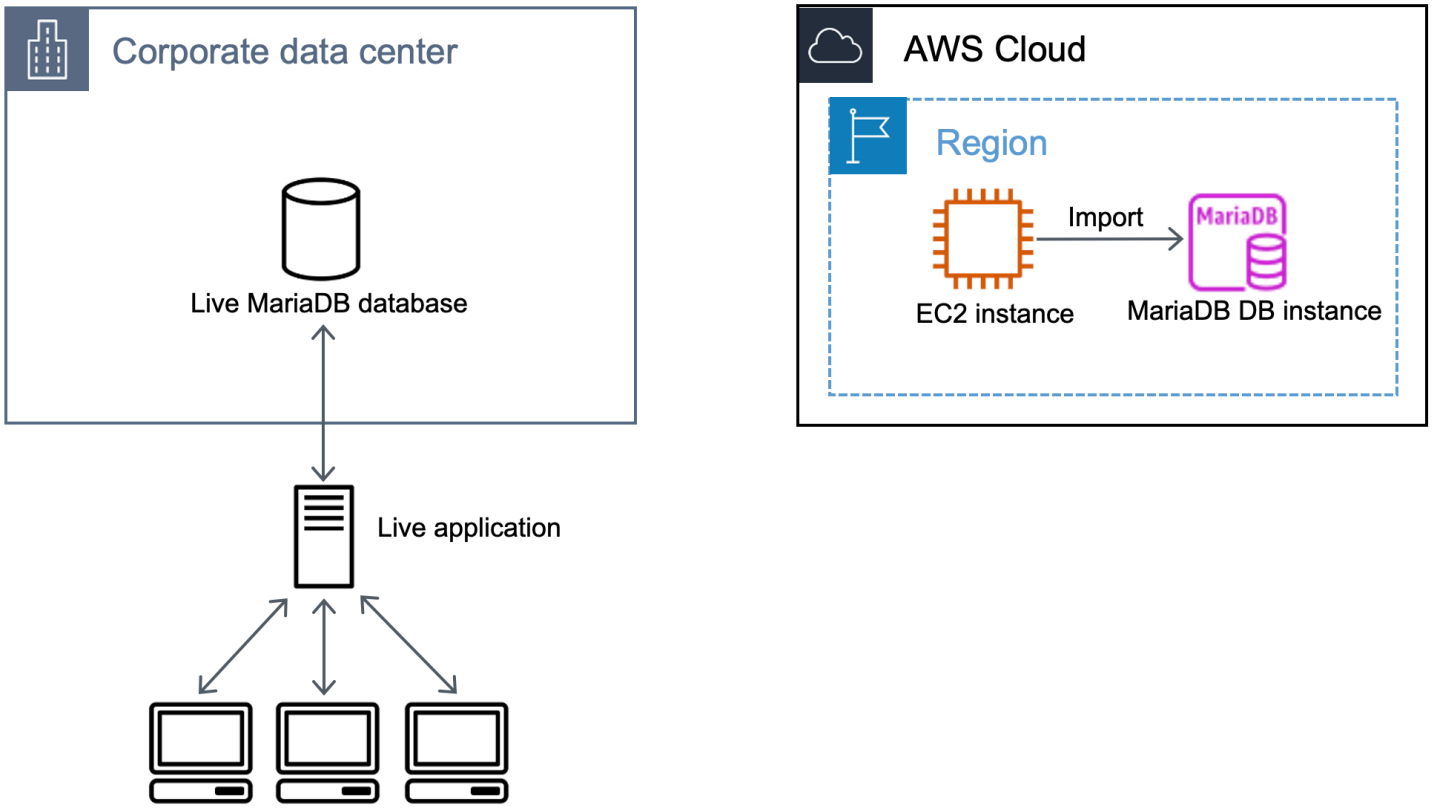
   - 区切り文字付きテキスト出力を解凍するには、次のコマンドを使用します。

```
tar xzvf backup.tar.gz
```

タスク 3: MariaDB データベースを作成し、Amazon EC2 インスタンスからデータをインポートする

Amazon EC2 インスタンスと同じ AWS リージョンで RDS for MariaDB DB インスタンスを作成することによって、インターネット経由でインポートするよりも速く、Amazon EC2 からデータベースのバックアップファイルをインポートできます。

次の図は、Amazon EC2 インスタンスから MariaDB データベースにバックアップをインポートする様子を示しています。



MariaDB データベースを作成し、データをインポートするには

1. この Amazon RDS データベースについて予想されるワークロードをサポートするのに必要な DB インスタンスクラスとストレージ領域の容量を決定します。このプロセスの一環として、データロードの手順に十分な領域と処理能力を決定します。また、本番稼働ワークロードの処理に必要なものも決定します。これは、ソースの MariaDB データベースのサイズおよびリソース

に基づいて見積もることができます。詳細については、「DB インスタンスクラス」を参照してください。

2. Amazon EC2 インスタンスを含む AWS リージョンに、DB インスタンスを作成します。「Amazon RDS DB インスタンスの作成」の指示に従い、次のガイドラインを使用します。

   - ソース DB インスタンスと互換性のある DB エンジンのバージョンを指定します。
   - Amazon EC2 インスタンスと同じ仮想プライベートクラウド (VPC) と VPC セキュリティグループを指定します。これにより、Amazon EC2 インスタンスと Amazon RDS インスタンスはネットワーク上で相互に表示されることが確認できます。DB インスタンスがパブリックにアクセスできることを確認してください。以下のセクションで説明するように、ソースデータベースでレプリケーションをセットアップするには、DB インスタンスにパブリックアクセス可能である必要があります。
   - データベースのバックアップのインポートが完了するまで、複数のアベイラビリティーゾーン、バックアップ保持、リードレプリカを設定しないでください。インポートが完了したら、本稼働インスタンスについて、マルチ AZ とバックアップ保持を設定できます。

3. Amazon RDS データベースのデフォルトの設定オプションを確認します。データベースのデフォルトパラメータグループに必要な設定オプションがない場合は、別のパラメータグループを検索するか、新しいパラメータグループを作成します。パラメータグループの作成の詳細については、「Amazon RDS のパラメータグループ」を参照してください。

4. マスターユーザーとして、新しい Amazon RDS データベースに接続します。DB インスタンスにアクセスする必要がある管理者、アプリケーション、およびサービスのサポートに必要なユーザーを作成します。Amazon RDS データベースのホスト名は、この DB インスタンスの [エンドポイント] の値からポート番号を除いた値 (例: `mysampledb.123456789012.us-west-2.rds.amazonaws.com`) です。エンドポイントの値は、Amazon RDS コンソールのデータベースの詳細で確認できます。

5. Amazon EC2 インスタンスに接続します。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の Linux インスタンスの「インスタンスをクリーンアップする」を参照してください。

6. `mysql` コマンドを使用して、Amazon EC2 インスタンスからリモートホストとして Amazon RDS データベースに接続します。コマンドの例を次に示します。

   ```
   mysql -h host_name -P 3306 -u db_master_user -p
   ```

   host_name は、Amazon RDS データベースのエンドポイントです。

7. mysql プロンプトで、データベースダンプファイルの名前を渡して source コマンドを実行します。このコマンドは、Amazon RDS DB インスタンスにデータをロードします。

   - SQL 形式の場合は、次のコマンドを使用します。

     ```
     MariaDB [(none)]> source backup.sql;
     ```

   - 区切り文字付きテキスト形式の場合は、Amazon RDS データベースをセットアップしたときに作成したデフォルトのデータベースではない場合、まず、データベースを作成します。

     ```
     MariaDB [(none)]> create database database_name;
     MariaDB [(none)]> use database_name;
     ```

     次にテーブルを作成します。

     ```
     MariaDB [(none)]> source table1.sql
     MariaDB [(none)]> source table2.sql
     etc...
     ```

     次にデータをインポートします。

     ```
     MariaDB [(none)]> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS
      TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
     MariaDB [(none)]> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS
      TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
     etc...
     ```

     パフォーマンスを向上させるために、複数の接続からこれらのオペレーションをパラレル実行して、すべてのテーブルを同時に作成およびロードすることができます。

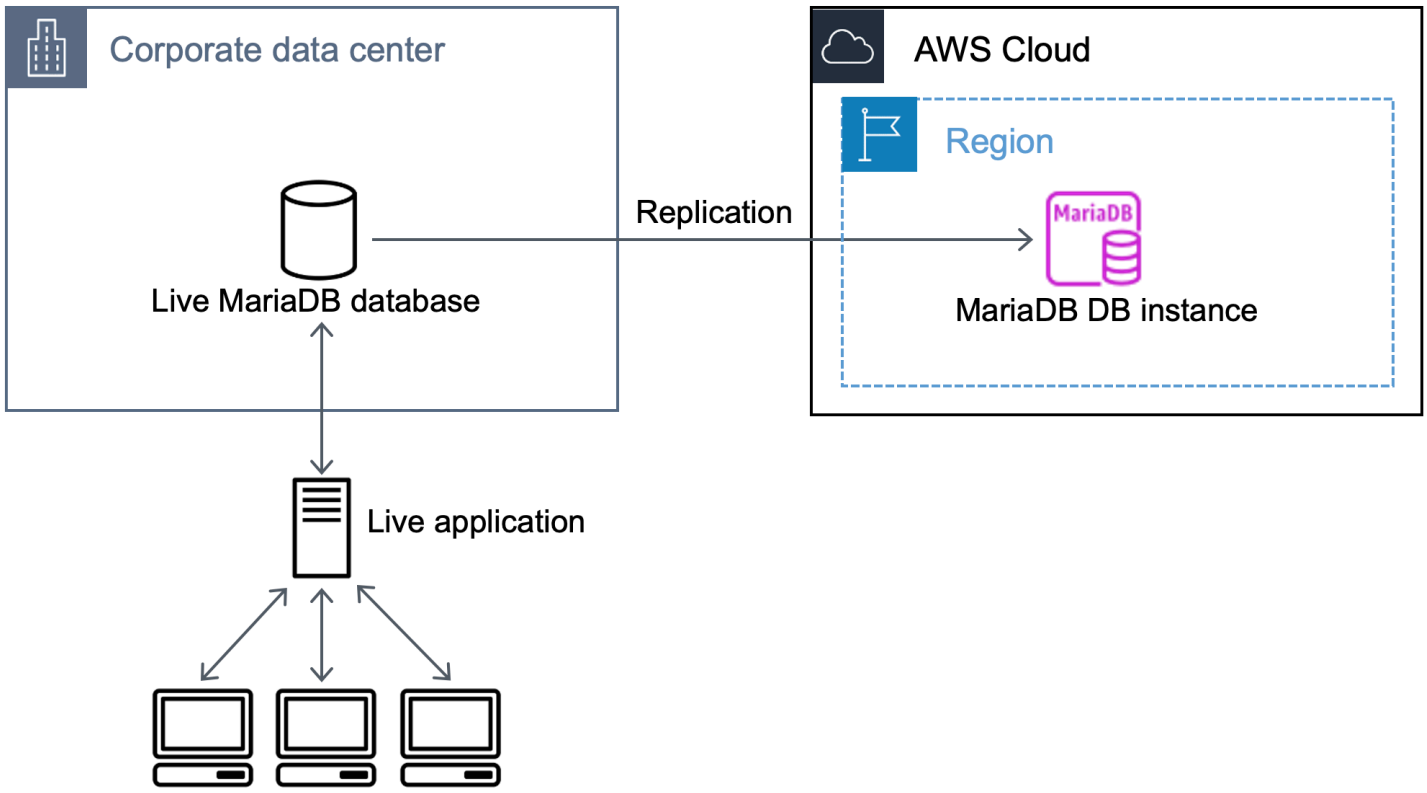
     > **Note**
     >
     > 最初にテーブルをダンプした際、mysqldump または mariadb-dump でデータをフォーマットするオプションを使用した場合は、データファイルのコンテンツが適切に解釈できるように、LOAD DATA LOCAL INFILE でも必ず同じオプションを使用する必要があります。

8. インポートしたデータベースの単一のテーブルまたは 2 つのテーブルに対してシンプルな SELECT クエリを実行して、インポートが正常に完了したかを検証します。

この手順で使用された Amazon EC2 インスタンスが今後不要な場合は、EC2 インスタンスを終了して、AWS リソース使用率を減らします。EC2 インスタンスを終了するには、「Amazon Elastic Compute Cloud ユーザーガイド」の「インスタンスの終了」を参照してください。

タスク 4: 外部データベースから新しい Amazon RDS データベースにデータをレプリケートする

ソースデータベースは、データをコピーして MariaDB データベースに転送するまでに更新された可能性があります。レプリケーションを使用して、コピーしたデータベースをソースデータベースで最新のものにすることができます。



Amazon RDS データベースでレプリケーションを開始するために必要なアクセス許可は限定されており、Amazon RDS マスターユーザーは利用できません。このため、適切な Amazon RDS ストアドプロシージャを使用します。

- mysql.rds_set_external_master
- レプリケーションを設定するには mysql.rds_set_external_master_gtid、レプリケーションを開始するには mysql.rds_start_replication

レプリケーションをスタートするには

タスク 1 のレプリケーションオプションの設定時に、バイナリログ記録を有効にし、ソースデータベースに一意のサーバー ID を設定しました。これで、ライブデータベースをソースレプリケーションインスタンスとして、Amazon RDS データベースをレプリカとしてセットアップできます。

1. Amazon RDS コンソールで、ソースデータベースをホストするサーバーの IP アドレスを Amazon RDS データベースの VPC セキュリティグループに追加します。VPC セキュリティグループの詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルールを設定する」を参照してください。

   ソースインスタンスと通信できるためには、Amazon RDS データベースの IP アドレスからの接続を許可するようにローカルネットワークを設定することも必要になる場合があります。Amazon RDS データベースの IP アドレスを確認するには、host コマンドを使用します。

   ```
   host host_name
   ```

   host_name は、Amazon RDS データベースのエンドポイントの DNS 名 (例: myinstance.123456789012.us-east-1.rds.amazonaws.com) です。エンドポイントの値は、Amazon RDS コンソールの DB インスタンスの詳細で確認できます。

2. 選択したクライアントを使用して、ソースインスタンスに接続し、レプリケーションに使用するユーザーを作成します。このアカウントはレプリケーション専用に使用され、セキュリティを強化するためにドメインに制限する必要があります。コマンドの例を次に示します。

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

3. ソースインスタンスについて、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。たとえば、すべてのデータベースに対する REPLICATION CLIENT と REPLICATION SLAVE の特権を、「repl_user」ユーザーに付与するには、次のコマンドを実行します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

4. SQL 形式を使用してバックアップファイルを作成しており、外部インスタンスが MariaDB 10.0.24 以降でない場合は、次のコマンドを実行してファイルのコンテンツを表示します。

```
cat backup.sql
```

このファイルに、マスターログファイルの名前と場所を示す CHANGE MASTER TO コメントが含まれています。--master-data で mysqldump オプションを使用した場合に、バックアップファイルにこのコメントが含まれます。MASTER_LOG_FILE と MASTER_LOG_POS の値に注意してください。

```
--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

バックアップファイルの作成に区切りテキスト形式を使用しており、外部のインスタンスが MariaDB 10.0.24 以降でない場合は、既存のデータベースのバックアップコピーの作成時に、タスク 1 のステップ 1 でバイナリログ座標を既に取得しているはずです。

外部のインスタンスが MariaDB 10.0.24 以降の場合は、既存のデータベースのバックアップコピーの作成時に、タスク 1 のステップ 2 でレプリケーションを開始するための GTID を既に取得しているはずです。

5. Amazon RDS データベースをレプリカにします。外部のインスタンスが MariaDB 10.0.24 以降でない場合、マスターユーザーとして Amazon RDS データベースに接続し、mysql.rds_set_external_master ストアドプロシージャを使用して、ソースのレプリケーションインスタンスとしてソースデータベースを特定します。

SQL 形式のバックアップファイルがある場合は、ステップ 4 で確認したマスターログのファイル名とマスターログの位置を使用します。区切り文字テキスト形式を使用した場合は、バックアップファイルの作成時に決定した名前と場所を使用します。コマンドの例を次に示します。

```
CALL mysql.rds_set_external_master ('myserver.mydomain.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

外部のインスタンスが MariaDB 10.0.24 以降の場合、マスターユーザーとして Amazon RDS データベースに接続し、mysql.rds_set_external_master_gtid ストアドプロシージャを使用して、ソースのレプリケーションインスタンスとしてソースデータベースを特定します。既存のデータベースのバックアップコピーの作成時に、タスク 1 のステップ 2 で確認した GTID を使用します。コマンドの例を次に示します。

```
CALL mysql.rds_set_external_master_gtid ('source_server_ip_address', 3306,
 'ReplicationUser', 'password', 'GTID', 1);
```

source_server_ip_address は、ソースレプリケーションインスタンスの IP アドレスです。EC2 プライベート DNS アドレスは現在サポートされていません。

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

6. Amazon RDS データベースでレプリケーションを開始するには、mysql.rds_start_replication ストアドプロシージャを使用する次のコマンドを実行します。

```
CALL mysql.rds_start_replication;
```

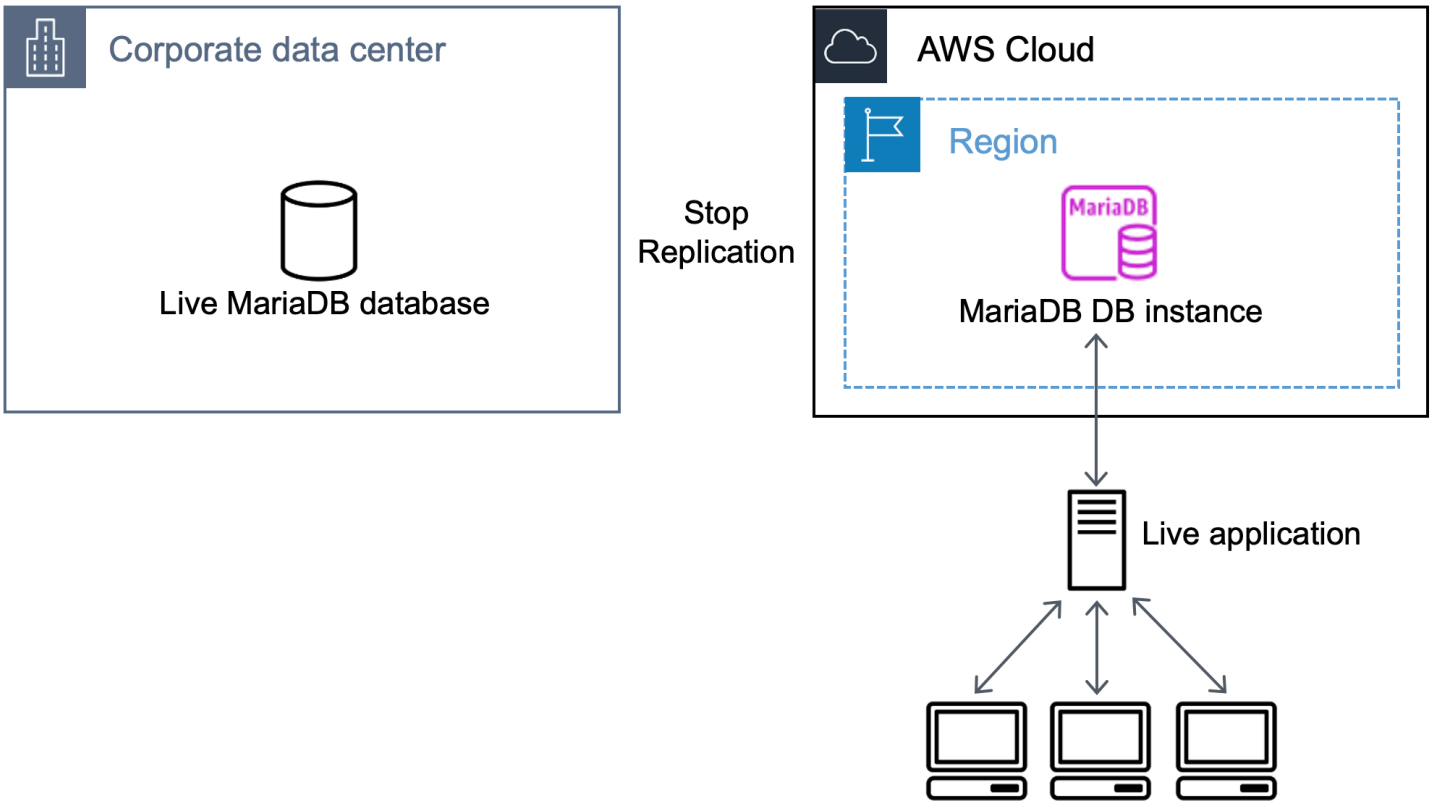
7. Amazon RDS データベースで、レプリカがいつソースレプリケーションインスタンスの最新の状態に更新されるかを特定するには、SHOW REPLICA STATUS コマンドを実行します。SHOW REPLICA STATUS コマンドの結果には、Seconds_Behind_Master フィールドが含まれます。Seconds_Behind_Master フィールドが 0 を返す場合、レプリカはソースレプリケーションインスタンスで最新の状態になります。

   MariaDB 10.5 以降の DB インスタンスの場合は、SQL コマンドの代わりに mysql.rds_replica_status ストアドプロシージャを使用します。

8. Amazon RDS データベースが最新の状態になったら、必要に応じてデータベースを復元できるように、自動バックアップを有効にします。Amazon RDS コンソールを使用して、Amazon RDS データベースの自動バックアップを有効化または変更できます。詳細については、「バックアップの概要」を参照してください。

タスク 5: ライブアプリケーションを Amazon RDS インスタンスにリダイレクトする

MariaDB データベースがソースレプリケーションインスタンスで最新の状態になったら、ライブアプリケーションを更新して、Amazon RDS インスタンスを使用できます。



ライブアプリケーションを MariaDB データベースにリダイレクトしてレプリケーションを停止するには

1. Amazon RDS データベースの VPC セキュリティグループを追加するには、アプリケーションをホストするサーバーの IP アドレスを追加します。VPC セキュリティグループの変更の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルールを設定する」を参照してください。

2. SHOW REPLICA STATUS コマンドの結果の Seconds_Behind_Master フィールドが 0 であることを確認します。この値は、レプリカがソースレプリケーションインスタンスの最新の状態であることを示します。

```
SHOW REPLICA STATUS;
```

   MariaDB 10.5 以降の DB インスタンスの場合は、SQL コマンドの代わりに mysql.rds_replica_status プロシージャを使用します。

3. トランザクションが終了したら、ソースへのすべての接続を閉じます。
4. Amazon RDS データベースを使用するようにアプリケーションを更新します。この更新には、一般に、Amazon RDS データベースのホスト名とポート、接続に使用するユーザーアカウントとパスワード、および使用するデータベースを識別する接続設定の変更が含まれます。
5. DB インスタンスに接続します。
6. mysql.rds_stop_replication ストアドプロシージャを使用する次のコマンドを実行して、Amazon RDS インスタンスのレプリケーションを停止します。

```
CALL mysql.rds_stop_replication;
```

7. Amazon RDS データベースで mysql.rds_reset_external_master ストアドプロシージャを使用する次のコマンドを実行して、レプリケーション設定をリセットします。これにより、このインスタンスはレプリカとして識別されなくなります。

```
CALL mysql.rds_reset_external_master;
```

8. マルチ AZ のサポートやリードレプリカなど、Amazon RDS のその他の機能を有効にします。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」および「DB インスタンスのリードレプリカの操作」を参照してください。

任意のソースから Amazon RDS for MariaDB DB インスタンスにデータをインポートする

Amazon RDS を使用すると、既存の MariaDB データを任意のソースから RDS for MariaDB DB インスタンスに移行できます。オンプレミスデータベース、他のクラウドプロバイダー、または既存の RDS for MariaDB DB インスタンスからターゲット RDS for MariaDB DB インスタンスにデータを転送できます。この機能を使用すると、データベースの統合、ディザスタリカバリソリューションの実装、セルフマネージド型データベースからの移行を行うことができます。一般的なシナリオには、

セルフホスト型の MariaDB サーバーからフルマネージド型の Amazon RDS DB インスタンスへの移行、複数の MariaDB データベースの 1 つの DB インスタンスへの統合、本番データを使用したテスト環境の作成などがあります。以下のセクションでは、mariadb-dump、バックアップファイル、レプリケーションなどのメソッドを使用して MariaDB データをインポートする詳細な手順について説明します。

ステップ 1: ロードするデータを含むフラットファイルを作成する

カンマ区切り値 (CSV) などの一般的な形式を使用して、ロードするデータを保存します。各テーブルには独自のファイルが必要です。複数のテーブルのデータを同じファイルに結合することはできません。各ファイルに、対応するテーブルと同じ名前を付けます。ファイル拡張子は何でもかまいません。例えば、テーブル名が sales の場合、ファイル名は sales.csv または sales.txt になります。

可能であれば、ロードされるテーブルのプライマリキーでデータをソートします。これにより、ロード時間が大幅に短縮され、ディスクストレージの要件が最小限に抑えられます。

この手順の速度と効率は、ファイルのサイズを小さく保つかどうかによって決まります。個々のファイルの非圧縮サイズが 1 GiB を超える場合、複数のファイルに分割して、1 つずつロードしてください。

Unix のようなシステム (Linux など) では、split コマンドを使用します。例えば、次のコマンドでは sales.csv ファイルが 1 GiB 未満の複数のファイルに分割されます。ただし、分割されるのは改行でのみです (-C 1024m)。新しいファイルの名前には、昇順の数値サフィックスが含まれます。次のコマンドは、sales.part_00 や sales.part_01 などの名前のファイルを生成します。

```
split -C 1024m -d sales.csv sales.part_
```

他のオペレーティングシステムにも同様のユーティリティを使用できます。

フラットファイルはどこにでも保存できます。ただし、ステップ 5 でデータをロードするときは、ファイルが存在するのと同じ場所から mysql シェルを呼び出すか、LOAD DATA LOCAL INFILE の実行時にファイルの絶対パスを使用する必要があります。

ステップ 2: ターゲット DB インスタンスにアクセスしているすべてのアプリケーションを停止する

大きなロードをスタートする前に、ロード先のターゲット DB インスタンスにアクセスするすべてのアプリケーションアクティビティを停止します。特に、他のセッションでロード中のテーブルや参照するテーブルを変更する場合は、これをお勧めします。これにより、ロード中に発生する制約違

反のリスクが軽減され、ロードパフォーマンスが向上します。また、ロードに関係しないプロセスによって行われた変更を失うことなく、DB インスタンスをロードの直前の時点に復元することもできます。

もちろん、これは可能でない場合や実用的ではない場合があります。アプリケーションによる DB インスタンスへのアクセスをロード前に停止できない場合は、データの可用性と整合性を確保するための手順を実行してください。必要となる具体的なステップは、実際のユースケースとサイト要件によって大きく異なります。

ステップ 3: DB スナップショットを作成する

データが含まれない新しい DB インスタンスにデータをロードする場合、このステップをスキップできます。それ以外の場合は、データロードの前後にターゲットの Amazon RDS DB インスタンスの DB スナップショットを作成することをお勧めします。Amazon RDS DB スナップショットは DB インスタンスの完全なバックアップであり、DB インスタンスを既知の状態に復元するために使用できます。DB スナップショットをスタートすると、データベースのバックアップのため DB インスタンスに対する I/O 操作が一時停止されます。

ロード直前に DB スナップショットを作成すると、必要が生じた場合にデータベースをロード前の状態に復元できます。ロード直後に DB スナップショットを作成しておくと、何らかの事故のときにそのスナップショットを使用すれば、データを再ロードせずに済みます。ロード後に DB スナップショットを使用して、新しいデータベースインスタンスにデータをインポートすることもできます。

次の例では、AWS CLI の create-db-snapshot コマンドを実行して、`AcmeRDS` インスタンスの DB スナップショットを作成し、その DB スナップショットに識別子 `"preload"` を指定します。

Linux、macOS、Unix の場合:

```
aws rds create-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds create-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

DB スナップショットからの復元機能を使用して、リハーサル用のテスト DB インスタンスを作成したり、ロード中に加えられた変更を元に戻すこともできます。

DB スナップショットからデータベースを復元すると、すべての DB インスタンスと同様に一意の識別子とエンドポイントを持つ新しい DB インスタンスが作成される点に留意してください。エンドポイントを変更せずに DB インスタンスを復元するには、エンドポイントを再利用できるように、まず DB インスタンスを削除します。

例えば、リハーサルや他のテスト用の DB インスタンスを作成するには、DB インスタンスに独自の識別子を指定します。この例での識別子は「AcmeRDS-2」です。この例では、AcmeRDS-2 に関連付けられたエンドポイントを使用して DB インスタンスに接続します。詳細については、「restore-db-instance-from-db-snapshot」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS-2 \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS-2 ^
    --db-snapshot-identifier preload
```

既存のエンドポイントを再利用するには、まず DB インスタンスを削除してから、復元されたデータベースに同じ識別子を指定します。詳細については、「delete-db-instance」を参照してください。

次の例でも、DB インスタンスを削除する前に最終的な DB スナップショットを取得しています。これはオプションですが推奨されます。

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier AcmeRDS \
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds delete-db-instance ^
```

```
    --db-instance-identifier AcmeRDS ^
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

ステップ 4 (オプション): Amazon RDS 自動バックアップをオフにする

> **Warning**
>
> ポイントインタイムリカバリを実行する必要がある場合は、自動バックアップをオフにしないでください。

自動バックアップをオフにすると、パフォーマンスが最適化します。これはデータのロードに必須ではありません。自動バックアップをオフにすると、既存のバックアップがすべて消去されます。その結果、自動バックアップをオフにすると、ポイントインタイムリカバリができなくなります。自動バックアップを無効にしても、手動 DB スナップショットに影響が及ぶことはありません。既存のすべての手動 DB スナップショットは引き続き復元で使用可能です。

自動バックアップを無効にするとロード時間が約 25% 短縮し、ロード時に必要なストレージ容量が減少します。データが含まれない新しい DB インスタンスにデータをロードする場合、バックアップを無効にすると、ロードを簡単に高速化でき、バックアップに必要な追加のストレージを使用する必要がなくなります。しかし、状況によっては、既にデータが含まれている DB インスタンスにロードする場合もあります。その場合、バックアップを無効にするメリットと、ポイントインタイムリカバリを実行できなくなることの影響を比較検討する必要があります。

DB インスタンスでは、デフォルトで自動バックアップが有効になっています (保持期間は 1 日です)。自動バックアップを無効にするには、バックアップ保持期間を 0 に設定します。ロード後、バックアップ保持期間を 0 以外の値に設定することでバックアップを再度有効にすることができます。バックアップを有効または無効にするには、Amazon RDS で DB インスタンスをシャットダウンおよび再起動して、MariaDB の口グ記録を有効または無効にします。

AWS CLI の `modify-db-instance` コマンドを実行し、バックアップ保持期間を 0 に設定して変更をすぐに適用します。保持期間を 0 に設定するには DB インスタンスを再起動する必要があるため、続行前に再起動が完了するまで待っています。詳細については、「modify-db-instance」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --apply-immediately \
    --backup-retention-period 0
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --apply-immediately ^
    --backup-retention-period 0
```

AWS CLI の describe-db-instances コマンドで DB インスタンスのステータスを確認できます。次の例では、AcmeRDS DB インスタンスにおける DB インスタンスのステータスを表示します。

```
aws rds describe-db-instances --db-instance-identifier AcmeRDS --query "*[].
{DBInstanceStatus:DBInstanceStatus}"
```

DB インスタンスのステータスが available になったら、次のステップに進む準備が整いました。

ステップ 5: データをロードする

フラットファイルから行を読み取ってデータベーステーブルにロードするには、MariaDB の LOAD DATA LOCAL INFILE ステートメントを使用します。

> **Note**
>
> フラットファイルが存在するのと同じ場所から mariadb シェルを呼び出すか、LOAD DATA LOCAL INFILE の実行時にファイルの絶対パスを使用する必要があります。

次の例は、sales.txt という名前のファイルからデータベース内の Sales という名前のテーブルにデータをロードする方法を示しています。

```
MariaDB [(none)]> LOAD DATA LOCAL INFILE 'sales.txt' INTO TABLE Sales FIELDS TERMINATED
 BY ' ' ENCLOSED BY '' ESCAPED BY '\\';
Query OK, 1 row affected (0.01 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

LOAD DATA ステートメントの詳細については、「MariaDB ドキュメント」の「LOAD DATA INFILE」を参照してください。

ステップ 6: Amazon RDS 自動バックアップを再度オンにする

ステップ 4 で Amazon RDS 自動バックアップをオフにした場合は、ロードが完了したら、バックアップ保持期間をロード前の値に戻して、自動バックアップをオンにします。ステップ 4 で説明したように、Amazon RDS は DB インスタンスを再起動するため、短時間の停止に備えてください。

次の例では、AWS CLI の modify-db-instance コマンドを実行して、AcmeRDS DB インスタンスの自動バックアップをオンにします。保持期間は 1 日に設定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --backup-retention-period 1 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --backup-retention-period 1 ^
    --apply-immediately
```

Amazon RDS での MariaDB のレプリケーションの使用

リードレプリカは通常、Amazon RDS の DB インスタンス間でレプリケーションを設定するために使用します。リードレプリカの概要については、「DB インスタンスのリードレプリカの操作」を参照してください。Amazon RDS for MariaDB でリードレプリカを操作する具体的な方法については、MariaDB リードレプリカの使用 を参照してください。

レプリケーションは、MariaDB DB インスタンスのバイナリログ座標に基づいて設定することもできます。MariaDB インスタンスでは、グローバルトランザクション ID (GTIDs) に基づいてより高度な耐クラッシュ性を提供するレプリケーションを設定できます。詳細については、「外部ソースインスタンスを使用した GTID ベースのレプリケーションを設定する」を参照してください。

以下は RDS for MariaDB で使用できる他のレプリケーションオプションです。

- RDS for MariaDB DB インスタンスと Amazon RDS の外部にある MySQL または MariaDB インスタンスとの間でレプリケーションを設定できます。外部ソースとのレプリケーションの設定については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。
- Amazon RDS の外部にある MySQL または MariaDB インスタンスからデータベースをインポートしたり、これらのインスタンスにデータベースを エクスポートしたりするようにレプリケーションを設定できます。詳細については、「ダウンタイムを短縮して Amazon RDS for MariaDB DB インスタンスにデータをインポートする」および「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

これらのどのレプリケーションオプションでも、行ベース、ステートメントベース、または混合レプリケーションが使用できます。行ベースのレプリケーションは、SQL ステートメントの結果として変更された行のみをレプリケートします。ステートメントベースのレプリケーションは、SQL ステートメント全体をレプリケートします。混合レプリケーションは、可能な場合にはステートメントレプリケーションを使用しますが、ステートメントベースのレプリケーションに対して安全でない SQL ステートメントが実行されると、行ベースのレプリケーションに切り替えます。ほとんどの場合には、混合レプリケーションをお勧めします。DB インスタンスのバイナリログ形式は、レプリケーションが行ベース、ステートメントベース、混合のいずれであるかを判断します。バイナリログ形式の設定については、「MariaDB バイナリログ記録の設定」を参照してください。

MariaDB バージョン間のレプリケーションの互換性については、MariaDB ドキュメントの「レプリケーションの互換性」を参照してください。

トピック

- MariaDB リードレプリカの使用
- 外部ソースインスタンスを使用した GTID ベースのレプリケーションを設定する
- 外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

MariaDB リードレプリカの使用

以下では、Amazon RDS for MariaDB でのリードレプリカの使用に関する特定の情報を確認することができます。リードレプリカと使用手順の概要については、「DB インスタンスのリードレプリカの操作」を参照してください。

- MariaDB を使用したレプリケーションフィルターの設定
- MariaDB での遅延レプリケーションの設定
- MariaDB でのリードレプリカの更新
- MariaDB でのマルチ AZ リードレプリカのデプロイの使用
- RDS for MariaDB でのカスケードリードレプリカの使用
- MariaDB リードレプリカのモニタリング
- MariaDB リードレプリカでのレプリケーションの開始と停止
- MariaDB リードレプリカの問題のトラブルシューティング

MariaDB でのリードレプリカの設定

MariaDB DB インスタンスがレプリケーションソースとして機能するには、バックアップ保持期間を 0 以外の値に設定することにより、ソース DB インスタンスで自動バックアップを有効にしてください。この要件は、別のリードレプリカのソース DB インスタンスであるリードレプリカにも適用されます。

同一リージョン内の 1 つの DB インスタンスから、最大 15 個のリードレプリカを作成できます。レプリケーションを効率的に行うには、各リードレプリカにソースの DB インスタンスと同程度のコンピューティングリソースとストレージリソースが必要です。ソースの DB インスタンスをスケールした場合は、リードレプリカもスケールする必要があります。

RDS for MariaDB では、リードレプリカのカスケードをサポートしています。リードレプリカのカスケードを設定する方法については、「RDS for MariaDB でのカスケードリードレプリカの使用」を参照してください。

同じソースの DB インスタンスを参照する複数のリードレプリカの作成や削除の操作は同時に実行できます。その操作を実行するには、ソースインスタンスごとに作成できるリードレプリカを 15 個に制限します。

MariaDB を使用したレプリケーションフィルターの設定

レプリケーションフィルターを使用して、リードレプリカでレプリケートするデータベースとテーブルを指定できます。レプリケーションフィルターは、データベースとテーブルをレプリケーションに含めることも、レプリケーションから除外することもできます。

レプリケーションフィルターの使用例は以下のとおりです。

- リードレプリカのサイズを縮小します。レプリケーションフィルタリングを使用すると、リードレプリカで必要のないデータベースとテーブルを除外できます。
- セキュリティ上の理由から、データベースとテーブルをリードレプリカから除外するため。
- 異なるリードレプリカで、特定のユースケースごとにさまざまなデータベースとテーブルを複製するため。例えば、分析やシャーディングに特定のリードレプリカを使用できます。
- 異なる AWS リージョン にリードレプリカがある DB インスタンスで、異なる AWS リージョンに異なるデータベースまたはテーブルを複製する場合。

> **Note**
>
> また、レプリケーションフィルターを使用して、インバウンドのレプリケーショントポロジでレプリカとして設定されているプライマリ MariaDB DB インスタンスでレプリケートするデータベースとテーブルを指定することもできます。この設定の詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

トピック

- RDS for MariaDB のレプリケーションフィルターパラメータの設定
- RDS for MariaDB のレプリケーションフィルターの制限
- RDS for MariaDB のレプリケーションフィルターの例
- リードレプリカのレプリケーションフィルターを表示する

RDS for MariaDB のレプリケーションフィルターパラメータの設定

レプリケーションフィルターを構成するには、リードレプリカに次のレプリケーションフィルターのパラメータを設定します。

- replicate-do-db - 指定したデータベースに変更を複製します。リードレプリカに対してこのパラメータを設定すると、パラメータで指定されたデータベースのみが複製されます。
- replicate-ignore-db - 指定したデータベースに変更を複製しないでください。リードレプリカに replicate-do-db パラメータが設定されている場合、このパラメータは評価されません。
- replicate-do-table -指定されたテーブルに変更を複製します。このパラメータをリードレプリカに設定した場合、パラメータで指定したテーブルのみが複製されます。また、replicate-do-db パラメータまたは replicate-ignore-db パラメータを設定する場合は、指定されたテーブルを含むデータベースをリードレプリカのレプリケーションに含める必要があります。
- replicate-ignore-table – 指定したテーブルに変更を複製しないでください。リードレプリカに replicate-do-table パラメータが設定されている場合、このパラメータは評価されません。
- replicate-wild-do-table - 指定したデータベースおよびテーブル名のパターンに基づいてテーブルを複製します。% および _ ワイルドカードの文字がサポート対象となります。replicate-do-db または replicate-ignore-db パラメータが設定されている場合は、リードレプリカを使用して、指定したテーブルを含むデータベースをレプリケーションに含めるようにしてください。
- replicate-wild-ignore-table - 指定したデータベースおよびテーブル名のパターンに基づいてテーブルを複製しないでください。% および _ ワイルドカードの文字がサポート対象となります。リードレプリカに replicate-do-table または replicate-wild-do-table パラメータが設定されている場合、このパラメータは評価されません。

パラメータは、記載されている順序に沿って評価されます。これらのパラメータの詳細な仕組みについては、MariaDB のドキュメントを参照してください。

デフォルトでは、これらの各パラメータの値は空です。各リードレプリカで、これらのパラメータを使用してレプリケーションフィルターを設定、変更、削除することができます。これらのパラメータの 1 つを設定する場合は、各フィルターを他のフィルターとコンマで区切ります。

% および _ パラメータで replicate-wild-do-table および replicate-wild-ignore-table ワイルドカードの文字を使用できます。% ワイルドカードは任意の文字数と一致し、_ ワイルドカードは 1 文字のみと一致します。

ソース DB インスタンスのバイナリログ形式は、データ変更のレコードを決定するため、レプリケーションでは重要です。binlog_format パラメータの設定により、レプリケーションが行ベースかステートメントベースかが決まります。詳細については、「MariaDB バイナリログ記録の設定」を参照してください。

> **Note**
>
> ソース DB インスタンスの binlog_format 設定に関係なく、すべてのデータ定義言語 (DDL) ステートメントはステートメントとして複製されます。

RDS for MariaDB のレプリケーションフィルターの制限

RDS for MariaDB のレプリケーションフィルターには、次の制限が適用されます。

- 各レプリケーションフィルターのパラメータには、2,000 文字といった制限があります。
- レプリケーションフィルターでは、カンマはサポートされていません。
- MariaDB binlog_do_db とバイナリログフィルターの binlog_ignore_db オプションはサポートされていません。
- レプリケーションフィルタリングは、XA トランザクションをサポートしていません。

  詳細については、MySQL ドキュメントの「Restrictions on XA Transactions」を参照してください。
- RDS for MariaDB バージョン 10.2 では、レプリケーションフィルターはサポートされていません。

RDS for MariaDB のレプリケーションフィルターの例

リードレプリカのレプリケーションフィルタリングを構成するには、リードレプリカに関連付けられているパラメータグループのレプリケーションフィルタリングパラメータを変更します。

> **Note**
>
> デフォルトのパラメータグループを変更することはできません。リードレプリカがデフォルトのパラメータグループを使用している場合は、新しいパラメータグループを作成してリードレプリカに関連付けます。DB パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、パラメータグループのパラメータを設定できます。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。パラメータグループにパラメータを設定すると、そのパラメータグループに関連付けられているすべての DB インスタンスでパラメータ設定を使用します。パラメータグループにレプリケーションフィルターのパラメータを設定する場合は、パラメータグループがリードレプリカにのみ関連付けられていることを確認してください。ソース DB インスタンスのレプリケーションフィルターのパラメータは空のままにします。

次の例では、AWS CLI を使用してパラメータを設定します。これらの例では、CLI コマンドが完了した直後にパラメータの変更が行われるように `ApplyMethod` を `immediate` に設定しています。リードレプリカの再起動後に保留中の変更を適用する場合は、`ApplyMethod` を `pending-reboot` に設定します。

以下の例では、レプリケーションフィルターを設定します。

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Escaping wildcard characters in names
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Exampleレプリケーションにデータベースを含める

次の例では、レプリケーションに `mydb1` データベースと `mydb2` データベースが含まれています。リードレプリカに対して `replicate-do-db` を設定すると、パラメータで指定されたデータベースだけがレプリケートされます。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-do-db", "ParameterValue": "mydb1,mydb2",
 "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-do-db", "ParameterValue": "mydb1,mydb2",
 "ApplyMethod":"immediate"}]"
```

Exampleレプリケーションにテーブルを含める

次の例では、データベース mydb1 の table1 テーブルと table2 テーブルをレプリケーションに含めています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-do-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-do-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Exampleワイルドカードの文字を使用してレプリケーションにテーブルを含める

次の例では、データベース mydb の orders および returns で始まる名前のテーブルをレプリケーションに含めています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
```

```
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Example名前のワイルドカード文字のエスケープ

次の例は、エスケープ文字 \ を使用して、名前の一部であるワイルドカードの文字をエスケープする方法を示しています。

データベース mydb1 には my_table で始まる複数のテーブル名があり、これらのテーブルをレプリケーションに含めることを想定しています。テーブル名には、ワイルドカードの文字でもあるアンダースコアが含まれているため、この例ではテーブル名のアンダースコアをエスケープしています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue": "my
\_table%", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-wild-do-table", "ParameterValue": "my
\_table%", "ApplyMethod":"immediate"}]"
```

Exampleレプリケーションからデータベースを除外する

次の例では、mydb1 データベースと mydb2 データベースをレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-ignore-db", "ParameterValue":
 "mydb1,mydb2", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-ignore-db", "ParameterValue":
 "mydb1,mydb2", "ApplyMethod":"immediate"}]"
```

Exampleレプリケーションからテーブルを除外する

次の例では、データベース table1 のテーブル table2 と mydb1 をレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-ignore-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "[{"ParameterName": "replicate-ignore-table", "ParameterValue":
 "mydb1.table1,mydb1.table2", "ApplyMethod":"immediate"}]"
```

Exampleワイルドカードの文字を使用したレプリケーションからテーブルを除外する

次の例では、データベース mydb の orders および returns で始まる名前のテーブルをレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "[{"ParameterName": "replicate-wild-ignore-table", "ParameterValue":
 "mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
```

```
  --parameters "[{"ParameterName": "replicate-wild-ignore-table", "ParameterValue":
"mydb.orders%,mydb.returns%", "ApplyMethod":"immediate"}]"
```

リードレプリカのレプリケーションフィルターを表示する

リードレプリカのレプリケーションフィルターは、次の方法で表示できます。

- リードレプリカに関連付けられているパラメータグループのレプリケーションフィルタリングパラメータの設定を確認してください。

  手順については、「Amazon RDS のDB パラメータグループのパラメータ値の表示」を参照してください。
- MariaDB クライアントで、リードレプリカに接続し、SHOW REPLICA STATUS ステートメントを実行します。

  出力の次のフィールドには、リードレプリカのレプリケーションフィルターが表示されます。
  - Replicate_Do_DB
  - Replicate_Ignore_DB
  - Replicate_Do_Table
  - Replicate_Ignore_Table
  - Replicate_Wild_Do_Table
  - Replicate_Wild_Ignore_Table

  これらのフィールドの詳細については、MySQL のドキュメントの Checking Replication Status を参照してください。

> **Note**
>
> MariaDB の旧バージョンは、SHOW SLAVE STATUS ではなく SHOW REPLICA STATUS を使用していました。10.5 より前の MariaDB バージョンを使用している場合は、SHOW SLAVE STATUS を使用します。

MariaDB での遅延レプリケーションの設定

遅延レプリケーションは、災害対策用の戦略として使用できます。遅延レプリケーションでは、ソースからリードレプリカへのレプリケーションを遅延させる最小時間を秒数で指定します。障害発生時 (意図しないテーブルの削除など) には、以下のステップを実行して障害から早急に復旧します。

- 障害を起こした変更がリードレプリカに送られる前に、リードレプリカへのレプリケーションを停止します。

  レプリケーションを停止するには、mysql.rds_stop_replication ストアドプロシージャを使用します。
- 「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順を使用してリードレプリカを新しいソースの DB インスタンスに昇格させます。

> **Note**
>
> - 遅延レプリケーションは MariaDB 10.6 以降でサポートされています。
> - 遅延レプリケーションを設定するには、ストアドプロシージャを使用します。遅延レプリケーションを AWS マネジメントコンソール、AWS CLI、または Amazon RDS API で設定することはできません。
> - 遅延レプリケーションの設定でグローバルなトランザクション識別子 (GTID) に基づくレプリケーションを使用できます。

トピック

- リードレプリカ作成時の遅延レプリケーションの設定
- 既存のリードレプリカの遅延レプリケーションの変更
- リードレプリカの昇格

リードレプリカ作成時の遅延レプリケーションの設定

DB インスタンスから今後作成するリードレプリカの遅延レプリケーションを設定するには、mysql.rds_set_configuration パラメータを指定して `target delay` ストアドプロシージャを実行します。

リードレプリカの作成時に遅延レプリケーションを設定するには

1. MariaDB クライアントを使用して、マスターユーザーとしてリードレプリカのソースとなる MariaDB DB インスタンスに接続します。
2. mysql.rds_set_configuration パラメータを指定して `target delay` ストアドプロシージャを実行します。

例えば、現在の DB インスタンスから作成されるリードレプリカへのレプリケーションを少なくとも 1 時間 (3600 秒) 遅延させるように指定するには、次のストアドプロシージャを実行します。

```
call mysql.rds_set_configuration('target delay', 3600);
```

> **Note**
>
> このストアドプロシージャを実行すると、AWS CLI または Amazon RDS API を使用して作成したリードレプリカには、指定した秒数で遅延するレプリケーションが設定されます。

既存のリードレプリカの遅延レプリケーションの変更

既存のリードレプリカの遅延レプリケーションを変更するには、mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降) ストアドプロシージャを実行します。

既存のリードレプリカの遅延レプリケーションを変更するには

1. MariaDB クライアントを使用して、マスターユーザーとしてリードレプリカに接続します。
2. レプリケーションを停止するには、mysql.rds_stop_replication ストアドプロシージャを使用します。
3. mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降) ストアドプロシージャを実行します。

   例えば、リードレプリカへのレプリケーションを少なくとも 1 時間 (3600 秒) 遅延させるように指定するには、次のストアドプロシージャを実行します。

   ```
   call mysql.rds_set_source_delay(3600);
   ```
4. mysql.rds_start_replication ストアドプロシージャを使用してレプリケーションを開始します。

リードレプリカの昇格

レプリケーションが停止したら、災害対策シナリオでリードレプリカを新しいソース DB インスタンスに昇格させます。リードレプリカの昇格については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

MariaDB でのリードレプリカの更新

リードレプリカは読み取りクエリをサポートするように設計されていますが、ときどき更新が必要になることがあります。例えば、インデックスを追加して、レプリカにアクセスする特定のタイプのクエリを高速化する必要が生じることがあります。更新を有効にするには、リードレプリカの DB パラメータグループで `read_only` パラメータを 0 に設定します。

MariaDB でのマルチ AZ リードレプリカのデプロイの使用

リードレプリカは、シングル AZ DB インスタンス配置からもマルチ AZ DB インスタンス配置からも作成できます。重要なデータの耐久性の高いと可用性を高めるにはマルチ AZ 配置を使用しますが、読み取り専用クエリを処理するためにマルチ AZ セカンダリを使用することはできません。代わりに、トラフィックの多いマルチ AZ DB インスタンスのリードレプリカを作成して、読み取り専用クエリをオフロードできます。マルチ AZ 配置のソースのインスタンスがセカンダリにフェイルオーバーすると、関連付けられているすべてのリードレプリカが (プライマリから) セカンダリをレプリケーションのソースとして使用するように自動的に切り替わります。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

リードレプリカは、マルチ AZ DB インスタンスとして作成できます。Amazon RDS では、レプリカのフェイルオーバーをサポートするため、別のアベイラビリティーゾーンにレプリカのスタンバイを作成します。リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。

RDS for MariaDB でのカスケードリードレプリカの使用

RDS for MariaDB では、リードレプリカのカスケードをサポートしています。カスケードリードレプリカにより、ソースの RDS for MariaDB DB インスタンスにオーバーヘッドを追加せずに読み取りをスケーリングできます。

カスケードリードレプリカを使用すると、RDS for MariaDB DB インスタンスは、チェーン内の最初のリードレプリカにデータを送信します。その後、そのリードレプリカは、チェーン内の 2 番目のレプリカにデータを送信し、その動作が順に続いていきます。その結果、チェーン内のすべてのリードレプリカに RDS for MariaDB DB インスタンスの更新が送信されますが、ソース DB インスタンスでのオーバーヘッドは発生しません。

ソースの RDS for MariaDB DB インスタンスから、チェーン内にリードレプリカを 3 層まで作成できます。例えば、RDS for MariaDB DB インスタンス、`mariadb-main` があるとします。以下の操作を行うことができます。

- mariadb-main で開始し、チェーン内に最初のリードレプリカ、read-replica-1 を作成します。
- 次に、read-replica-1 で、チェーン内に次のリードレプリカ、read-replica-2 を作成します。
- 最後に、read-replica-2 で、チェーン内に 3 番目のリードレプリカ、read-replica-3 を作成します。

mariadb-main の層では、この 3 番目のカスケードリードレプリカに続く、別のリードレプリカを作成することはできません。一連の完全なインスタンス (RDS for MariaDB のソース DB インスタンスから、この層の最後のカスケードリードレプリカまで) は、最大 4 つの DB インスタンスで構成できます。

リードレプリカのカスケードを設定するには、RDS for MariaDB DB インスタンスで自動バックアップを有効にします。リードレプリカで自動バックアップを有効にするには、まずリードレプリカを作成し、次に自動バックアップを有効にするようにリードレプリカを変更します。詳細については、「リードレプリカの作成」を参照してください。

他のリードレプリカと同様に、カスケードの一部となっているリードレプリカを昇格できます。リードレプリカのチェーン内でリードレプリカを昇格させると、そのレプリカはチェーンから削除されます。例えば、mariadb-main DB インスタンスのワークロードの一部を、経理部のみが使用する新しいインスタンスに移動するとします。この例では、3 つのリードレプリカから成るチェーンがあると仮定し、read-replica-2 を昇格させることにします。チェーンは以下のような影響を受けます。

- 昇格する read-replica-2 は、レプリケーションチェーンから削除されます。
  - このリードレプリカは、完全な読み取り/書き込み DB インスタンスになります。
  - 昇格前と同じように、read-replica-3 へのレプリケーションを継続します。
- mariadb-main は、read-replica-1 へのレプリケーションを継続します。

リードレプリカの昇格についての詳細は、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

MariaDB リードレプリカのモニタリング

MariaDB のリードレプリカでは、Amazon CloudWatch で Amazon RDS の ReplicaLag メトリクスを確認することでレプリケーションの遅延をモニタリングできます。ReplicaLag メトリクスに

は、Seconds_Behind_Master コマンドの SHOW REPLICA STATUS フィールドの値が報告されます。

> **Note**
>
> MariaDB の旧バージョンは、SHOW SLAVE STATUS ではなく SHOW REPLICA STATUS を使用していました。10.5 より前の MariaDB バージョンを使用している場合は、SHOW SLAVE STATUS を使用します。

MariaDB のレプリケーション遅延の一般的な原因は以下のとおりです。

- ネットワークが停止している。
- リードレプリカで、インデックスがあるテーブルに書き込んでいる。read_only パラメータがリードレプリカで 0 に設定されていない場合、レプリケーションが中断されることがあります。
- MyISAM などの非トランザクションストレージエンジンを使用している。レプリケーションは、MariaDB 上の InnoDB ストレージエンジンでのみサポートされます。

ReplicaLag メトリックが 0 に達すると、レプリカがソース DB インスタンスに追いついています。ReplicaLag メトリクスにより -1 が返された場合、レプリケーションは現在アクティブではありません。ReplicaLag = -1 は Seconds_Behind_Master = NULL と同等です。

MariaDB リードレプリカでのレプリケーションの開始と停止

システムのストアドプロシージャ mysql.rds_stop_replication および mysql.rds_start_replication を呼び出すことにより、Amazon RDS DB インスタンスでレプリケーションプロセスを停止して再スタートすることができます。これは、大きいインデックスの作成など、長時間実行されている操作の 2 つの Amazon RDS インスタンス間でレプリケーションするときに実行できます。レプリケーションは、データベースをインポートまたはエクスポートするときに停止してスタートする必要もあります。詳細については、「ダウンタイムを短縮して Amazon RDS for MariaDB DB インスタンスにデータをインポートする」および「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

レプリケーションを手動で停止するかレプリケーションエラーで停止してから連続して 30 日を超えると、Amazon RDS はソースの DB インスタンスとすべてのリードレプリカの間のレプリケーションを終了します。これは、ソース DB インスタンスでの所要ストレージの増大と長期間のフェイルオーバーを防ぐためです。リードレプリカの DB インスタンスは引き続き使用できます。ただし、レ

プリケーションが終了されるとリードレプリカに必要なバイナリログがソースの DB インスタンスから削除されるため、レプリケーションを再開することはできません。レプリケーションを再度行うには、ソースの DB インスタンスの新しいリードレプリカを作成します。

MariaDB リードレプリカの問題のトラブルシューティング

MariaDB のレプリケーションテクノロジーは非同期です。非同期であるため、ソースの DB インスタンスの `BinLogDiskUsage` やリードレプリカの `ReplicaLag` が増加する場合があります。例えば、ソース DB インスタンスへの大量の書き込みオペレーションは並行して実行できます。一方、リードレプリカへの書き込みオペレーションは単一の I/O スレッドでシリアルで行われるため、ソースのインスタンスとリードレプリカの間で遅延が発生する場合があります。MariaDB ドキュメントの読み取り専用のレプリカについては、「レプリケーションの概要」を参照してください。

ソースの DB インスタンスに対する更新とそれに続くリードレプリカに対する更新の間の遅延を低減するには、次のいくつかの方法があります。

- ストレージサイズと DB インスタンスクラスがソース DB インスタンスと同程度となるようにリードレプリカのサイズを決定します。
- ソース DB インスタンスとリードレプリカにより使用される DB パラメータグループのパラメータ設定に互換性を確保します。詳細と例については、このセクションの後方にある `max_allowed_packet` パラメータの説明を参照してください。

Amazon RDS は、リードレプリカのレプリケーションの状態をモニタリングし、何らかの理由でレプリケーションが停止した場合はリードレプリカのインスタンスの `Replication State` フィールドを `Error` に更新します。これには、リードレプリカで実行された DML クエリがソースの DB インスタンスで行われた更新と競合した場合などがあります。

[`Replication Error`] フィールドを参照することで、MariaDB エンジンによりスローされた関連エラーの詳細を確認できます。リードレプリカのステータスを示すイベントが生成されます (RDS-EVENT-0045、RDS-EVENT-0046、RDS-EVENT-0047 など) 。イベントについてとイベントへのサブスクライブの詳細については、「Amazon RDS イベント通知の操作」を参照してください。MariaDB エラーメッセージが返された場合、『MariaDB のエラーメッセージドキュメント』でエラーを確認してください。

レプリケーションエラーを引き起こす一般的な問題は、リードレプリカの `max_allowed_packet` パラメータの値がソース DB インスタンスの `max_allowed_packet` パラメータより小さいことです。`max_allowed_packet` パラメータは、DB パラメータグループに設定できるカスタムパラメータで、データベースで実行できる DML コードの最大サイズを指定するために使用されま

す。場合によっては、ソースの DB インスタンスに関連付けらている DB パラメータグループの max_allowed_packet パラメータの値が、ソースのリードレプリカに関連付けられている DB パラメータグループの max_allowed_packet パラメータの値より小さいことがあります。このような場合、レプリケーションプロセスからエラー (パケットが 'max_allowed_packet' バイトを超える) がスローされ、レプリケーションが停止することがあります。ソースとリードレプリカで同じ max_allowed_packet パラメータ値を持つ DB パラメータグループが使用されるように設定することにより、エラーを修正できます。

レプリケーションエラーを引き起こす可能性があります他の一般的な状況は次のとおりです。

- リードレプリカのテーブルに書き込んでいる。リードレプリカでインデックスを作成する場合、read_only パラメータを 0 に設定してインデックスを作成する必要があります。リードレプリカのテーブルに書き込んだ場合、レプリケーションが中断する可能性があります。
- MyISAM. などの非トランザクションストレージエンジンを使用している。リードレプリカにはトランザクションストレージエンジンが必要です。レプリケーションは、MariaDB 上の InnoDB ストレージエンジンでのみサポートされます。
- SYSDATE() など、安全でない非決定的クエリを使用している。詳細については、「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください。

エラーを安全にスキップできると判断した場合、RDS for MySQL の現在のレプリケーションエラーのスキップ で説明されているステップに従うことができます。それ以外の場合は、リードレプリカを削除し、同じ DB インスタンス識別子を使用してインスタンスを作成することで、エンドポイントを前のリードレプリカと同じままにすることができます。レプリケーションエラーが解決すると、[Replication State] は [レプリケーション中] に変化します。

MariaDB DB インスタンスでは、障害時にバイナリログ (binlog) のイベントがフラッシュされない場合、リードレプリカをセカンダリに切り替えられないことがあります。その場合、リードレプリカを手動で削除して作成し直します。次のパラメータ値 (sync_binlog=1、innodb_flush_log_at_trx_commit=1) を設定することで、これが発生する可能性を減らすことができます。これらの設定によりパフォーマンスが低下することがあるため、本稼働環境で変更を実装する前に影響をテストしてください。

外部ソースインスタンスを使用した GTID ベースのレプリケーションを設定する

バージョン 10.0.24 以降の外部 MariaDB インスタンスから RDS for MariaDB DB インスタンスへのグローバルなトランザクション識別子 (GTID) に基づいてレプリケーションを設定できま

す。Amazon RDS で外部ソースインスタンスとレプリカをセットアップする場合は、次のガイドラインに従ってください。

- レプリカである RDS for MariaDB の DB インスタンスのフェイルオーバーイベントをモニタリングします。フェイルオーバーが発生すると、レプリカである DB インスタンスが、新しいホスト上に別のネットワークアドレスで再作成されます。フェイルオーバーイベントをモニタリングする方法については、「Amazon RDS イベント通知の操作」を参照してください。
- ソースインスタンスのバイナリログ (binlog) は、それらがレプリカに適用されていることを確認するまで保持します。このメンテナンスによって、障害発生時にソースインスタンスを復元できます。
- Amazon RDS にある MariaDB DB インスタンスの自動バックアップを有効にします。自動バックアップを有効にすると、ソースインスタンスとレプリカを再同期する必要がある場合に、特定の時点にレプリカを復元できます。バックアップとポイントインタイム復元の詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

> **Note**
>
> MariaDB DB インスタンスでレプリケーションを開始するために必要なアクセス権限は限定されており、Amazon RDS マスターユーザーは利用できません。このため、Amazon RDS の mysql.rds_set_external_master_gtid コマンドと mysql.rds_start_replication コマンドを使用して、ライブデータベースと RDS for MariaDB データベースとのレプリケーションを設定する必要があります。

外部のソースインスタンスと Amazon RDS 上の MariaDB DB インスタンス間でレプリケーションを開始するには、次の手順に従います。

レプリケーションを開始するには

1. ソース MariaDB インスタンスを読み取り専用にします。

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. 外部 MariaDB インスタンスの現在の GTID を取得します。mysql または選択したクエリエディタを使用して SELECT @@gtid_current_pos; を実行することで、取得できます。

GTID は、<domain-id>-<server-id>-<sequence-id> の形式となります。一般的な GTID は **0-1234510749-1728** のようになります。GTID とコンポーネントパートの詳細については、MariaDB ドキュメントの「グローバルトランザクション ID」を参照してください。

3. mysqldump を使用して、外部 MariaDB インスタンスから MariaDB DB インスタンスにデータベースをコピーします。非常に大きなデータベースでは、「ダウンタイムを短縮して Amazon RDS for MariaDB DB インスタンスにデータをインポートする」の手順を使用することが必要になる場合があります。

   Linux、macOS、Unix の場合:

```
mysqldump \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
        -pRDS_password
```

   Windows の場合:

```
mysqldump ^
    --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p オプションと入力するパスワードの間にスペースがないことを確認します。

セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

--host コマンドで、--user (-u)、--port、-p、mysql オプションを使用して、MariaDB DB インスタンスに接続するためのホスト名、ユーザー名、ポート、パスワードを指定します。ホスト名は MariaDB DB インスタンスのエンドポイントの DNS 名 (例: myinstance.123456789012.us-east-1.rds.amazonaws.com) です。エンドポイントの値は、Amazon RDS マネジメントコンソールのインスタンスの詳細で確認できます。

4. もう一度ソース MariaDB インスタンスを書き込み可能にします。

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

5. Amazon RDS マネジメントコンソールで、外部の MariaDB データベースをホストするサーバーの IP アドレスを、MariaDB DB インスタンスの VPC セキュリティグループに追加します。VPC セキュリティグループの変更方法の詳細については、Amazon Virtual Private Cloud ユーザーガイドの「VPC のセキュリティグループ」を参照してください。

   以下の条件が満たされると、IP アドレスが変更される場合があります。

   - 外部ソースインスタンスと DB インスタンス間の通信にパブリック IP アドレスを使用している。
   - 外部ソースインスタンスが停止して再起動した。

   これらの条件が満たされている場合は、追加する前に IP アドレスを確認してください。

   外部の MariaDB インスタンスと通信可能にするために、MariaDB DB インスタンスの IP アドレスからの接続を許可するようにローカルネットワークを設定することも必要になる場合があります。MariaDB の DB インスタンスの IP アドレスを確認するには、host コマンドを使用します。

```
host db_instance_endpoint
```

   ホスト名は MariaDB DB インスタンスのエンドポイントの DNS 名です。

6. 選択したクライアントを使用して、外部の MariaDB インスタンスに接続し、レプリケーションに使用される MariaDB ユーザーを作成します。このアカウントはレプリケーション専用に使用

され、セキュリティを強化するためにドメインに制限する必要があります。以下に例を示します。

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

7. 外部の MariaDB インスタンスについて、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。例えば、すべてのデータベースに対する REPLICATION CLIENT および REPLICATION SLAVE 権限を "repl_user" ユーザーに付与するには、以下のコマンドを実行します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

8. MariaDB DB インスタンスをレプリカにします。MariaDB DB インスタンスにマスターユーザーとして接続し、mysql.rds_set_external_master_gtid コマンドを使用して、外部の MariaDB データベースをレプリケーションソースインスタンスとして指定します。ステップ 2 で決定した GTID を使用します。以下に例を示します。

```
CALL mysql.rds_set_external_master_gtid ('mymasterserver.mydomain.com', 3306,
 'repl_user', 'password', 'GTID', 1);
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

9. MariaDB DB インスタンスで、mysql.rds_start_replication コマンドを実行してレプリケーションを開始します。

```
CALL mysql.rds_start_replication;
```

外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

バイナリログファイルのレプリケーションを使用して、RDS for MySQLまたは MariaDB DB インスタンスと Amazon RDS の外部にある MySQL または MariaDB インスタンスとの間でレプリケーションを設定できます。

トピック

- 開始する前に
- 外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

開始する前に

レプリケートされたトランザクションのバイナリログファイルの位置を使用して、レプリケーションを設定できます。

Amazon RDS DB インスタンスでレプリケーションをスタートするために必要なアクセス権限は限定されており、Amazon RDS マスターユーザーは利用できません。そのため、Amazon RDS の mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) または mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)、および mysql.rds_start_replication コマンドを使用して、ライブデータベースと Amazon RDS のデータベースのレプリケーションを設定する必要があります。

MySQL または MariaDB データベースにバイナリログ形式を設定するには、`binlog_format` パラメータを更新します。DB インスタンスがデフォルト DB インスタンスパラメータグループを使用している場合、新しい DB パラメータグループを作成して `binlog_format` パラメータを変更します。MariaDB および MySQL 8.0 以前のバージョンでは、`binlog_format` のデフォルトは `MIXED` です。ただし、特定バイナリログ (binlog) 形式が必要な場合は `binlog_format` を `ROW` または `STATEMENT` に設定する必要もあります。変更を適用するには、DB インスタンスを再起動します。MySQL 8.4 以降のバージョンでは、`binlog_format` のデフォルトは `ROW` です。

`binlog_format` パラメータの設定については、インスタンスデプロイの RDS for MySQL バイナリログの設定 を参照してください。さまざまな MySQL レプリケーションタイプの詳細については、MySQL ドキュメントの「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

Amazon RDS で外部ソースインスタンスとレプリカをセットアップする場合は、次のガイドラインに従ってください。

- レプリカである Amazon RDS DB インスタンスのフェイルオーバーのイベントをモニタリングします。フェイルオーバーが発生すると、レプリカである DB インスタンスが、新しいホスト上に別のネットワークアドレスで再作成されます。フェイルオーバーイベントをモニタリングする方法については、「Amazon RDS イベント通知の操作」を参照してください。
- ソースインスタンスのバイナリログがレプリカに適用されたことを確認するまで、これらのバイナリログを保持します。このメンテナンスによって、障害発生時にソースインスタンスを復元できます。
- Amazon RDS DB インスタンスにある自動バックアップを有効にします。自動バックアップを有効にすると、ソースインスタンスとレプリカを再同期する必要がある場合に、特定の時点にレプリカを復元できます。バックアップと特定の時点への復元の詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

外部ソースインスタンスを使用してバイナリログファイルのレプリケーションを設定するには

1. ソース MySQL または MariaDB インスタンスを読み取り専用にします。

   ```
   mysql> FLUSH TABLES WITH READ LOCK;
   mysql> SET GLOBAL read_only = ON;
   ```

2. ソース MySQL または MariaDB インスタンスで SHOW MASTER STATUS コマンドを実行して、binlog の場所を特定します。

   次の例のような出力を受け取ります。

   ```
   File                          Position
   ------------------------------------
    mysql-bin-changelog.000031      107
   ------------------------------------
   ```

3. mysqldump を使用して、外部のインスタンスから Amazon RDS DB インスタンスにデータベースをコピーします。非常に大きなデータベースでは、「ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする」の手順を使用することが必要になる場合があります。

Linux、macOS、Unix の場合:

```
mysqldump --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
        -pRDS_password
```

Windows の場合:

```
mysqldump --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p オプションと入力するパスワードの間にスペースがないことを確認します。

Amazon RDS DB インスタンスに接続するためのホスト名、ユーザー名、ポート、およびパスワードを指定するには、`--host` コマンドで `--user` (`-u`)、`--port`、`-p` および `mysql` オプションを使用します。ホスト名は、Amazon RDS DB インスタンスのエンドポイントのドメインネームサービス (DNS) 名 (例: `myinstance.123456789012.us-east-1.rds.amazonaws.com`) です。エンドポイントの値は、AWS マネジメントコンソールのインスタンスの詳細で確認できます。

4. もう一度ソース MySQL または MariaDB インスタンスを書き込み可能にします。

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

レプリケーションで使用するバックアップの作成の詳細については、MySQL ドキュメントを参照してください。

5. AWS マネジメントコンソール で、外部のデータベースをホストするサーバーの IP アドレスを、Amazon RDS DB インスタンスの仮想プライベートクラウド (VPC) のセキュリティグループに追加します。VPC セキュリティグループの変更方法の詳細については、Amazon Virtual Private Cloudユーザーガイドの「VPC のセキュリティグループ」を参照してください。

   以下の条件が満たされると、IP アドレスが変更される場合があります。

   - 外部ソースインスタンスと DB インスタンス間の通信にパブリック IP アドレスを使用している。
   - 外部ソースインスタンスが停止して再起動した。

   これらの条件が満たされている場合は、追加する前に IP アドレスを確認してください。

   Amazon RDS の DB インスタンスの IP アドレスからの接続を許可するようにローカルネットワークを設定することも必要になる場合があります。これは、ローカルネットワークから外部の MySQL または MariaDB インスタンスと通信できるようにするためです。Amazon RDS DB インスタンスの IP アドレスを確認するには、`host` コマンドを使用します。

   ```
   host db_instance_endpoint
   ```

   ホスト名は Amazon RDS DB インスタンスのエンドポイントの DNS 名です。

6. 選択したクライアントを使用して、外部のインスタンスに接続し、レプリケーションに使用される ユーザーを作成します。このアカウントをレプリケーション専用に使用し、セキュリティを強化するためドメインに制限します。以下に例を示します。

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

7. 外部の インスタンスについて、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。例えば、すべてのデータベースに対する REPLICATION CLIENT および REPLICATION SLAVE 権限を "repl_user" ユーザーに付与するには、以下のコマンドを実行します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

8. Amazon RDS DB インスタンスをレプリカにします。これを行うには、まず、マスターユーザーとして Amazon RDS の DB インスタンスに接続します。次に、mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降) または mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) のコマンドを使用して、外部の MySQL または MariaDB データベースをソースインスタンスとして指定します。ステップ 2 で特定したマスターログファイル名とマスターログの場所を使用します。次のコマンドはその例です。

   MySQL 8.4

```
CALL mysql.rds_set_external_source ('mysourceserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

   MariaDB と MySQL 8.0 および 5.7

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
```

> **Note**
>
> RDS for MySQL では、代わりに mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降) または mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) のストアドプロシージャを実行することで、遅延レプリケーションを使用するよう選択できます。RDS for MySQL で遅延レプリケーションを使用する 1 つの理由

は、mysql.rds_start_replication_until ストアドプロシージャで災害対策を有効にするためです。現在、RDS for MariaDB では遅延レプリケーションはサポートされていますが、mysql.rds_start_replication_until プロシージャはサポートされていません。

9. Amazon RDS DB インスタンスで、mysql.rds_start_replication コマンドを実行してレプリケーションをスタートします。

```
CALL mysql.rds_start_replication;
```

MariaDB データベースエンジンのオプション

ここでは、MariaDB DB エンジンを実行する Amazon RDS インスタンスで使用できるオプションまたは追加機能について説明します。これらのオプションをオンにするには、カスタムオプショングループにオプションを追加して、そのオプショングループを DB インスタンスに関連付けます。オプショングループの操作方法の詳細については、「オプショングループを使用する」を参照してください。

Amazon RDS では、以下の MariaDB 用オプションがサポートされています。

オプション ID	エンジンバージョン
MARIADB_AUDIT_PLUGIN	MariaDB 10.3 以降

MariaDB 監査プラグインのサポート

Amazon RDS では、MariaDB データベースインスタンスでの MariaDB 監査プラグインの使用をサポートしています。MariaDB 監査プラグインは、データベースへのユーザーのログオンやデータベースに対して実行されたクエリなどのデータベースアクティビティを記録します。データベースのアクティビティのレコードはログファイルに保存されます。

監査プラグインのオプション設定

Amazon RDS では、MariaDB 監査プラグインのオプションの次の設定がサポートされています。

> **Note**
>
> RDS コンソールでオプション設定を構成しない場合、RDS はデフォルト設定を使用します。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	ログファイルの場所。ログファイルには、SERVER_AUDIT_EVENTS で指定されたアクティビティのレコードが含まれます。詳細に

オプション設定	有効な値	デフォルト値	説明
			ついては、「データベースログファイルの表示とリスト化」および「MariaDB データベースのログファイル」を参照してください。
SERVER_AUDIT_FILE_ROTATE_SIZE	1-1000000000	1000000	このバイト数のサイズに達するとファイルがローテーションします。詳細については、「MariaDB のログのローテーションと保持」を参照してください。
SERVER_AUDIT_FILE_ROTATIONS	0-100	9	server_audit_output_type=file 時に保存するログローテーション数。0 に設定すると、ログファイルはローテーションされません。詳細については、MariaDB のログのローテーションと保持およびデータベースログファイルのダウンロードを参照してください。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_EVENTS	CONNECT, QUERY, TABLE, QUERY_DDL , QUERY_DML , QUERY_DML_NO_SELECT , QUERY_DCL	CONNECT, QUERY	ログに記録するアクティビティのタイプ。MariaDB 監査プラグインのインストール自体も記録されます。 • CONNECT: データベースへ接続の成功と失敗、およびデータベースからの切断を記録します。 • QUERY: データベースに対して実行されたすべてのクエリのテキストを記録します。 • TABLE: データベースに対してクエリが実行された際に影響を受けたテーブルを記録します。 • QUERY_DDL : QUERY イベントと同様ですが、返るのは、データ定義言語 (DDL) クエリ (CREATE、ALTER など) のみです。 • QUERY_DML : QUERY イベントと同様ですが、返るのは、データ操作言語 (DML) クエリ (INSERT、UPDATE、SELECT など) のみです。 • QUERY_DML_NO_SELECT : QUERY_DML イベントと類似していますが、SELECT クエリを口グ記録しません。 • QUERY_DCL : QUERY イベントと同様ですが、返るのは、データ制御言語 (DCL) クエリ (GRANT、REVOKE など) のみです。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_INCL_USERS	複数のカンマ区切り値	なし	指定されたユーザーからのアクティビティのみを含めます。デフォルトでは、アクティビティはすべてのユーザーについて記録されます。SERVER_AUDIT_INCL_USERS と SERVER_AUDIT_EXCL_USERS は相互に排他的です。SERVER_AUDIT_INCL_USERS に値を追加する場合は、SERVER_AUDIT_EXCL_USERS に追加される値がないことを確認してください。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_EXCL_USERS	複数のカンマ区切り値	なし	指定されたユーザーからのアクティビティを除外します。デフォルトでは、アクティビティはすべてのユーザーについて記録されます。SERVER_AUDIT_INCL_USERS と SERVER_AUDIT_EXCL_USERS は相互に排他的です。SERVER_AUDIT_EXCL_USERS に値を追加する場合は、SERVER_AUDIT_INCL_USERS に追加される値がないことを確認してください。 rdsadmin ユーザーは 1 秒ごとにデータベースをクエリしてデータベースのヘルスチェックを行います。そのほかの設定によっては、このアクティビティによってログファイルのサイズが急激に増大する可能性があります。このアクティビティを記録する必要がない場合は、SERVER_AUDIT_EXCL_USERS リストにrdsadminユーザーを追加します。 **Note** CONNECT アクティビティは、ユーザーがこのオプション設定で指定されていても、すべてのユーザーについて常に記録されます。
SERVER_AUDIT_LOGGING	ON	ON	ログ記録がアクティブです。唯一の有効な値は ON です。Amazon RDS では、ログ記録の非アクティブ化はサポートしていません。ログ記録を非アクティブ化する場合は、MariaDB 監査プラグインを削除します。詳細については、「MariaDB 監査プラグインの削除」を参照してください。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_QUERY_LOG_LIMIT	0-2147483647	1024	レコードのクエリ文字列の長さに対する制限。

MariaDB 監査プラグインの追加

MariaDB 監査プラグインを DB インスタンスに追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

MariaDB 監査プラグインを追加した後で、DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、直ちに監査が開始されます。

MariaDB 監査プラグインを追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。それ以外の場合は、カスタム DB オプショングループを作成します。[Engine] (エンジン) で [mariadb] を選択し、[Major engine version] (メジャーエンジンバージョン) で [10.3] またはそれ以降を選択します。詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [MARIADB_AUDIT_PLUGIN] オプションを追加し、オプションを設定します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。各設定の詳細については、「監査プラグインのオプション設定」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。
   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

- 既存の DB インスタンスの場合は、DB インスタンスを修正し、新しいオプショングループをアタッチすることで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

MariaDB 監査プラグインのログの表示とダウンロード

MariaDB 監査プラグインを有効にした後は、他のテキストベースのログファイルと同様の方法でログファイル内の結果にアクセスします。監査ログファイルは `/rdsdbdata/log/audit/` にあります。コンソールでログファイルを表示する方法の詳細については、「データベースログファイルの表示とリスト化」を参照してください。ログファイルのダウンロードについては、「データベースログファイルのダウンロード」を参照してください。

MariaDB 監査プラグインの設定の変更

MariaDB 監査プラグインを有効にした後、プラグインの設定を変更できます。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。各設定の詳細については、「監査プラグインのオプション設定」を参照してください。

MariaDB 監査プラグインの削除

Amazon RDS では、MariaDB 監査プラグインのログ記録の無効化はサポートされていません。ただし、DB インスタンスからプラグインを削除することはできます。MariaDB 監査プラグインを削除すると、DB インスタンスが自動的に再起動され、監査が停止します。

MariaDB 監査プラグインを DB インスタンスから削除するには、次のいずれかを実行します。

- MariaDB 監査プラグインが所属するオプショングループからプラグインを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、プラグインが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

MariaDB のパラメータ

デフォルトでは、MariaDB DB インスタンスは MariaDB データベースに固有の DB パラメータグループを使用します。このパラメータグループには、MySQL データベースエンジンの Amazon RDS DB パラメータグループに含まれるいくつかのパラメータがありますが、すべてのパラメータがあるわけではありません。また、いくつかの新しい MariaDB 固有のパラメータも含まれます。パラメータグループの操作とパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

MariaDB パラメータの表示

RDS for MariaDB パラメータは、選択したストレージエンジンのデフォルト値に設定されます。MariaDB パラメータの詳細については、MariaDB のドキュメントを参照してください。MariaDB ストレージエンジンの詳細については、「Amazon RDS の MariaDB でサポートされているストレージエンジン」を参照してください。

特定の RDS for MariaDB バージョンで使用できるパラメータは、RDS コンソールまたは AWS CLI で確認できます。RDS コンソールでの MariaDB パラメータグループ内のパラメータの表示方法については、「Amazon RDS のDB パラメータグループのパラメータ値の表示」を参照してください。

AWS CLI を使用して `describe-engine-default-parameters` コマンドを実行すると、RDS for MariaDB バージョンのパラメータを表示できます。`--db-parameter-group-family` オプションには、次の値のうち 1 つを指定します。

- `mariadb12.3`
- `mariadb11.8`
- `mariadb11.4`
- `mariadb10.11`
- `mariadb10.6`
- `mariadb10.5`
- `mariadb10.4`
- `mariadb10.3`

例えば、RDS for MariaDB バージョン 10.6 のパラメータを表示するには、次のコマンドを実行します。

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6
```

出力は次のようになります。

```
{
    "EngineDefaults": {
        "Parameters": [
            {
                "ParameterName": "alter_algorithm",
                "Description": "Specify the alter table algorithm.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "string",
                "AllowedValues": "DEFAULT,COPY,INPLACE,NOCOPY,INSTANT",
                "IsModifiable": true
            },
            {
                "ParameterName": "analyze_sample_percentage",
                "Description": "Percentage of rows from the table ANALYZE TABLE will
 sample to collect table statistics.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "float",
                "AllowedValues": "0-100",
                "IsModifiable": true
            },
            {
                "ParameterName": "aria_block_size",
                "Description": "Block size to be used for Aria index pages.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "integer",
                "AllowedValues": "1024-32768",
                "IsModifiable": false
            },
            {
                "ParameterName": "aria_checkpoint_interval",
                "Description": "Interval in seconds between automatic checkpoints.",
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "0-4294967295",
                "IsModifiable": true
```

```
            },
        ...
```

RDS for MariaDB バージョン 10.6 の変更可能なパラメータのみを一覧表示するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6 \
   --query 'EngineDefaults.Parameters[?IsModifiable==`true`]'
```

Windows の場合:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mariadb10.6 ^
   --query "EngineDefaults.Parameters[?IsModifiable==`true`]"
```

使用できない MySQL パラメータ

以下の MySQL パラメータは、MariaDB 固有の DB パラメータグループでは使用できません。

- bind_address
- binlog_error_action
- binlog_gtid_simple_recovery
- binlog_max_flush_queue_time
- binlog_order_commits
- binlog_row_image
- binlog_rows_query_log_events
- binlogging_impossible_mode
- block_encryption_mode
- core_file
- default_tmp_storage_engine
- div_precision_increment
- end_markers_in_json
- enforce_gtid_consistency
- eq_range_index_dive_limit

- explicit_defaults_for_timestamp
- gtid_executed
- gtid-mode
- gtid_next
- gtid_owned
- gtid_purged
- log_bin_basename
- log_bin_index
- log_bin_use_v1_row_events
- log_slow_admin_statements
- log_slow_slave_statements
- log_throttle_queries_not_using_indexes
- master-info-repository
- optimizer_trace
- optimizer_trace_features
- optimizer_trace_limit
- optimizer_trace_max_mem_size
- optimizer_trace_offset
- relay_log_info_repository
- rpl_stop_slave_timeout
- slave_parallel_workers
- slave_pending_jobs_size_max
- slave_rows_search_algorithms
- storage_engine
- table_open_cache_instances
- timed_mutexes
- transaction_allow_batching
- validate-password
- validate_password_dictionary_file
- validate_password_length

- validate_password_mixed_case_count
- validate_password_number_count
- validate_password_policy
- validate_password_special_char_count

MySQL パラメータの詳細については、MySQL のドキュメントを参照してください。

MySQL DB スナップショットから MariaDB DB インスタンスへのデータ移行

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、RDS for MySQL DB スナップショットを MariaDB を実行している新しい DB インスタンスに移行できます。MySQL 5.6 か 5.7 を実行している Amazon RDS DB インスタンスから作成された DB スナップショットを使用する必要があります。RDS for MySQL DB スナップショットを作成する方法については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

スナップショットの移行は、スナップショットを取得した元の DB インスタンスには影響しません。元の DB インスタンスの代わりとしてトラフィックを転送する前に、新しい DB インスタンスをテストして検証できます。

MySQL から MariaDB に移行した後、MariaDB DB インスタンスは、デフォルトの DB パラメータグループやオプショングループに関連付けられます。DB スナップショットを復元した後、新しい DB インスタンスにカスタム DB パラメータグループを関連付けることができます。ただし、MariaDB パラメータグループには異なる設定可能なシステム可変があります。MySQL と MariaDB のシステム可変の違いについては、「MariaDB と MySQL のシステム可変の違い」を参照してください。DB パラメータグループの詳細については、Amazon RDS のパラメータグループ を参照してください。オプションのグループの詳細については、オプショングループを使用する を参照してください。

移行の実行

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、RDS for MySQL DBスナップショットを新しいMariaDB DBインスタンスに移行できます。

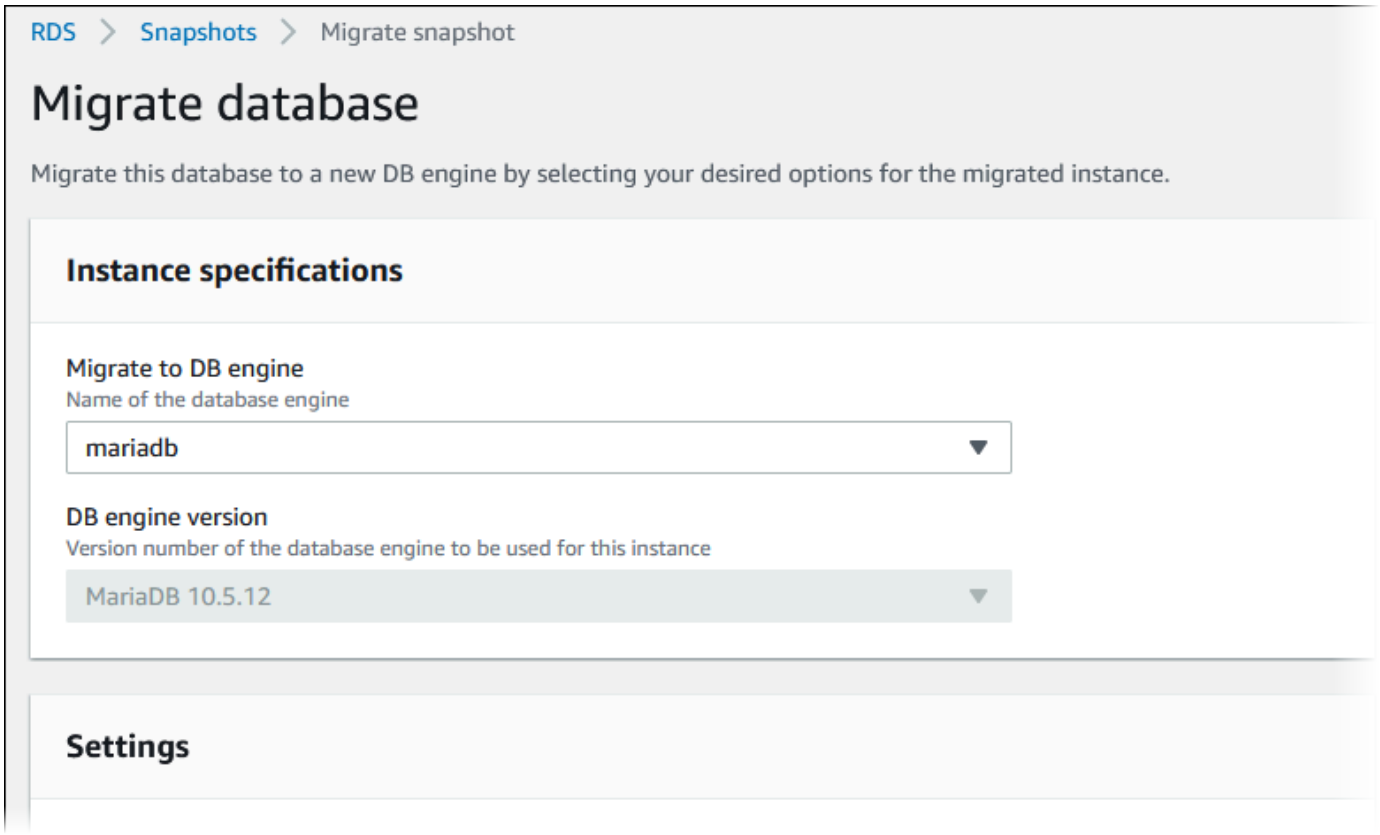
コンソール

MySQL DB スナップショットを MariaDB DB インスタンスに移行するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択し、移行する MySQL DB スナップショットを選択します。
3. 「アクション」で「スナップショットの移行」 を選択します。「Migrate Database」 ページが表示されます。

4. [Migrate to DB Engine] で、[mariadb] を選択します。

   Amazon RDS は DB エンジンバージョンを自動的に選択します。DB エンジンバージョンを変更することはできません。



5. 残りのセクションで、DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。
6. 移行 を選択します。

AWS CLI

MySQL DB スナップショットから MariaDB DB インスタンスにデータを移行するには、以下のオプションを指定して AWS CLI の `restore-db-instance-from-db-snapshot` コマンドを実行します。

- --DB インスタンス識別子 - DB スナップショットから作成される DB インスタンスの名前。
- --db-snapshot-identifier - リストア元の DB スナップショットの識別子。
- --engine - 新しいインスタンスに使用するデータベースエンジン。

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier newmariadbinstance \
    --db-snapshot-identifier mysqlsnapshot \
    --engine mariadb
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier newmariadbinstance ^
    --db-snapshot-identifier mysqlsnapshot ^
    --engine mariadb
```

API

MySQL DB スナップショットから MariaDB DB インスタンスにデータを移行するには、Amazon RDS API オペレーション RestoreDBInstanceFromDBSnapshot を呼び出します。

MariaDB と MySQL の間の非互換性

MySQL と MariaDB の間の非互換性は以下のとおりです。

- MySQL 8.0 で作成した DB スナップショットは MariaDB に移行できません。
- ソース MySQL データベースで SHA256 パスワードハッシュを使用している場合、MariaDB データベースに接続する前に SHA256 でハッシュ化されたユーザーパスワードをリセットしてください。次のコードは、SHA256 でハッシュ化されたパスワードをリセットする方法を示しています。

  ```
  SET old_passwords = 0;
  UPDATE mysql.user SET plugin = 'mysql_native_password',
  Password = PASSWORD('new_password')
  WHERE (User, Host) = ('master_user_name', %);
  FLUSH PRIVILEGES;
  ```

- RDS マスターユーザーアカウントが SHA-256 パスワードハッシュを使用している場合、AWS マネジメントコンソール、modify-db-instance AWS CLI コマンド、または ModifyDBInstance

RDS API オペレーションを使用してパスワードをリセットしてください。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

- MariaDB では、Memcached プラグインはサポートされていません。ただし、Memcached プラグインで使用されるデータは InnoDB テーブルとして保存されます。MySQL DB スナップショットを移行した後、SQL を使用して Memcached プラグインで使用されるデータにアクセスできます。innodb_memcache データベースに関する詳細については、「InnoDB memcached Plugin Internals」を参照してください。

Amazon RDS SQL での MariaDB リファレンス

ここでは、MariaDB DB エンジンを実行する Amazon RDS インスタンスで使用できるシステムストアドプロシージャについて説明します。

MariaDB DB インスタンスと MySQL DB インスタンスに対して利用できるシステムストアドプロシージャを使用できます。これらのストアドプロシージャは、「RDS for MySQL ストアドプロシージャリファレンス」に記載されています。MariaDB DB インスタンスは、`mysql.rds_start_replication_until` と `mysql.rds_start_replication_until_gtid` を除くすべてのストアドプロシージャをサポートします。

さらに、次のシステムストアドプロシージャが、MariaDB を実行する Amazon RDS DB インスタンスでのみサポートされています。

- mysql.rds_replica_status
- mysql.rds_set_external_master_gtid
- mysql.rds_kill_query_id
- mysql.rds_execute_operation

mysql.rds_replica_status

MariaDB リードレプリカのレプリケーションステータスを表示します。

リードレプリカでこの手順を呼び出して、レプリカスレッドの基本パラメータに関するステータス情報を表示します。

構文

```
CALL mysql.rds_replica_status;
```

使用に関する注意事項

この手順は、MariaDB バージョン 10.5 以降を実行している MariaDB DB インスタンスのみでサポートされます。

この手順は、`SHOW REPLICA STATUS` コマンドと同等です。このコマンドは、MariaDB バージョン 10.5 以降の DB インスタンスではサポートされません。

MariaDB の以前のバージョンでは、同等の SHOW SLAVE STATUS コマンドには REPLICATION SLAVE 特権が必要でした。MariaDB バージョン 10.5 以上では、REPLICATION REPLICA ADMIN 権限が必要です。MariaDB 10.5 以上の DB インスタンスの RDS 管理を保護するために、この新しい権限は RDS マスターユーザーに付与されません。

例

次の例は、MariaDB リードレプリカのステータスを示しています。

```
call mysql.rds_replica_status;
```

応答は次の例のようになります。

```
*************************** 1. row ***************************
               Replica_IO_State: Waiting for master to send event
                    Source_Host: XX.XX.XX.XXX
                    Source_User: rdsrepladmin
                    Source_Port: 3306
                  Connect_Retry: 60
                Source_Log_File: mysql-bin-changelog.003988
            Read_Source_Log_Pos: 405
                 Relay_Log_File: relaylog.011024
                  Relay_Log_Pos: 657
          Relay_Source_Log_File: mysql-bin-changelog.003988
             Replica_IO_Running: Yes
            Replica_SQL_Running: Yes
                Replicate_Do_DB:
            Replicate_Ignore_DB:
             Replicate_Do_Table:
         Replicate_Ignore_Table:
mysql.rds_sysinfo,mysql.rds_history,mysql.rds_replication_status
        Replicate_Wild_Do_Table:
    Replicate_Wild_Ignore_Table:
                     Last_Errno: 0
                     Last_Error:
                   Skip_Counter: 0
            Exec_Source_Log_Pos: 405
                Relay_Log_Space: 1016
                Until_Condition: None
                 Until_Log_File:
                  Until_Log_Pos: 0
             Source_SSL_Allowed: No
```

```
              Source_SSL_CA_File:
              Source_SSL_CA_Path:
                 Source_SSL_Cert:
               Source_SSL_Cipher:
                  Source_SSL_Key:
           Seconds_Behind_Master: 0
   Source_SSL_Verify_Server_Cert: No
                   Last_IO_Errno: 0
                   Last_IO_Error:
                  Last_SQL_Errno: 0
                  Last_SQL_Error:
     Replicate_Ignore_Server_Ids:
                Source_Server_Id: 807509301
                  Source_SSL_Crl:
              Source_SSL_Crlpath:
                      Using_Gtid: Slave_Pos
                     Gtid_IO_Pos: 0-807509301-3980
         Replicate_Do_Domain_Ids:
     Replicate_Ignore_Domain_Ids:
                   Parallel_Mode: optimistic
                       SQL_Delay: 0
             SQL_Remaining_Delay: NULL
       Replica_SQL_Running_State: Reading event from the relay log
              Replica_DDL_Groups: 15
Replica_Non_Transactional_Groups: 0
    Replica_Transactional_Groups: 3658
1 row in set (0.000 sec)

Query OK, 0 rows affected (0.000 sec)
```

mysql.rds_set_external_master_gtid

Amazon RDS 外部で動作する MariaDB インスタンスから MariaDB DB インスタンスに対して、GTID ベースのレプリケーションを設定します。このストアドプロシージャは、外部の MariaDB インスタンスのバージョンが 10.0.24 以降である場合のみサポートされます。1 つまたは両方のインスタンスが MariaDB のグローバルトランザクション識別子 (GTID) をサポートしていない場合にレプリケーションを設定する際は、「mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)」を使用します。

レプリケーションで GTID を使用すると、クラッシュのリスクがない安全性機能が実現されます。これはバイナリログのレプリケーションでは提供されません。したがって、レプリケーションのインスタンスがサポートしている場合は、GTID の使用を推奨します。

構文

```
CALL mysql.rds_set_external_master_gtid(
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , gtid
  , ssl_encryption
);
```

パラメータ

host_name

文字列。Amazon RDS の外部で動作する MariaDB インスタンス (ソースインスタンス) のホスト名または IP アドレス。

host_port

整数。Amazon RDS の外部で動作する MariaDB インスタンス (ソースインスタンス) で使用されるポート。ポート番号を変換する SSH ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって公開されるポート番号を指定します。

replication_user_name

文字列。リードレプリカとして設定される MariaDB DB インスタンスでの `REPLICATION SLAVE` アクセス許可を持つユーザーの ID。

replication_user_password

文字列。`replication_user_name` で指定されたユーザー ID のパスワード。

gtid

文字列。レプリケーションが開始されるソースインスタンスのグローバルトランザクション ID。

レプリケーションの設定中にソースインスタンスがロックされている場合は、`@@gtid_current_pos` を使用して現在の GTID を入手できます。したがって、GTID の取得時とレプリケーションの開始時点の間では、バイナリログは変更されません。

それ以外の場合、`mysqldump` のバージョン 10.0.13 以降を使用しているか、レプリケーションを開始する前にレプリカインスタンスを入力している場合は、`--master-data` オプションまたは `--dump-slave` オプションを使用して、出力の GTID の場所を取得できます。`mysqldump` の

バージョン 10.0.13 以降を使用していない場合は、SHOW MASTER STATUS を実行するか、または同じ mysqldump オプションを使用してバイナリログファイル名と場所を取得できます。次に外部 MariaDB インスタンスで BINLOG_GTID_POS を実行して、それを GTID に変換します。

```
SELECT BINLOG_GTID_POS('<binary log file name>', <binary log file position>);
```

GTID の MariaDB 実装の詳細については、MariaDB ドキュメントの「グローバルトランザクション ID」を参照してください。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

使用に関する注意事項

mysql.rds_set_external_master_gtid プロシージャは、マスターユーザーが実行してください。Amazon RDS の外部で動作する MariaDB インスタンスのレプリカとして設定される、MariaDB DB インスタンスで実行する必要があります。mysql.rds_set_external_master_gtid を実行する前に、Amazon RDS の外部で動作する MariaDB インスタンスをソースインスタンスとして必ず設定してください。詳細については、「Amazon RDS for MariaDB DB インスタンスへのデータのインポート」を参照してください。

> **Warning**
>
> 2 つの Amazon RDS DB インスタンス間でレプリケーションを管理するために mysql.rds_set_external_master_gtid を使用しないでください。このストアドプロシージャは、RDS の外部で動作する 1 つの MariaDB インスタンスでレプリケーションする場合にのみ使用します。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_master_gtid を呼び出し、Amazon RDS DB インスタンスをリードレプリカとして設定した後で、レプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスを開始できます。mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_master_gtid が呼び出されると、Amazon RDS では、時刻、ユーザー、"マスターの設定" アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

例

MariaDB DB インスタンスで動作させる場合は、次の例にあるように、Amazon RDS の外部で動作する MariaDB のインスタンスのレプリカとして設定します。

```
call mysql.rds_set_external_master_gtid
 ('Sourcedb.some.com',3306,'ReplicationUser','SomePassW0rd','0-123-456',0);
```

mysql.rds_kill_query_id

MariaDB サーバーに対して実行されているクエリを終了して、長時間実行されているクエリや問題のあるクエリを終了します。クエリ ID を識別し、特定のクエリを効果的に停止して、パフォーマンスの問題に対処し、最適なデータベースオペレーションを維持できます。

構文

```
CALL mysql.rds_kill_query_id(queryID);
```

パラメータ

queryID

整数。終了するクエリの識別子。

使用に関する注意事項

MariaDB サーバーに対して実行中のクエリを停止するには、mysql.rds_kill_query_id プロシージャを使用して、そのクエリの ID を渡します。クエリ ID を取得するには、次に示すように、MariaDB の 情報スキーマ PROCESSLIST テーブルを参照します。

```
SELECT USER, HOST, COMMAND, TIME, STATE, INFO, QUERY_ID FROM
                INFORMATION_SCHEMA.PROCESSLIST WHERE USER = '<user name>';
```

MariaDB サーバーへの接続は保持されます。

例

次の例では、230040 のクエリ ID を持つクエリを終了します。

```
call mysql.rds_kill_query_id(230040);
```

mysql.rds_execute_operation

InnoDB オペレーションを実行して、バッファプールの状態と一時テーブルスペースを管理します。この手順を使用して、バッファプールの状態のダンプとロード、一時テーブルスペースの切り捨てなどの InnoDB オペレーションを動的に制御できます。

構文

```
CALL mysql.rds_execute_operation(operation);
```

パラメータ

オペレーション

文字列。実行する InnoDB オペレーション。次の値を指定できます。

- innodb_buffer_pool_dump_now – バッファプールの現在の状態をダンプするオペレーション。
- innodb_buffer_pool_load_now – 保存されたバッファプールの状態をロードするオペレーション。
- innodb_buffer_pool_load_abort – バッファプールのロードオペレーションを中止するオペレーション。
- innodb_truncate_temporary_tablespace_now – 一時テーブルスペースを切り捨てるオペレーション。

使用に関する注意事項

この手順は、MariaDB バージョン 11.8 以降を実行している MariaDB DB インスタンスのみでサポートされます。

実行中、バイナリログ記録は、これらの管理コマンドのレプリケーションを防ぐために一時的に無効になります。

この手順では、mysql.rds_history テーブル内のすべてのオペレーションをログに記録することで監査証跡を維持します。

例

次の例は、mysql.rds_execute_operation を使用した一時的なテーブルスペースの縮小を示しています。

現在の一時テーブルスペースのサイズを確認するには、次のクエリを実行します。

```
SELECT FILE_SIZE FROM information_schema.innodb_sys_tablespaces WHERE name LIKE
 'innodb_temporary';
+------------+
| FILE_SIZE  |
+------------+
| 6723469312 |  -- 6.3 GB
+------------+
```

一時テーブルを削除しても、グローバルテーブルスペースのストレージ使用量は減りません。グローバルテーブルスペースのサイズを減らすには、mysql.rds_execute_operation コマンドを実行して一時テーブルスペースを縮小します。

```
CALL mysql.rds_execute_operation('innodb_truncate_temporary_tablespace_now');
Query OK, 2 rows affected (0.004 sec)
```

手順を実行したら、スペースが再利用されたことを確認します。

```
SELECT FILE_SIZE FROM information_schema.innodb_sys_tablespaces WHERE name LIKE
 'innodb_temporary';
+-----------+
| FILE_SIZE |
+-----------+
|  12582912 |  -- 12 MB
+-----------+
```

> **Note**
>
> 一時テーブルスペースのサイズと現在のワークロードによっては、縮小オペレーションに時間がかかる場合があります。

> **Important**
>
> 一時テーブルスペースは、そのサイズに寄与したすべての一時テーブルが使用されなくなった場合にのみ縮小されます。インスタンスにアクティブな一時テーブルスペースがない場合は、この手順を実行することをお勧めします。

MariaDB DB インスタンスのローカルタイムゾーン

デフォルトでは、MariaDB DB インスタンスのタイムゾーンは協定世界時 (UTC) です。代わりに、DB インスタンスのタイムゾーンをアプリケーションのローカルタイムゾーンに設定できます。

DB インスタンスのローカルタイムゾーンを設定するには、DB インスタンスのパラメータグループの `time_zone` パラメータを、このセクションで後述するサポートされている値のいずれかに設定します。パラメータグループの `time_zone` パラメータを設定すると、そのパラメータグループを使用しているすべての DB インスタンスとリードレプリカは、新しいローカルタイムゾーンを使用するように変更されます。パラメータグループのパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

ローカルタイムゾーンを設定した後、データベースへのすべての新しい接続にその変更が反映されます。ローカルタイムゾーンを変更するときにデータベースへの接続を開いている場合、その接続を閉じて新しい接続を開くまで、ローカルタイムゾーンは更新されません。

DB インスタンスとそのリードレプリカには異なるローカルタイムゾーンを設定できます。そのためには、DB インスタンスとレプリカに異なるパラメータグループを使用し、各パラメータグループの `time_zone` パラメータを異なるローカルタイムゾーンに設定します。

AWS リージョン 間のレプリケーションを実行する場合は、ソース DB インスタンスとリードレプリカに異なるパラメータグループ (パラメータグループは AWS リージョン に固有のもの) を使用します。各インスタンスに同じローカルタイムゾーンを使用するには、インスタンスとリードレプリカのパラメータグループの `time_zone` パラメータを設定する必要があります。

DB スナップショットから DB インスタンスを復元すると、ローカルタイムゾーンが UTC に設定されます。復元が完了したら、タイムゾーンをローカルタイムゾーンに更新できます。DB インスタンスをある時点まで復元する場合、復元された DB インスタンスのローカルタイムゾーンは、復元された DB インスタンスのパラメータグループに設定されているタイムゾーンです。

Internet Assigned Numbers Authority (IANA) は年に数回、https://www.iana.org/time-zones で新しいタイムゾーンを公開します。RDS が MariaDB の新しいマイナーメンテナンスリリースをリリースするたびに、リリース時の最新のタイムゾーンデータが同梱されます。MariaDB の最新バージョンの RDS を使用すると、RDS からの最新のタイムゾーンデータが得られます。DB インスタンスに最新のタイムゾーンデータがあることを確認するには、DB エンジンの上位バージョンにアップグレードすることをお勧めします。または、MariaDB DB インスタンスのタイムゾーンテーブルを手動で変更することもできます。そのためには、SQL コマンドを使用するか、SQL クライアントで mysql_tzinfo_to_sql ツールを実行します。タイムゾーンデータを手動で更新して、DB インスタンスを再起動し、変更を有効にします。RDS は、実行中の DB インスタンスのタイムゾーンデータを変

更またはリセットしません。新しいタイムゾーンデータは、データベースエンジンのバージョンアップグレードを実行する場合にのみインストールされます。

ローカルタイムゾーンは以下のいずれかの値に設定できます。

ゾーン	タイムゾーン
アフリカ	Africa/Cairo、Africa/Casablanca、Africa/Harare、Africa/Monrovia、Africa/Nairobi、Africa/Tripoli、Africa/Windhoek
南北アメリカ大陸	America/Araguaina、America/Asuncion、America/Bogota、America/Buenos_Aires、America/Caracas、America/Chihuahua、America/Cuiaba、America/Denver、America/Fortaleza、America/Guatemala、America/Halifax、America/Manaus、America/Matamoros、America/Monterrey、America/Montevideo、America/Phoenix、America/Santiago、America/Tijuana
アジア	Asia/Amman、Asia/Ashgabat、Asia/Baghdad、Asia/Baku、Asia/Bangkok、Asia/Beirut、Asia/Calcutta、Asia/Damascus、Asia/Dhaka、Asia/Irkutsk、Asia/Jerusalem、Asia/Kabul、Asia/Karachi、Asia/Kathmandu、Asia/Krasnoyarsk、Asia/Magadan、Asia/Muscat、Asia/Novosibirsk、Asia/Riyadh、Asia/Seoul、Asia/Shanghai、Asia/Singapore、Asia/Taipei、Asia/Tehran、Asia/Tokyo、Asia/Ulaanbaatar、Asia/Vladivostok、Asia/Yakutsk、Asia/Yerevan
大西洋	Atlantic/Azores
オーストラリア	Australia/Adelaide、Australia/Brisbane、Australia/Darwin、Australia/Hobart、Australia/Perth、Australia/Sydney
ブラジル	Brazil/DeNoronha、Brazil/East
カナダ	Canada/Newfoundland, Canada/Saskatchewan, Canada/Yukon
欧州	Europe/Amsterdam、Europe/Athens、Europe/Dublin、Europe/Helsinki、Europe/Istanbul、Europe/Kaliningrad Europe/Moscow、Europe/Paris、Europe/Prague、Europe/Sarajevo
太平洋	Pacific/Auckland、Pacific/Fiji、Pacific/Guam、Pacific/Honolulu、Pacific/Samoa

ゾーン	タイムゾーン
米国	US/Alaska、US/Central、US/East-Indiana、US/Eastern、US/Pacific
UTC	UTC

Amazon RDS for MySQL の既知の問題と制限

RDS for MariaDB を使用する際の既知の問題と制限事項は次のとおりです。

> **Note**
>
> これはすべてを網羅したリストではありません。

トピック

- Amazon RDS での MariaDB のファイルサイズ制限
- InnoDB 予約語
- カスタムポート
- Performance Insights

Amazon RDS での MariaDB のファイルサイズ制限

MariaDB DB インスタンスの場合、InnoDB file-per-table テーブルスペースを使用するとき、テーブルのサイズは最大 16 TB です。また、システムのテーブルスペースも最大 16 TB に制限されています。InnoDB file-per-table テーブルスペース (各テーブルが独自のテーブルスペースに存在) が、デフォルトで MariaDB DB インスタンスに設定されます。この制限は MariaDB DB インスタンスの最大ストレージ制限とは関係ありません。ストレージの制限の詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。

InnoDB file-per-table テーブルスペースの使用は、アプリケーションにより長所と短所があります。アプリケーションに最適な方法を判断するには、MySQL ドキュメントの「File-Per-Table テーブルスペース」を参照してください。

テーブルを最大ファイルサイズまで拡張可能にすることはお勧めしません。一般的に望ましいのは、データを小さなテーブルにパーティショニングすることです。これにより、パフォーマンスと復旧時間が改善される可能性があります。

大きなテーブルを小さなテーブルに分ける方法の 1 つはパーティション化です。パーティション化では、指定したルールに基づいて、大きなテーブルの各部分を個別のファイルに分散します。例えば、トランザクションを日付ごとに保存する場合、パーティション化を使用して古いトランザクションを別々のファイルに分散させるパーティションルールを作成できます。また、定期的に、アプリ

ケーションですぐに使用する必要のない履歴トランザクションデータをアーカイブできます。詳細については、MySQL ドキュメントの「Partitioning」を参照してください。

すべての InnoDB テーブルスペースのサイズを確認するには

- 次の SQL コマンドを使用して、パーティション化の対象になる過度に大きなテーブルがあるか判断します。

> **Note**
>
> MariaDB 10.6 以降では、このクエリは InnoDB システムテーブルスペースのサイズも返します。
> 10.6 より前のバージョンの MariaDB では、システムテーブルをクエリして InnoDB システムテーブルスペースのサイズを判断することはできません。新しいバージョンにアップグレードすることをお勧めします。

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_SYS_TABLESPACES ORDER BY 3 DESC;
```

InnoDB 以外のユーザーテーブルのサイズを確認するには

- 次の SQL コマンドを使用して、InnoDB 以外のユーザーテーブルが過度に大きいかどうかを確認します。

```
SELECT TABLE_SCHEMA, TABLE_NAME, round(((DATA_LENGTH + INDEX_LENGTH+DATA_FREE)
/ 1024 / 1024/ 1024), 2) As "Approximate size (GB)" FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema')
and ENGINE<>'InnoDB';
```

InnoDB file-per-table テーブルスペースを有効にするには

- DB インスタンスのパラメータグループの `innodb_file_per_table` パラメータを `1` に設定します。

InnoDB file-per-table テーブルスペースを無効にするには

- DB インスタンスのパラメータグループの innodb_file_per_table パラメータを 0 に設定します。

パラメータグループの更新については、「Amazon RDS のパラメータグループ」を参照してください。

InnoDB file-per-table テーブルスペースを有効または無効にした場合は、ALTER TABLE コマンドを発行できます。このコマンドを使用して、テーブルをグローバルテーブルスペースからテーブル独自のテーブルスペースに移動できます。または、テーブルを独自のテーブルスペースからグローバルテーブルスペースに移動できます。次に例を示します。

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

InnoDB 予約語

InnoDB は MariaDB の RDS の予約語です。MariaDB データベースにこの名前を使用することはできません。

カスタムポート

Amazon RDS は MariaDB エンジンのカスタムポート 33060 への接続をブロックします。MariaDB エンジン用に別のポートを選択してください。

Performance Insights

InnoDB カウンターは、MariaDB コミュニティではサポートされなくなったため、MariaDB バージョン 10.11 の RDS の Performance Insights には表示されません。

Amazon RDS for Microsoft SQL Server

Amazon RDS は、複数のバージョンやエディションの Microsoft SQL Server をサポートします。以下の表は、メジャーバージョンごとのサポートされている最新のマイナーバージョンを示しています。サポートされているバージョン、エディション、および RDS エンジンのバージョンの詳細なリストについては、「Amazon RDS での Microsoft SQL Server バージョン」を参照してください。

メジャーバージョン	サービスパック/ GDR	累積更新	マイナーバージョン	ナレッジベース記事	リリース日
SQL Server 2025	該当しない	CU6	17.0.4055.5	KB5093421	2026 年 6 月 17 日
SQL Server 2022	該当しない	CU25	16.0.4255.1	KB5081477	2026 年 5 月 20 日
SQL Server 2019	GDR	CU32 GDR	15.0.4470.1	KB5090407	2026 年 5 月 12 日
SQL Server 2017	GDR	CU31 GDR	14.0.3530.2	KB5090354	2026 年 5 月 12 日
SQL Server 2016	SP3 GDR	該当しない	13.0.6490.1	KB5089271	2026 年 5 月 12 日

SQL Server のライセンスについては、「Amazon RDS での Microsoft SQL Server のライセンス」を参照してください。SQL Server のビルドの詳細については、「最新の SQL Server ビルドに関する情報の検索場所」に関するこの Microsoft のサポート記事を参照してください。

Amazon RDS では、DB インスタンス、DB スナップショット、ポイントインタイムリカバリ、自動バックアップ、手動バックアップを作成できます。SQL Server を実行する DB インスタンスは VPC 内で使用できます。SQL Server を実行している DB インスタンスへの接続に Secure Sockets Layer (SSL) を使用することもできます。さらに、保管時のデータの暗号化に Transparent Data Encryption (TDE) を使用することも可能です。Amazon RDS は現在、SQL Server データベースミラーリング (DBM) または Always On Availability グループ (AG) を、高可用性フェイルオーバーソリューションとして使用することで、SQL Server 向けのマルチ AZ デプロイをサポートしています。

マネージド型サービスを提供するために、Amazon RDS では DB インスタンスへのシェルアクセスはできないように設定されています。また、高度な権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されています。Amazon RDS では、Microsoft SQL Server Management Studio などの標準的な SQL クライアントアプリケーションを使用しての、DB インスタンス上のデータベースに対するアクセスがサポートされています。Amazon RDS では、Telnet、Secure Shell (SSH)、または Windows のリモートデスクトップ接続を使用した、DB インスタンスへの直接的なホストアクセスは許可されません。DB インスタンスを作成すると、マスターユーザーに対して、そのインスタンス上のすべてのユーザーデータベースに対する db_owner ロールが割り当てられ、バックアップに使用するアクセス許可を除き、すべてのデータベースレベルのアクセス許可が付与されます。バックアップは、Amazon RDS により自動的に実行されます。

最初の DB インスタンスを作成する前に、このガイドの「セットアップ」セクションの手順を完了してください。詳細については、「Amazon RDS 環境のセットアップ」を参照してください。

トピック

- Amazon RDS の Microsoft SQL Server 用の一般的な管理タスク
- Microsoft SQL Server DB インスタンスの制限
- Microsoft SQL Server の DB インスタンスクラスのサポート
- RDS for SQL Server インスタンスの CPU を最適化する
- Microsoft SQL Server のセキュリティ
- Microsoft SQL Server DB インスタンス用のコンプライアンスプログラムサポート
- Amazon RDS での Microsoft SQL Server バージョン
- Amazon RDS for Microsoft SQL Server のバージョンポリシー
- Amazon RDS での Microsoft SQL Server の機能
- Microsoft SQL Server のデータベースミラーリングまたは Always On 可用性グループを使用したマルチ AZ 配置
- Transparent Data Encryption を使用した保管時のデータの暗号化
- Amazon RDS for Microsoft SQL Server 用の関数とストアドプロシージャ
- Microsoft SQL Server DB インスタンスのローカルタイムゾーン
- Amazon RDS での Microsoft SQL Server のライセンス
- SQL Server DB インスタンスへの接続
- RDS for SQL Server での SQL Server Developer Edition の使用
- RDS for SQL Server の Bring Your Own Media (BYOM)
- RDS for SQL Server による Active Directory の操作

- Microsoft SQL Server DB エンジンのアップグレード
- RDS for SQL Server のストレージの使用
- ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート
- Amazon RDS での Microsoft SQL Server 用のリードレプリカの使用
- Amazon RDS for Microsoft SQL Server のマルチ AZ 配置
- Amazon RDS での Microsoft SQL Server の追加機能
- Microsoft SQL Server データベースエンジンのオプション
- Amazon RDS for Microsoft SQL Server の一般的な DBA タスク

Amazon RDS の Microsoft SQL Server 用の一般的な管理タスク

以下に示しているのは、Amazon RDS for SQL Server DB インスタンスで実行する一般的な管理タスクと、各タスクの関連ドキュメントへのリンクです。

タスク領域	説明	関連資料
インスタンスクラス、ストレージ、PIOPS	本稼働用に DB インスタンスを作成する場合、インスタンスクラス、ストレージタイプ、およびプロビジョンド IOPS が Amazon RDS でどのように機能するか理解する必要があります。	Microsoft SQL Server の DB インスタンスクラスのサポート Amazon RDS ストレージタイプ
マルチ AZ 配置	本稼働 DB インスタンスは、マルチ AZ 配置を使用する必要があります。マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。SQL Server のマルチ AZ 配置は、SQL Server ネイティブの DBM または AG テクノロジーを使用して実装されます。	Amazon RDS でのマルチ AZ 配置の設定と管理 Microsoft SQL Server のデータベースミラーリングまたは Always On 可用性グループを使用したマルチ AZ 配置

タスク領域	説明	関連資料
Amazon Virtual Private Cloud (VPC)	AWS アカウントにデフォルト VPC がある場合、DB インスタンスがデフォルト VPC 内に自動的に作成されます。アカウントにデフォルト VPC がない場合、DB インスタンスを VPC に作成する必要があるときは、DB インスタンスを作成する前に VPC とサブネットグループを作成する必要があります。	VPC 内の DB インスタンスの使用
セキュリティグループ:	デフォルトでは、DB インスタンスが作成されると、アクセスを禁止するファイアウォールが設定されます。したがって、DB インスタンスにアクセスするために、正しい IP アドレスとネットワーク構成を備えたセキュリティグループを作成する必要があります。	セキュリティグループによるアクセス制御
パラメータグループ	DB インスタンスに特定のデータベースパラメータが必要になる場合は、DB インスタンスを作成する前にパラメータグループを作成する必要があります。	Amazon RDS のパラメータグループ
オプショングループ	DB インスタンスに特定のデータベースオプションが必要になる場合は、DB インスタンスを作成する前にオプショングループを作成する必要があります。	Microsoft SQL Server データベースエンジンのオプション

タスク領域	説明	関連資料
DB インスタンスへの接続	セキュリティグループを作成し、それを DB インスタンスに関連付けると、Microsoft SQL Server Management Studio などの標準的な SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。	SQL Server DB インスタンスへの接続
バックアップと復元	DB インスタンスを作成するとき、自動バックアップが作成されるように設定できます。完全バックアップファイル (.bak ファイル) を使用することで、データベースを手動でバックアップおよび復元することもできます。	バックアップの概要 ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート
モニタリング:	CloudWatch Amazon RDS メトリクス、イベント、および拡張モニタリングを使用することで、SQL Server DB インスタンスをモニタリングできます。	Amazon RDS コンソールでのメトリクスの表示 Amazon RDS イベントの表示
ログファイル	SQL Server DB インスタンスのログファイルにアクセスできます。	Amazon RDS ログファイルのモニタリング Amazon RDS for Microsoft SQL Server データベースのログファイル

SQL Server DB インスタンスを使用するための高度な管理タスクがあります。詳細については、次のドキュメントを参照してください。

- Amazon RDS for Microsoft SQL Server の一般的な DBA タスク.
- RDS for SQL Server による AWS Managed Active Directory の操作
- tempdb データベースへのアクセス

Microsoft SQL Server DB インスタンスの制限

DB インスタンスへの Microsoft SQL Server の Amazon RDS 実装には、注意が必要ないくつかの制限があります。

- DB インスタンスでサポートされるデータベースの最大数は、インスタンスクラスタイプと可用性モードのシングル AZ、マルチ AZ データベースミラーリング (DBM)、またはマルチ AZ 可用性グループ (AG) によって異なります。Microsoft SQL Server システムは、この制限にはカウントされません。

  次の表は、各インスタンスクラスタイプと可用性モードでサポートされるデータベースの最大数を示しています。この表は、あるインスタンスクラスタイプから別のインスタンスクラスタイプに移動するか、ある可用性モードから別の可用性モードに移行することができるのかを判断するのに役立ちます。ソース DB インスタンスに、ターゲットインスタンスのクラスタイプまたは可用性モードがサポートできる数より多いデータベースがある場合、DB インスタンスの変更は失敗します。リクエストのステータスは、[イベント] ウィンドウで確認できます。

インスタンスクラスのタイプ	RDS で設定された vCPU	Single-AZ	マルチ AZ (DBM)	マルチ AZ (Always On AG)
db.*.micro to db.*.medium	該当なし	30	該当なし	該当なし
db.*.large	該当なし	30	30	30
db.*.xlarge から db.*.16xlarge	4 vCPU ~ 64 vCPU	100	50	75
db.*.24xlarge〜 db.*.32xlarge	4 vCPU ~ 64 vCPU	100	50	75
db.*.24xlarge〜 db.*.32xlarge	96 vCPU ~ 128 vCPU	100	50	100

* は、インスタンスクラスの異なるタイプを表します。

例えば、DB インスタンスが、シングル AZ の db.*.16xlarge で実行されており、76 のデータベースを持っているとします。マルチ AZ の Always On AG を使用してアップグレードする DB インスタンスを変更します。DB インスタンスにターゲット設定でサポートできる以上のデータベースが含まれているため、このアップグレードは失敗します。インスタンスクラスタイプを db.*.24xlarge にアップグレードする場合には、その変更は成功します。

アップグレードが失敗すると、次のようなイベントおよびメッセージが表示されます。

- データベースインスタンスクラスを変更できません。インスタンスには 76 個のデータベースがありますが、変換後にサポートされるのは 75 個のみです。
- DB インスタンスクラスを マルチ AZ に変換できません。インスタンスには 76 のデータベースがありますが、変換後にサポートされるのは 75 のみです。

ポイントインタイムリストアまたはスナップショットリストアに失敗すると、次のようなイベントやメッセージが表示されます。

- データベースインスタンスが互換性のない復元になりました。インスタンスには 76 個のデータベースがありますが、変換後にサポートされるのは 75 個のみです。

- 以下のポートは、Amazon RDS 用に予約されているため、DB インスタンスの作成時には使用できません: `1234`, `1434`, `3260`, `3343`, `3389`, `47001`, および `49152-49156`。
- 169.254.0.0/16 の範囲内の IP アドレスからのクライアント接続は許可されていません。これは、ローカルリンクのアドレス指定に使用される Automatic Private IP Addressing Range (APIPA) です。
- DB インスタンスにソフトウェアの制限 (24 コア、4 ソケット、128 GB RAM) よりも多くのプロセッサがある場合、SQL Server Standard Edition は使用可能なプロセッサのサブセットのみを使用します。この例は、db.m5.24xlarge および db.r5.24xlarge インスタンスクラスです。

  詳細については、Microsoft のドキュメントの「Editions and supported features of SQL Server 2019 (15.x)」のスケール制限の表を参照してください。
- Amazon RDS for SQL Server では、msdb データベースへのデータのインポートがサポートされていません。
- SQL Server マルチ AZ 配置の DB インスタンスにあるデータベースの名前は変更できません。
- RDS for SQL Server で次の DB パラメータを設定する場合は、必ずこれらのガイドラインを使用してください。

  - `max server memory (mb)` >= 256 MB
  - `max worker threads` >= (論理CPUの数 * 7)

  DB パラメータの設定の詳細については、「Amazon RDS のパラメータグループ」を参照してください。
- SQL Server DB インスタンスの最大ストレージのサイズは次のとおりです。
  - 汎用 (SSD) ストレージ – すべてのエディションで 16 TiB
  - プロビジョンド IOPS ストレージ – すべてのエディションで 64 TiB
  - マグネティックストレージ – すべてのエディションで 1 TiB

  さらに大きいサイズのストレージを必要とするシナリオでは、複数の DB インスタンスにまたがるシャーディングを使用することによって制限を回避できます。このアプローチでは、シャーディングされたシステムに接続するアプリケーションに、データに依存するルーティングロジックが必要です。既存のシャーディングフレームワークを使用するか、カスタムコードを記述してシャーディングを有効にできます。既存のフレームワークを使用する場合、このフレームワークは DB インスタンスと同じサーバーのコンポーネントにインストールできません。
- SQL Server DB インスタンスの最小ストレージサイズは次のとおりです。
  - 汎用 (SSD) ストレージ – Enterprise、Standard、Web および Express Edition 向けは 20 GiB
  - プロビジョンド IOPS ストレージ – Enterprise、Standard、Web および Express Editions 向けは 20 GiB
  - マグネティックストレージ – Enterprise、Standard、Web および Express エディション向けは 20 GiB
- Amazon RDS では、RDS DB インスタンスと同じサーバー上でのこれらのサービスの実行をサポートしていません。
  - Data Quality Services
  - マスターデータサービス

  これらの機能を使用するには、SQL Server を Amazon EC2 インスタンスにインストールするか、オンプレミスの SQL Server インスタンスを使用します。このような場合、EC2 インスタンスまたは SQL Server インスタンスは、Amazon RDS の SQL Server DB インスタンスのマスターデータサービスサーバーとして機能します。Microsoft のライセンスポリシーに従って、Amazon EBS とともに Amazon EC2 インスタンスに SQL サーバーをインストールできます。
- Microsoft SQL Server の制限により、`DROP DATABASE` の実行が成功する前の時点に復元した場合、その時点のデータベースの状態が反映されない可能性があります。例えば、削除されたデータベースは通常、`DROP DATABASE` コマンドが発行される前の最大 5 分前の状態に復元されま

す。このタイプの復元は、削除されたデータベースに数分間に行われたトランザクションを復元できないことを意味します。この問題に対処するには、復元オペレーションが完了してから DROP DATABASE コマンドを再発行します。データベースを削除すると、そのデータベースのトランザクションログが削除されます。

- SQL Server の場合は、DB インスタンスを作成した後、データベースを作成します。データベース名は、通常の SQL Server の命名ルールに従いますが、以下の点が異なります。
  - データベース名を rdsadmin で始めることはできません。
  - 先頭や末尾にスペースやタブを含めることはできません。
  - 新しい行を作成する文字を含めることはできません。
  - 一重引用符 (') を含めることはできません。
- SQL Server Web Edition では、新しい RDS for SQL Server DB インスタンスを作成するときにのみ、開発/テストテンプレートを使用できます。
- SQL Server Web Edition は、パブリックおよびインターネットにアクセスできるウェブページ、ウェブサイト、ウェブアプリケーション、ウェブサービスをホストするウェブホストおよびウェブ VAP 向けに設計されています。詳細については、「Amazon RDS での Microsoft SQL Server のライセンス」を参照してください。

Microsoft SQL Server の DB インスタンスクラスのサポート

DB インスタンスの計算とメモリの容量は、DB インスタンスクラスによって決まります。必要な DB インスタンスクラスは、処理能力とメモリの要件によって異なります。詳細については、「DB インスタンスクラス」を参照してください。

Microsoft SQL Server でサポートされている DB インスタンスクラスの以下のリストは、参考のために示しています。最新のリストについては、RDS コンソール (https://console.aws.amazon.com/rds/) でご確認ください。

すべての DB インスタンスクラスが、サポートされているすべての SQL Server マイナーバージョンで使用できるわけではありません。例えば、db.r6i などの新しい DB インスタンスクラスは、古いマイナーバージョンでは使用できません。describe-orderable-db-instance-options AWS CLI コマンドを使用して、SQL Server のエディションとバージョンで使用可能な DB インスタンスクラスを調べることができます。

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
Enterprise Edition	db.t3.xlarge –db.t3.2xlarge	db.t3.xlarge –db.t3.2xlarge	db.t3.xlarge –db.t3.2xlarge	db.t3.xlarge –db.t3.2xlarge
	db.r5.xlarge –db.r5.24xlarge	db.r5.xlarge –db.r5.24xlarge	db.r5.xlarge –db.r5.24xlarge	db.r5.xlarge –db.r5.24xlarge
	db.r5b.xlarge –db.r5b.24xlarge	db.r5b.xlarge –db.r5b.24xlarge	db.r5b.xlarge –db.r5b.24xlarge	db.r5b.xlarge –db.r5b.24xlarge
	db.r5d.xlarge –db.r5d.24xlarge	db.r5d.xlarge –db.r5d.24xlarge	db.r5d.xlarge –db.r5d.24xlarge	db.r5d.xlarge –db.r5d.24xlarge
	db.r6i.xlarge –db.r6i.32xlarge	db.r6i.xlarge –db.r6i.32xlarge	db.r6i.xlarge –db.r6i.32xlarge	db.r6i.xlarge –db.r6i.32xlarge
	db.r7i.xlarge –db.r7i.48xlarge	db.r7i.xlarge –db.r7i.48xlarge	db.r7i.xlarge –db.r7i.48xlarge	db.r7i.xlarge –db.r7i.48xlarge
	db.r8a.xlarge –db.r8a.16xlarge	db.r8a.xlarge –db.r8a.16xlarge	db.r8a.xlarge –db.r8a.16xlarge	db.r8a.xlarge –db.r8a.16xlarge
	db.r8i.xlarge –db.r8i.96xlarge	db.r8i.xlarge –db.r8i.96xlarge	db.r8i.xlarge –db.r8i.96xlarge	db.r8i.xlarge –db.r8i.96xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
	db.m5.xlarge –db.m5.24xlarge	db.m5.xlarge –db.m5.24xlarge	db.m5.xlarge –db.m5.24xlarge	db.m5.xlarge –db.m5.24xlarge
	db.m5d.xlarge –db.m5d.24xlarge	db.m5d.xlarge –db.m5d.24xlarge	db.m5d.xlarge –db.m5d.24xlarge	db.m5d.xlarge –db.m5d.24xlarge
	db.m6i.xlarge –db.m6i.32xlarge	db.m6i.xlarge –db.m6i.32xlarge	db.m6i.xlarge –db.m6i.32xlarge	db.m6i.xlarge –db.m6i.32xlarge
	db.m7i.xlarge –db.m7i.48xlarge	db.m7i.xlarge –db.m7i.48xlarge	db.m7i.xlarge –db.m7i.48xlarge	db.m7i.xlarge –db.m7i.48xlarge
	db.m8a.xlarge –db.m8a.16xlarge	db.m8a.xlarge –db.m8a.16xlarge	db.m8a.xlarge –db.m8a.16xlarge	db.m8a.xlarge –db.m8a.16xlarge
	db.m8i.xlarge –db.m8i.96xlarge	db.m8i.xlarge –db.m8i.96xlarge	db.m8i.xlarge –db.m8i.96xlarge	db.m8i.xlarge –db.m8i.96xlarge
	db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge
	db.x2m.xlarge –db.x2m.32xlarge	db.x2m.xlarge –db.x2m.32xlarge	db.x2m.xlarge –db.x2m.32xlarge	db.x2m.xlarge –db.x2m.32xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
	db.z1d.xlarge –db.z1d.12xlarge	db.z1d.xlarge –db.z1d.12xlarge	db.z1d.xlarge –db.z1d.12xlarge	db.z1d.xlarge –db.z1d.12xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
Standa Edition	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.8xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.r7i.large –db.r7i.16xlarge db.r8a.large –db.r8a.16xlarge db.r8i.large –db.r8i.16xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.8xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.r7i.large –db.r7i.12xlarge db.r8a.large –db.r8a.12xlarge db.r8i.large –db.r8i.12xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.24xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.r7i.large –db.r7i.12xlarge db.r8a.large –db.r8a.12xlarge db.r8i.large –db.r8i.12xlarge	db.t3.xlarge –db.t3.2xlarge db.r5.large –db.r5.24xlarge db.r5b.large –db.r5b.24xlarge db.r5d.large –db.r5d.24xlarge db.r6i.large –db.r6i.8xlarge db.r7i.large –db.r7i.12xlarge db.r8a.large –db.r8a.12xlarge db.r8i.large –db.r8i.12xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
	db.m5.large –db.m5.24xlarge	db.m5.large –db.m5.24xlarge	db.m5.large –db.m5.24xlarge	db.m5.large –db.m5.24xlarge
	db.m5d.large –db.m5d.24xlarge	db.m5d.large –db.m5d.24xlarge	db.m5d.large –db.m5d.24xlarge	db.m5d.large –db.m5d.24xlarge
	db.m6i.large –db.m6i.8xlarge	db.m6i.large –db.m6i.8xlarge	db.m6i.large –db.m6i.8xlarge	db.m6i.large –db.m6i.8xlarge
	db.m7i.large –db.m7i.16xlarge	db.m7i.large –db.m7i.12xlarge	db.m7i.large –db.m7i.12xlarge	db.m7i.large –db.m7i.12xlarge
	db.m8a.large –db.m8a.16xlarge	db.m8a.large –db.m8a.12xlarge	db.m8a.large –db.m8a.12xlarge	db.m8a.large –db.m8a.12xlarge
	db.m8i.large –db.m8i.16xlarge	db.m8i.large –db.m8i.12xlarge	db.m8i.large –db.m8i.12xlarge	db.m8i.large –db.m8i.12xlarge
	db.x2iedn.xlarge –db.x2iedn.8xlarge	db.x2iedn.xlarge –db.x2iedn.8xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge	db.x2iedn.xlarge –db.x2iedn.32xlarge
	db.x2m.xlarge –db.x2m.16xlarge	db.x2m.xlarge –db.x2m.8xlarge	db.x2m.xlarge –db.x2m.8xlarge	db.x2m.xlarge –db.x2m.8xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
	db.z1d.large –db.z1d.12xlarge	db.z1d.large –db.z1d.12xlarge	db.z1d.large –db.z1d.12xlarge	db.z1d.large –db.z1d.12xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
Web Edition	該当しない	db.t3.small –db.t3.xlarge db.r5.large –db.r5.4xlarge db.r5b.large –db.r5b.4xlarge db.r5d.large –db.r5d.4xlarge db.r6i.large –db.r6i.4xlarge db.r7i.large –db.r7i.4xlarge db.r8a.large –db.r8a.8xlarge db.r8i.large –db.r8i.8xlarge	db.t3.small –db.t3.2xlarge db.r5.large –db.r5.4xlarge db.r5b.large –db.r5b.4xlarge db.r5d.large –db.r5d.4xlarge db.r6i.large –db.r6i.4xlarge db.r7i.large –db.r7i.4xlarge db.r8a.large –db.r8a.8xlarge db.r8i.large –db.r8i.8xlarge	db.t3.small –db.t3.2xlarge db.r5.large –db.r5.4xlarge db.r5b.large –db.r5b.4xlarge db.r5d.large –db.r5d.4xlarge db.r6i.large –db.r6i.4xlarge db.r7i.large –db.r7i.4xlarge db.r8a.large –db.r8a.8xlarge db.r8i.large –db.r8i.8xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
		db.m5.large –db.m5.4xlarge db.m5d.large –db.m5d.4xlarge db.m6i.large –db.m6i.4xlarge db.m7i.large –db.m7i.4xlarge db.m8a.large –db.m8a.8xlarge db.m8i.large –db.m8i.8xlarge db.z1d.large –db.z1d.3xlarge	db.m5.large –db.m5.4xlarge db.m5d.large –db.m5d.4xlarge db.m6i.large –db.m6i.4xlarge db.m7i.large –db.m7i.4xlarge db.m8a.large –db.m8a.8xlarge db.m8i.large –db.m8i.8xlarge db.z1d.large –db.z1d.3xlarge	db.m5.large –db.m5.4xlarge db.m5d.large –db.m5d.4xlarge db.m6i.large –db.m6i.4xlarge db.m7i.large –db.m7i.4xlarge db.m8a.large –db.m8a.8xlarge db.m8i.large –db.m8i.8xlarge db.z1d.large –db.z1d.3xlarge
Express Edition	db.t3.micro –db.t3.xlarge	db.t3.micro –db.t3.xlarge	db.t3.micro –db.t3.xlarge	db.t3.micro –db.t3.xlarge

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
Develo Editior (2022 以前)	該当しない	db.m6i.xlarge –db.m6i.32xlarge db.m8a.xlarge –db.m8a.16xlarge db.m8i.xlarge –db.m8i.96xlarge db.r6i.xlarge –db.r6i.32xlarge db.r8a.xlarge –db.r8a.16xlarge db.r8i.xlarge –db.r8i.96xlarge db.x2m.xlarge –db.x2m.32xlarge	該当しない	該当しない

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
Develo Edition EE (SQL Server 2025)	db.r6i.xlarge –db.r6i.32xlarge db.r7i.xlarge –db.r7i.48xlarge db.m6i.xlarge –db.m6i.32xlarge db.m7i.xlarge –db.m7i.48xlarge db.r8i.large –db.r8i.96xlarge db.m8i.large –db.m8i.96xlarge db.r8a.large –db.r8a.16xlarge db.m8a.large –db.m8a.16xlarge	該当しない	該当しない	該当しない

SQL Server エディション	2025 サポートの範囲	2022 サポートの範囲	2019 サポートの範囲	2017 および 2016 サポートの範囲
	db.x2m.xlarge –db.x2m.32xlarge			
Develo Edition SE (SQL Server 2025)	db.r7i.large –db.r7i.12xlarge db.m7i.large –db.m7i.12xlarge db.r8i.large –db.r8i.12xlarge db.m8i.large –db.m8i.12xlarge db.r8a.large –db.r8a.12xlarge db.m8a.large –db.m8a.12xlarge db.x2m.xlarge –db.x2m.16xlarge	該当しない	該当しない	該当しない

Note

- 第 7 世代のインスタンスクラス以降、インスタンスサイズが 2xlarge 以上の RDS SQL Server ではハイパースレッディングは無効になっています。これにより、使用可能な vCPU の総数は、対応する EC2 インスタンスでサポートされている数の半分になります。例えば、EC2 インスタンスタイプ `m7i.2xlarge` はデフォルトで 4 つのコアと threadsPerCore 2 をサポートしているため、合計 8 つの vCPU になります。これとは対照的に、ハイパースレッディングが無効になっている RDS for SQL Server `db.m7i.2xlarge` インスタンスでは、4 つのコアと threadsPerCore 1 で、合計 4 つの vCPU になります。
- Amazon RDS for SQL Server は、Amazon EC2 X2iedn インスタンスをベースとしたメモリ最適化 X2M データベースインスタンスの提供を開始しました。RDS SQL Server では、X2m データベースインスタンスのインスタンスサイズが 2xlarge 以上の場合、ハイパースレッディングは無効になります。これにより、使用可能な vCPU の総数は、対応する EC2 インスタンスでサポートされている数の半分になります。例えば、EC2 インスタンスタイプ `x2m.2xlarge` (`x2iedn.2xlarge`) はデフォルトで 4 つのコアと threadsPerCore 2 をサポートしているため、合計 8 つの vCPU になります。これとは対照的に、ハイパースレッディングが無効になっている RDS for SQL Server `x2m.2xlarge` (`x2iedn.2xlarge`) インスタンスでは、4 つのコアと threadsPerCore 1 で、合計 4 つの vCPU になります。
- 第 7 世代インスタンスおよび X2M インスタンス以降、請求では RDS DB インスタンスとサードパーティーのライセンス料金の詳細な内訳が提供されます。詳細については、「RDS SQL Server の料金」を参照してください。
- AMD インスタンスの場合、各 vCPU は物理 CPU コアに対応し、コアごとに一貫したパフォーマンスを実現するように設計されています。2xlarge 以上の AMD インスタンスの場合、Amazon RDS はデフォルトで CPU コアの 50% を無効にします。この設定は、ワークロードのパフォーマンスを維持しながら、Windows および SQL Server ソフトウェアのライセンスコストを削減するのに役立ちます。追加の CPU 容量を有効にするには、CPU の最適化を使用して、使用可能なコアを最大 100% 有効にできます。インスタンスの料金はコア設定に関係なく同じであるため、vCPU 数をカスタマイズして、コンピューティング料金に影響を与えることなく、パフォーマンス要件とライセンスコストのバランスを取ることができます。xlarge 以下の AMD インスタンス (4 vCPU 以下) の場合、デフォルトで 100% のコアが有効になっており、コア数のカスタマイズは使用できません。これは、Microsoft が SQL Server ライセンスに対して最低 4 つの vCPU を課金するため、これらのインスタンスの vCPU の数を減らすことができないためです。

RDS for SQL Server インスタンスの CPU を最適化する

RDS for SQL Server では、プロセッサ機能を指定して CPU を最適化し、同じメモリと IOPS を維持しながら DB インスタンスの vCPU 数を設定できます。特定のデータベースワークロード要件に必要なメモリ対 CPU 比を実現し、vCPU 数に基づく Microsoft Windows OS と SQL Server のライセンスコストを削減できます。

プロセッサ機能を指定するには、次のパラメータを使用します。

```
--processor-features "Name=coreCount,Value=value" \
 "Name=threadsPerCore,Value=value"
```

- coreCount – DB インスタンスのライセンスコストを最適化するために、DB インスタンスの CPU コア数を指定します。選択したインスタンスタイプのコア数の許容値を確認するには、「CPU の最適化をサポートする DB インスタンスクラス」を参照してください。
- threadsPerCore – CPU コアあたりのスレッド数を定義するには、コアあたりのスレッドを指定します。選択したインスタンスタイプのコアあたりのスレッドの許容値を確認するには、「CPU の最適化をサポートする DB インスタンスクラス」を参照してください。

CPU 設定を最適化して RDS for SQL Server インスタンスを作成するためのサンプルコマンド:

```
aws rds create-db-instance \
    --engine sqlserver-ee \
    --engine-version 16.00 \
    --license-model license-included \
    --allocated-storage 300 \
    --master-username myuser \
    --master-user-password xxxxx \
    --no-multi-az \
    --vpc-security-group-ids myvpcsecuritygroup \
    --db-subnet-group-name mydbsubnetgroup \
    --db-instance-identifier my-rds-instance \
    --db-instance-class db.m7i.8xlarge \
    --processor-features "Name=coreCount,Value=8" "Name=threadsPerCore,Value=1"
```

この例では、デフォルトで coreCount が 16 の db.m7i.8xlarge インスタンスを作成します。CPU の最適化を使用すると、coreCount が 8 になり、有効な vCPU 数が 8 になります。

--processor-features パラメータなしでインスタンスを作成する場合、コア数は 16 に設定され、コアあたりのスレッド数はデフォルトで 1 に設定されるため、デフォルトの vCPU 数は 16 になります。

プロセッサ機能を指定する際に留意すべき考慮事項は次のとおりです。

- 作成 – 許可された値から processor-features パラメータに coreCount と threadsPerCore の両方を指定します。「CPU の最適化をサポートする DB インスタンスクラス」を参照してください。
- 変更 – CPU 設定の最適化で設定されたインスタンスクラスから CPU 設定の最適化をサポートするインスタンスクラスに変更する場合は、--use-default-processor-features パラメータを使用してデフォルトのプロセッサ設定を指定するか、変更リクエスト中にオプションを明示的に定義する必要があります。

> **Note**
>
> vCPU 数を変更すると、DB インスタンスに関連するライセンス料金に影響する可能性があります。

- スナップショットの復元 – スナップショットをソースと同じインスタンスタイプに復元する場合、復元された DB インスタンスはスナップショットから CPU 設定の最適化を継承します。別のインスタンスタイプに復元する場合は、ターゲットインスタンスの CPU 設定の最適化を定義するか、--use-default-processor-features パラメータを指定する必要があります。
- ポイントインタイムリストア – ポイントインタイムリストア (PITR) では、PITR の指定された時間に基づいて特定のスナップショットを復元し、その後、すべてのトランザクションログバックアップをそのスナップショットに適用して、インスタンスを指定されたポイントインタイムに移行します。PITR の場合、CPU 設定の最適化、coreCount および threadsPerCore は、PITR リクエスト中にカスタム値が指定されている場合を除き、ソーススナップショット (ポイントインタイムではない) から取得されます。使用されているソーススナップショットが CPU 設定の最適化で有効になっており、PITR に別のインスタンスタイプを使用している場合は、ターゲットインスタンスの CPU 設定の最適化を定義するか、–-use-default-processor-features パラメータを指定する必要があります。

制限事項

CPU の最適化を使用する際は、次の制限が適用されます。

- CPU の最適化では、Enterprise、Standard、Web (ライセンス込み)、および Enterprise、Standard、Developer (Bring Your Own Media) の各エディションをサポートしています。サポートされているインスタンスファミリーには、m7i、r7i、m8i、r8i、m8a、r8a、x2m などがあります。
- CPU の最適化では、インスタンスクラスに応じて異なる有効なコア設定が提供されます。詳細については、以下を参照してください。CPU の最適化をサポートする DB インスタンスクラス
- CPU コアの数のカスタマイズは、2xlarge 以上のインスタンスサイズでサポートされています。これらのインスタンスタイプでは、CPU の最適化でサポートされる vCPCU の最小数は 4 です。
- CPU の最適化では、CPU の最適化をサポートする第 7 世代以降のインスタンスでハイパースレッディングが無効になるため、コアあたり 1 つのスレッドしか許可されません。

CPU の最適化をサポートする DB インスタンスクラス

RDS for SQL Server は、第 7 世代のインスタンスクラスタイプで始まる CPU の最適化をサポートしています。さらに、RDS は、CPU の最適化機能が有効になっているかどうかにかかわらず、第 7 世代インスタンスクラスタイプから RDS DB インスタンスとサードパーティーライセンス料金の詳細な請求内訳を提供します。

RDS for SQL Server では、特定のインスタンスサイズで CPU の最適化がサポートされ、サポートされている最小インスタンスサイズは 2xlarge です。サポートされている最小設定は 4 vCPU です。次の表は、CPU コアのデフォルト値と有効な値、コアあたりの CPU スレッド、vCPU など、CPU の最適化をサポートする DB インスタンスクラスの概要を示しています。

Intel インスタンス

汎用インスタンス

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m7i.l, m8i.l	2	1	1	2

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m7i.xe, m8i.xe	4	2	1、2	2
m7i.2ge, m8i.2ge	4	4	1、2、3、4	1
m7i.4ge, m8i.4ge	8	8	1、2、3、4、5、6、7、8	1
m7i.8ge, m8i.8ge	16	16	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16	1
m7i.12rge, m8i.12rge	24	24	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m7i.1rge, m8i.1rge	32	32	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32	1
m7i.2rge	48	48	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32、33、34、35、36、37、38、39、40、41、42、43、44、45、46、47、48	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m8i.2rge	48	48	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1
m8i.3rge	64	64	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m7i.4rge	96	96	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64、66、68、70、72、74、76、78、80、82、84、86、888、90、92、94、96	1
m8i.4rge	96	96	6、9、12、15、18、21、24、27、30、33、36、39、42、45、48、51、54、57、60、63、66、69、72、75、78、81、84、87、90、93、96	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m8i.9rge	192	192	12、18、24、30、36、42、48、54、60、66、72、78、84、90、96、102、108、114、120、126、132、138、144、150、156、162、168、174、180、186、192	1

メモリ最適化インスタンス

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r7i.l, r8i.l	2	1	1	2
r7i.xe, r8i.xe	4	2	1、2	2

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r7i.2ge, r8i.2ge	4	4	1、2、3、4	1
r7i.4ge, r8i.4ge	8	8	1、2、3、4、5、6、7、8	1
r7i.8ge, r8i.8ge	16	16	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16	1
r7i.12rge, r8i.12rge	24	24	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r7i.1rge, r8i.1rge	32	32	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32	1
r7i.2rge	48	48	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32、33、34、35、36、37、38、39、40、41、42、43、44、45、46、47、48	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r8i.2rge	48	48	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48	1
r8i.3rge	64	64	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r7i.4rge	96	96	4、6、8、10、12、14、16、18、20、22、24、26、28、30、32、34、36、38、40、42、44、46、48、50、52、54、56、58、60、62、64、66、68、70、72、74、76、78、80、82、84、86、888、90、92、94、96	1
r8i.4rge	96	96	6、9、12、15、18、21、24、27、30、33、36、39、42、45、48、51、54、57、60、63、66、69、72、75、78、81、84、87、90、93、96	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r8i.9rge	192	192	12、18、24、30、36、42、48、54、60、66、72、78、84、90、96、102、108、114、120、126、132、138、144、150、156、162、168、174、180、186、192	1

X2M インスタンス

Amazon RDS for SQL Server は、Amazon EC2 X2iedn インスタンスをベースとしたメモリ最適化 X2M データベースインスタンスの提供を開始しました。RDS SQL Server では、X2m データベースインスタンスのインスタンスサイズが 2xlarge 以上の場合、ハイパースレッディングは無効になります。これにより、使用可能な vCPU の総数は、対応する EC2 インスタンスでサポートされている数の半分になります。例えば、EC2 インスタンスタイプ `x2m.2xlarge` (`x2iedn.2xlarge`) はデフォルトで 4 つのコアと threadsPerCore 2 をサポートしているため、合計 8 つの vCPU になります。これとは対照的に、ハイパースレッディングが無効になっている RDS for SQL Server `x2m.2xlarge` (`x2iedn.2xlarge`) インスタンスでは、4 つのコアと threadsPerCore 1 で、合計 4 つの vCPU になります。

バンドルされていない料金では、データベースのコストは、インスタンス料金 (CPU 時間あたりの料金) とライセンス料金 (vCPU 時間あたりの料金) で個別に計算されます。料金の詳細については、「Amazon RDS for SQL Server の料金」を参照してください。次の表は、CPU コアのデフォルト値と有効な値、コアあたりの CPU スレッド、vCPU など、CPU の最適化をサポートする X2M インスタンスクラスの概要を示しています。

メモリ最適化インスタンス

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
x2m.xe	4	2	1、2	2
x2m.2ge	4	4	1、2、3、4	1
x2m.4ge	8	8	1、2、3、4、5、6、7、8	1
x2m.8ge	16	16	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16	1
x2m.16rge	32	32	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32	1
x2m.24rge	48	48	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
			22、23、24、25、26、27、28、29、30、31、32、33、34、35、36、37、38、39、40、41、42、43、44、45、46、47、48	
x2m.large	64	64	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、24、25、26、27、28、29、30、31、32、33、34、35、36、37、38、39、40、41、42、43、44、45、46、47、48、49、50、51、52、53、54、55、56、57、58、59、60、61、62、63、64	1

AMD インスタンス

同等の Intel インスタンスとは異なり、AMD インスタンスの各 vCPU は物理 CPU コアに対応し、コアごとに一貫したパフォーマンスを実現するように設計されています。

2xlarge 以上の AMD インスタンスの場合、Amazon RDS はデフォルトで CPU コアの 50% を無効にします。この設定は、ワークロードのパフォーマンスを維持しながら、Windows および SQL Server ソフトウェアのライセンスコストを削減するのに役立ちます。追加の CPU 容量を有効にするには、CPU の最適化機能を使用して、使用可能なコアの最大 100% を有効にできます。インスタンスの料金はコア設定に関係なく同じであるため、vCPU 数をカスタマイズして、コンピューティング料金に影響を与えることなく、パフォーマンス要件とライセンスコストのバランスを取ることができます。

xlarge 以下の AMD インスタンス (4 vCPU 以下) の場合、デフォルトで 100% のコアが有効になっており、コア数のカスタマイズは使用できません。これは、Microsoft が SQL Server ライセンスに対して最低 4 つの vCPU を課金するため、これらのインスタンスの vCPU の数を減らすことができないためです。

バンドルされていない料金では、データベースのコストは、インスタンス料金 (CPU 時間あたりの料金) とライセンス料金 (vCPU 時間あたりの料金) で個別に計算されます。料金の詳細については、「Amazon RDS for SQL Server の料金」を参照してください。次の表は、CPU コアのデフォルト値と有効な値、コアあたりの CPU スレッド、vCPU など、CPU の最適化をサポートする AMD インスタンスクラスの概要を示しています。

汎用インスタンス

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m8a.l	2	2	1、2	1
m8a.x e	4	4	1、2、3、4	1
m8a.2 ge	4	4	1、2、3、4、5、6、7、8	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
m8a.4ge	8	8	1、2、4、6、8、10、12、14、16	1
m8a.8ge	16	16	1、2、3、4、8、12、16、20、24、28、32	1
m8a.16rge	24	24	1、2、3、4、5、6、12、18、24、30、36、42、48	1
m8a.24rge	32	32	1、2、3、4、5、6、7、8、16、24、32、40、48、56、64	1

メモリ最適化インスタンス

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r8a.large	2	2	1、2	1
r8a.xlarge	4	4	1、2、3、4	1

インスタンスタイプ	デフォルト vCPU	デフォルトの CPU コア	有効な CPU コア	コアあたりの有効なスレッド
r8a.2ge	4	4	1、2、3、4、5、6、7、8	1
r8a.4ge	8	8	1、2、4、6、8、10、12、14、16	1
r8a.8ge	16	16	1、2、3、4、8、12、16、20、24、28、32	1
r8a.12rge	24	24	1、2、3、4、5、6、12、18、24、30、36、42、48	1
r8a.16rge	32	32	1、2、3、4、5、6、7、8、16、24、32、40、48、56、64	1

DB インスタンスクラスの CPU コア数と CPU コアあたりのスレッド数の設定

次のオペレーションを実行するときに、DB インスタンスクラスの CPU コア数とコアあたりのスレッド数を設定できます。

- Amazon RDS DB インスタンスの作成
- Amazon RDS DB インスタンスを変更する
- DB インスタンスへの復元
- Amazon RDS の DB インスタンスを特定の時点に復元する

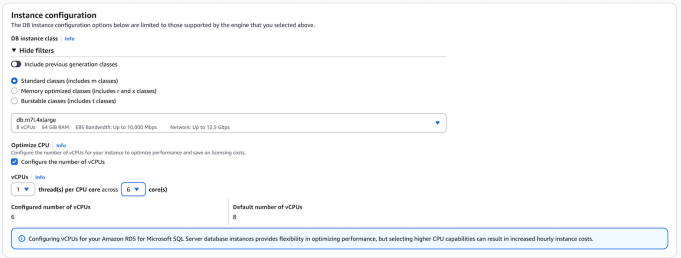
> **Note**
>
> DB インスタンスを変更してコアごとの CPU 数とスレッド数を指定する際に、インスタンスクラスを変更したときと同様に、短時間の停止が発生します。

AWS マネジメントコンソール、AWS CLI または RDS API を使用して CPU コアを設定します。

コンソール

コアを設定するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベースの作成] を選択します。
3. [インスタンス設定] オプションを設定する場合:
   a. [CPU を最適化] オプションを選択します。
   b. コア数を選択して [vCPU] オプションを設定します。



4. 他の選択が完了したら、[データベースの作成] を選択します。

AWS CLI

コアを設定するには

1. AWS CLI を使用して CPU の最適化を設定するには、コマンドに `--processor-features` オプションを含めます。`coreCount` および `threadsPerCore` を 1 にして、CPU コアの数を指定します。
2. 次の構文を使用します。

```
aws rds create-db-instance \
    --engine sqlserver-ee \
    --engine-version 16.00 \
    --license-model license-included \
    --allocated-storage 300 \
    --master-username myuser \
    --master-user-password xxxxx \
    --no-multi-az \
    --vpc-security-group-ids myvpcsecuritygroup \
    --db-subnet-group-name mydbsubnetgroup \
    --db-instance-identifier my-rds-instance \
    --db-instance-class db.m7i.4xlarge \
    --processor-features "Name=coreCount,Value=6" "Name=threadsPerCore,Value=1"
```

Example DB インスタンスクラスの有効なプロセッサ値を確認する

describe-orderable-db-instance-options コマンドを使用して、デフォルトの vCPU、コア、コアあたりのスレッドを表示します。例えば、次のコマンドの出力は db.r7i.2xlarge インスタンスクラスのプロセッサオプションを示します。

```
aws rds describe-orderable-db-instance-options --engine sqlserver-ee \
--db-instance-class db.r7i.2xlarge

Sample output:
-------------------------------------------------------------
|            DescribeOrderableDBInstanceOptions             |
+-----------------------------------------------------------+
||               OrderableDBInstanceOptions                ||
|+-----------------------------------+--------------------+|
||  DBInstanceClass                  |  db.r7i.2xlarge    ||
||  Engine                           |  sqlserver-ee      ||
||  EngineVersion                    |  13.00.6300.2.v1   ||
||  LicenseModel                     |  license-included  ||
||  MaxIopsPerDbInstance             |                    ||
||  MaxIopsPerGib                    |                    ||
||  MaxStorageSize                   |  64000             ||
||  MinIopsPerDbInstance             |                    ||
||  MinIopsPerGib                    |                    ||
||  MinStorageSize                   |  20                ||
||  MultiAZCapable                   |  True              ||
||  OutpostCapable                   |  False             ||
```

```
||  ReadReplicaCapable                |  True              ||
||  StorageType                       |  gp2               ||
||  SupportsClusters                  |  False             ||
||  SupportsDedicatedLogVolume        |  False             ||
||  SupportsEnhancedMonitoring        |  True              ||
||  SupportsGlobalDatabases           |  False             ||
||  SupportsIAMDatabaseAuthentication |  False             ||
||  SupportsIops                      |  False             ||
||  SupportsKerberosAuthentication    |  True              ||
||  SupportsPerformanceInsights       |  True              ||
||  SupportsStorageAutoscaling        |  True              ||
||  SupportsStorageEncryption         |  True              ||
||  SupportsStorageThroughput         |  False             ||
||  Vpc                               |  True              ||
|+-----------------------------------+--------------------+|
|||                  AvailabilityZones                   |||
||+------------------------------------------------------+||
|||                         Name                         |||
||+------------------------------------------------------+||
|||  us-west-2a                                          |||
|||  us-west-2b                                          |||
|||  us-west-2c                                          |||
||+------------------------------------------------------+||
|||             AvailableProcessorFeatures               |||
||+-----------------+-----------------+------------------+||
|||  AllowedValues  |  DefaultValue   |       Name       |||
||+-----------------+-----------------+------------------+||
|||  1,2,3,4        |  4              |  coreCount       |||
|||  1              |  1              |  threadsPerCore  |||
||+-----------------+-----------------+------------------+||
```

さらに、次のコマンドを実行して DB インスタンスのクラスのプロセッサ情報を取得できます。

- `describe-db-instances` - 指定された DB インスタンスのプロセッサ情報を示します
- `describe-db-snapshots` - 指定された DB スナップショットのプロセッサ情報を示します
- `describe-valid-db-instance-modifications` - 指定された DB インスタンスのプロセッサに対する有効な変更を示します

前述のコマンドの出力では、CPU の最適化が設定されていない場合、プロセッサ機能の値は `null` になります。

Example DB インスタンスの CPU コア数の設定

次の例では、CPU コアの数を 4、threadsPerCore を 1 に設定して *mydbinstance* を変更します。--apply-immediately を使用して変更をすぐに適用します。変更を次の予定されるメンテナンスウィンドウ中に適用するには、--apply-immediately option を省略します。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r7i.8xlarge \
    --processor-features "Name=coreCount,Value=4" "Name=threadsPerCore,Value=1" \
    --apply-immediately
```

Example DB インスタンスのデフォルトのプロセッサ設定に戻す

次の例では、*mydbinstance* をデフォルトのプロセッサ値に戻すことで変更します。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r7i.8xlarge \
    --use-default-processor-features \
    --apply-immediately
```

Microsoft SQL Server のセキュリティ

Microsoft SQL Server データベースエンジンではロールベースのセキュリティを使用します。DB インスタンスを作成する際に指定するマスターユーザー名は、processadmin、public、および setupadmin 固定サーバーロールのメンバーである SQL Server 認証のログインです。

データベースを作成するユーザーには、そのデータベースの db_owner ロールが割り当てられ、バックアップに使用されるものを除き、データベースレベルのアクセス許可がすべて付与されます。バックアップは、Amazon RDS により自動的に実行されます。

次のサーバーレベルのロールは、Amazon RDS for SQL Server では使用できません。

- bulkadmin
- dbcreator
- diskadmin
- securityadmin
- serveradmin

- sysadmin

RDS for SQL Server DB インスタンスでは、次のサーバーレベルの許可は使用できません。

- ALTER ANY DATABASE (任意のデータベースの変更)
- ALTER ANY EVENT NOTIFICATION (すべてのイベント通知の変更)
- ALTER RESOURCES (リソースの変更)
- ALTER SETTINGS (設定の変更。DB パラメータグループ API オペレーションを使用してパラメータを変更できます。詳細については、「Amazon RDS のパラメータグループ」を参照)
- AUTHENTICATE SERVER (サーバーの認証)
- CONTROL_SERVER (コントロールサーバー)
- CREATE DDL EVENT NOTIFICATION (DDL イベントの通知の作成)
- CREATE ENDPOINT (エンドポイントの作成)
- CREATE SERVER ROLE
- CREATE TRACE EVENT NOTIFICATION (追跡イベント通知の作成)
- DROP ANY DATABASE (任意のデータベースのドロップ)
- EXTERNAL ACCESS ASSEMBLY (外部アクセスアセンブリ)
- SHUTDOWN (代わりに RDS 起動オプションを使用できます)
- UNSAFE ASSEMBLY (安全ではないアセンブリ)
- 任意の可用性グループの変更
- 可用性グループを作成する

Microsoft SQL Server DB インスタンスの SSL サポート

アプリケーションと Microsoft SQL Server を実行する Amazon RDS DB インスタンス間の接続は、SSL を使用して暗号化できます。また、DB インスタンスへのすべての接続に SSL の使用を強制することができます。接続に SSL の使用を強制する場合、これはクライアントに対して透過的に行われ、クライアントは SSL を使用するための作業を行う必要はありません。

SSL は、すべての AWS リージョンおよび対応しているすべての SQL Server エディションでサポートされています。詳細については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。

Microsoft SQL Server DB インスタンスでの SSL の使用

クライアントアプリケーションと Microsoft SQL Server を実行する Amazon RDS DB インスタンス間の接続は、Secure Sockets Layer (SSL) を使用して暗号化できます。SSL サポートは、AWS のすべてのリージョンで、サポート対象の SQL Server のすべてのエディションでご利用いただけます。

SQL Server DB インスタンスを作成する際、Amazon RDS はそのインスタンスの SSL 証明書を作成します。SSL 証明書には、なりすまし攻撃から保護するために、SSL 証明書の共通名（CN）として DB インスタンスのエンドポイントが含まれています。

SSL を使用して SQL Server DB インスタンスに接続する方法は 2 とおりあります。

- すべての接続に SSL を強制する — これはクライアントに対して透過的に行われ、クライアントは SSL を使用するための作業を行う必要はありません。

  > **Note**
  >
  > `rds.force_ssl` を 1 に設定して SSMS バージョン 19.3、20.0、および 20.2 を使用する場合は、以下を確認します。
  >
  > - SSMS で [信頼サーバー証明書] を有効にします。
  > - システムに証明書をインポートします。

- 特定の接続を暗号化する — 特定のクライアントコンピュータから SSL 接続を確立します。接続を暗号化するためにクライアントで作業を行う必要があります。

SQL Server での Transport Layer Security (TLS) サポートについては、「Microsoft SQL Server 用の TLS 1.2 のサポート」を参照してください。

DB インスタンスへの接続に SSL を使用させる

DB インスタンスへのすべての接続に SSL の使用を強制することができます。接続に SSL の使用を強制する場合、これはクライアントに対して透過的に行われ、クライアントは SSL を使用するための作業を行う必要はありません。

SSL の使用を強制する場合、`rds.force_ssl` パラメータを使用します。デフォルトでは、`rds.force_ssl` パラメータが `0` (`off`) に設定されています。接続での SSL の使用を強制するには、`rds.force_ssl` パラメータを `1` (`on`) に設定します。`rds.force_ssl` パラメーターは静的であるため、値を変更した後にその変更を有効にするには、DB インスタンスを再起動する必要があります。

DB インスタンスへの接続で SSL の使用を強制するには

1. DB インスタンスにアタッチされているパラメータグループを決定します。

   a. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   b. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

   c. ナビゲーションペインで、[データベース] を選択し、詳細を表示する DB インスタンスの名前を選択します。

   d. [設定] タブを選択します。セクションで [パラメータグループ] を見つけます。

2. 必要に応じて、新しいパラメーターグループを作成します。DB インスタンスでデフォルトのパラメーターグループが使用されている場合は、新しいパラメーターグループを作成する必要があります。DB インスタンスでデフォルト以外のパラメーターグループが使用されている場合は、既存のパラメーターグループを編集するか、新しいパラメーターグループを作成するかを選択できます。既存のパラメーターグループを編集する場合、その変更は、そのパラメーターグループを使用するすべての DB インスタンスに適用されます。

   新しいパラメーターグループを作成するには、「Amazon RDS での DB パラメータグループの作成」の手順に従います。

3. 新規または既存のパラメーターグループを編集して、`rds.force_ssl` パラメーターを `true` に設定します。パラメーターグループを編集するには、「Amazon RDS の DB パラメータグループのパラメータの変更」の手順に従います。

4. 新しいパラメーターグループを作成した場合は、そのパラメーターグループがアタッチされるように DB インスタンスを変更します。DB インスタンスの [DB Parameter Group] 設定を変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

5. DB インスタンスを再起動します。詳細については、「 DB インスタンスの再起動」を参照してください。

特定の接続の暗号化

DB インスタンスへのすべての接続で SSL を使用するように強制することも、特定のクライアントコンピュータからの接続のみを暗号化することもできます。特定のクライアントからの接続に SSL を使用するには、クライアントコンピュータの証明書を取得し、クライアントコンピュータで証明書をインポートしてから、クライアントコンピュータからの接続を暗号化する必要があります。

> **Note**
>
> 2014 年 8 月 5 日以降に作成されたすべての SQL Server インスタンスは、SSL 証明書の共通名 (CN) フィールドで DB インスタンスのエンドポイントを使用します。2014 年 8 月 5 日より前は、VPC ベースの SQL Server のインスタンスで SSL 証明書を利用できませんでした。2014 年 8 月 5 日より前に作成された VPC ベースの SQL Server DB インスタンスの場合、SSL 証明書認証を使用し、そのインスタンスのエンドポイントを CN としてその DB インスタンスの SSL 証明書に含めるには、インスタンスの名前を変更します。DB インスタンスの名前を変更すると、新しい証明書がデプロイされ、新しい証明書を有効にするためにインスタンスは再起動されます。

クライアントコンピュータの証明書の取得

クライアントコンピュータから、Microsoft SQL Server を実行する Amazon RDS DB インスタンスへの接続を暗号化するには、クライアントコンピュータに証明書が必要です。

その証明書を取得するには、クライアントコンピュータに証明書をダウンロードします。すべてのリージョンで使用できるルート証明書は、ダウンロードできます。また、古いルート証明書と新しいルート証明書の両方を含む証明書バンドルもダウンロードできます。さらに、リージョン固有の中間証明書をダウンロードすることもできます。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

適切な証明書をダウンロードしたら、以下のセクションの手順に従って Microsoft Windows オペレーティングシステムに証明書をインポートします。

クライアントコンピュータでの証明書のインポート

以下の手順を使用して、クライアントコンピュータの Microsoft Windows オペレーティングシステムに証明書をインポートできます。

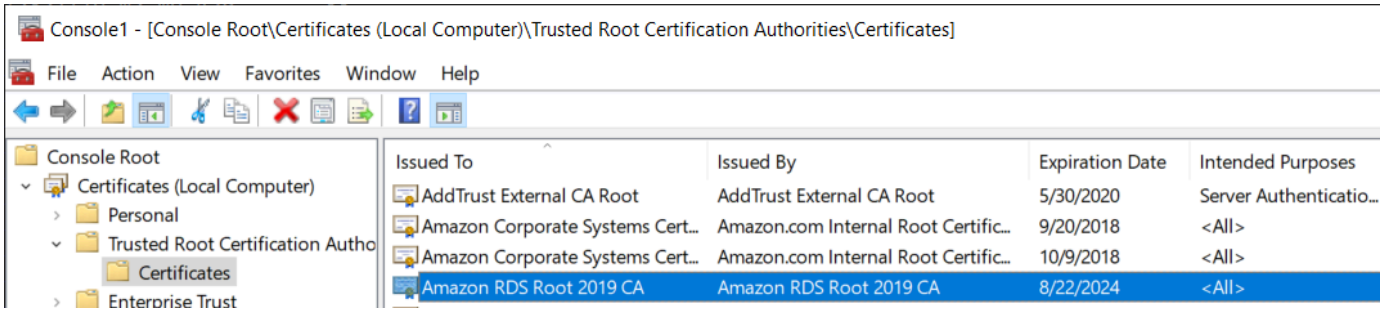
Windows オペレーティングシステムに証明書をインポートするには:

1. [Start] メニューで、検索ボックスに「**Run**」と入力し、Enter キーを押します。
2. [Open] ボックスに「**MMC**」と入力し、[OK] をクリックします。
3. MMC コンソールの [File] メニューで、[Add/Remove Snap-in] を選択します。
4. [Add or Remove Snap-ins] ダイアログボックスの [Available snap-ins] で [**Certificates**]、[Add] の順に選択します。

5. [Certificates snap-in] ダイアログボックスで、[Computer account]、[Next] の順に選択します。
6. [Select computer] ダイアログボックスで、[Finish] を選択します。
7. [Add or Remove Snap-ins] ダイアログボックスで、[OK] を選択します。
8. MMC コンソールで、[Certificates] を展開し、[Trusted Root Certification Authorities] のコンテキスト (右クリック) メニューを開いて、[All Tasks]、[Import] の順に選択します。
9. Certificate Import Wizard の最初のページで、[Next] を選択します。
10. Certificate Import Wizard の 2 番目のページで、[Browse] を選択します。参照ウィンドウで、ファイルタイプを [All files (*.*)] に変更する必要があります。.pem は証明書の標準の拡張子ではないためです。先ほどダウンロードした .pem ファイルを見つけます。

> **Note**
>
> SQL Server Management Studio (SSMS) などの Windows クライアントから接続する場合は、global-bundle.pem ファイルの代わりに PKCS#7 (.p7b) 証明書形式を使用することをお勧めします。.p7b 形式により、ルートおよび中間認証機関 (CA) を含む完全な証明書チェーンが Windows 証明書ストアに正しくインポートされます。これにより、.pem インポートでチェーン全体が正しくインストールされない可能性があるため、必須の暗号化が有効になっている場合に接続エラーが発生するのを防ぐことができます。

11. [Open] を選択して証明書ファイルを選択したら、[Next] を選択します。
12. Certificate Import Wizard の 3 番目のページで、[Next] を選択します。
13. Certificate Import Wizard の 4 番目のページで、[Finish] を選択します。インポートが成功したことを示すダイアログボックスが表示されます。
14. MMC コンソールで、[Certificates]、[Trusted Root Certification Authorities] の順に展開し、[Certificates] を選択します。次に示すように、証明書を探してそれが存在することを確認します。



Microsoft SQL Server を実行している Amazon RDS DB インスタンスへの接続の暗号化

クライアントコンピュータに証明書をインポートした後、クライアントコンピュータから、Microsoft SQL Server を実行する Amazon RDS DB インスタンスへの接続を暗号化できます。

SQL Server Management Studio では、以下の手順を使用します。SQL Server Management Studio の詳細については、「SQL Server Management Studio の使用」を参照してください。

SQL Server Management Studio からの接続を暗号化するには

1. SQL Server Management Studio を起動します。
2. [Connect to server] で、サーバー情報、ログインユーザー名、パスワードを入力します。
3. [Options] を選択します。
4. [Encrypt connection] を選択します。
5. [接続] を選択します。
6. 次のクエリを実行して接続が暗号化されていることを確認します。クエリが `true` を `encrypt_option` に対して返すことを確認します。

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

その他の SQL クライアントの場合は、以下の手順を実行します。

他の SQL クライアントからの接続を暗号化するには

1. 接続文字列に `encrypt=true` を追加します。この文字列はオプションとして、または GUI ツールの接続ページのプロパティとして使用できます。

> **Note**
>
> JDBC を使用して接続するクライアントの SSL 暗号化を有効にするには、Java CA 証明書 (cacerts) ストアへの Amazon RDS SQL 証明書の追加が必要になる場合があります。これは、キーツールユーティリティを使用して行うことができます。

2. 次のクエリを実行して接続が暗号化されていることを確認します。クエリが `true` を `encrypt_option` に対して返すことを確認します。

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

SQL Server のセキュリティプロトコルおよび暗号の設定

DB パラメータを使用して、特定のセキュリティプロトコルと暗号のオン/オフを切り替えることができます。設定できるセキュリティパラメータ (TLS バージョン 1.2 を除く) を次の表に示します。

DB パラメータ	許可される値 (デフォルトは太字)	説明
rds.tls10	デフォルト、有効、無効	TLS 1.0.
rds.tls11	デフォルト、有効、無効	TLS 1.1.
rds.tls12	default	TLS 1.2. この値は変更できません。
rds.fips	0、1	パラメータを 1 に設定すると、RDS は連邦情報処理規格 (FIPS) 140-2 標準に準拠したモジュールの使用を強制します。 詳細については、Microsoft のドキュメントの Use SQL Server 2016 in FIPS 140-2-compliant mode を参照してください。
rds.rc4	デフォルト、有効、無効	RC4 ストリーム暗号です。
rds.diffie-hellman	デフォルト、有効、無効	Diffie-Hellman キー交換暗号化。
rds.diffie-hellman-min-key-bit-length	デフォルト、1024、2048、3072、4096	Diffie-Hellman キーの最小ビット長。
rds.curve25519	デフォルト、有効、無効	Curve25519 elliptic-curve 暗号化暗号。このパラメータは、すべてのエンジンバージョン

DB パラメータ	許可される値 (デフォルトは太字)	説明
		でサポートされているわけではありません。
rds.3des168	デフォルト、有効、無効	168 ビットのキー長を持つトリプルデータ暗号化標準 (DES) 暗号化暗号。

> **Note**
>
> 16.00.4120.1、15.00.4365.2、14.00.3465.1、13.00.6435.1、および 12.00.6449.1 以降のマイナーエンジンバージョンでは、DB パラメータ `rds.tls10`、`rds.tls11`、`rds.rc4`、`rds.curve25519`、`rds.3des168` のデフォルト設定は無効になっています。それ以外の場合、デフォルト設定は有効です。
> 16.00.4120.1、15.00.4365.2、14.00.3465.1、13.00.6435.1、および 12.00.6449.1 以降のマイナーエンジンバージョンでは、`rds.diffie-hellman-min-key-bit-length` のデフォルト設定は 3072 です。それ以外の場合、デフォルト設定は 2048 です。

セキュリティプロトコルと暗号を設定するには、次のプロセスを使用します。

1. カスタム DB パラメータグループを作成します。
2. パラメータグループのパラメータを変更します。
3. DB パラメータグループを DB インスタンスに関連付けます。

DB パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

セキュリティ関連のパラメータグループの作成

DB インスタンスの SQL Server のエディションとバージョンに対応するセキュリティ関連パラメータのパラメータグループを作成します。

コンソール

以下の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [パラメータグループの作成] を選択します。
4. [パラメータグループの作成] ペインで、次の操作を行います。
   a. [パラメータグループファミリー] で、[sqlserver-se-13.0] を選択します。
   b. [グループ名] に、パラメータグループの識別子 (**sqlserver-ciphers-se-13** など) を入力します。
   c. [説明] に「**Parameter group for security protocols and ciphers**」と入力します。
5. [Create] (作成) を選択します。

CLI

以下の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name sqlserver-ciphers-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "Parameter group for security protocols and ciphers"
```

  Windows の場合:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name sqlserver-ciphers-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "Parameter group for security protocols and ciphers"
```

セキュリティ関連のパラメータの変更

DB インスタンスの SQL Server のエディションとバージョンに対応するパラメータグループのセキュリティ関連のパラメータを変更します。

コンソール

以下の手順では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。この例では、TLS バージョン 1.0 をオフにします。

パラメータグループを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [sqlserver-ciphers-se-13] などのパラメータグループを選択します。
4. [パラメータ] で、パラメータのリストを **rds** でフィルタ処理します。
5. [Edit parameters] を選択します。
6. [rds.tls10] を選択します。
7. [値] で、[無効] を選択します。
8. [Save changes] (変更の保存) をクリックします。

CLI

以下の手順では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。この例では、TLS バージョン 1.0 をオフにします。

パラメータグループを変更するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name sqlserver-ciphers-se-13 \
    --parameters
 "ParameterName='rds.tls10',ParameterValue='disabled',ApplyMethod=pending-reboot"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name sqlserver-ciphers-se-13 ^
    --parameters
 "ParameterName='rds.tls10',ParameterValue='disabled',ApplyMethod=pending-reboot"
```

セキュリティ関連のパラメータグループと DB インスタンスの関連付け

パラメータグループを DB インスタンスに関連付けるには、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

パラメータグループを新規または既存の DB インスタンスに関連付けることができます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれを関連付けます。詳しくは、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

パラメータグループを新規または既存の DB インスタンスに関連付けることができます。

パラメータグループを使用して DB インスタンスを作成するには

- パラメータグループの作成時に使用したものと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
```

```
    --allocated-storage 100 \
    --master-user-password secret123 \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --db-parameter-group-name sqlserver-ciphers-se-13
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --master-user-password secret123 ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --db-parameter-group-name sqlserver-ciphers-se-13
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

DB インスタンスを変更し、パラメータグループを関連付けるには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name sqlserver-ciphers-se-13 \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name sqlserver-ciphers-se-13 ^
    --apply-immediately
```

新しい SSL/TLS 証明書を使用して Microsoft SQL Server DB インスタンスに接続するようにアプリケーションを更新する

2023 年 1 月 13 日に Amazon RDS は、Secure Socket Layer または Transport Layer Security (SSL/TLS) を使用して RDS DB インスタンスに接続するための新しい認証局 (CA) 証明書を公開しました。ここでは、新しい証明書を使用するためのアプリケーションの更新について説明します。

このトピックでは、クライアントアプリケーションが SSL/TLS を使用して DB インスタンスに接続されているかどうかを判断できます。該当する場合はさらに、それらのアプリケーションの接続に証明書の検証が必要かどうかを確認できます。

> **Note**
>
> 一部のアプリケーションは、サーバー上の証明書を正常に検証できる場合にのみ、SQL Server DB インスタンスに接続されるように設定されています。
> そのようなアプリケーションの場合は、新しい CA 証明書を含むように、クライアントアプリケーションの信頼ストアを更新する必要があります。

クライアントアプリケーションの信頼ストアで CA 証明書を更新した後、DB インスタンスで証明書をローテーションできます。これらの手順を開発環境またはステージング環境でテストしてから、本番環境で実装することを強くお勧めします。

証明書のローテーションの詳細については、「SSL/TLS 証明書のローテーション」を参照してください。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。Microsoft SQL Server DB インスタンスで SSL/TLS を使用する方法については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。

トピック

- アプリケーションが SSL を使用して Microsoft SQL Server DB インスタンスに接続しているかどうかの確認
- クライアントが接続するために証明書の検証を必要とするかどうかの確認
- アプリケーション信頼ストアの更新

アプリケーションが SSL を使用して Microsoft SQL Server DB インスタンスに接続しているかどうかの確認

DB インスタンスの設定で rds.force_ssl パラメータの値を確認します。デフォルトでは、rds.force_ssl パラメータが 0 (オフ) に設定されています。rds.force_ssl パラメータが 1 (オン) に設定されている場合、クライアントは接続に SSL/TLS を使用する必要があります。パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

次のクエリを実行して、DB インスタンスへのすべての開いている接続に対する現在の暗号化オプションを取得します。接続が暗号化されている場合、列 ENCRYPT_OPTION は TRUE を返します。

```
select SESSION_ID,
    ENCRYPT_OPTION,
    NET_TRANSPORT,
    AUTH_SCHEME
    from SYS.DM_EXEC_CONNECTIONS
```

このクエリは現在の接続のみを表示します。過去に接続および切断したアプリケーションが SSL を使用したかどうかは表示しません。

クライアントが接続するために証明書の検証を必要とするかどうかの確認

異なるタイプのクライアントは、接続するために証明書の検証が必要かどうかを確認できます。

> **Note**
>
> リストにあるもの以外のコネクタを使用する場合、暗号化接続を強制する方法については、具体的なコネクタのドキュメントを参照してください。詳細については、Microsoft SQL Server のドキュメントの「Connection modules for Microsoft SQL databases」を参照してください。

SQL Server Management Studio

SQL Server Management Studio 接続に暗号化が実行されているかどうかを確認します。

1. SQL Server Management Studio を起動します。
2. [サーバーに接続] に、サーバー情報、ログインユーザー名、パスワードを入力します。

3. [Options] を選択します。
4. 接続ページで [暗号化接続] が選択されているかどうかを確認します。

SQL Server Management Studio の詳細については、「SQL Server Management Studio の使用」を参照してください。

Sqlcmd

次の sqlcmd クライアントの例では、スクリプトの SQL Server 接続をチェックして、正常な接続に有効な証明書が必要かどうかを確認する方法を示します。詳細については、Microsoft SQL Server のドキュメントの「Connecting with sqlcmd」を参照してください。

sqlcmd を使用するとき、次の例のように、接続を暗号化するために -N コマンドを使用する場合、SSL 接続にはサーバー証明書に対する検証が必要です。

```
$ sqlcmd -N -S dbinstance.rds.amazon.com -d ExampleDB
```

> **Note**
>
> sqlcmd が -C オプションで呼び出される場合、クライアント側の信頼ストアと一致しない場合でも、サーバー証明書を信頼します。

ADO.NET

次の例では、アプリケーションは SSL を使用して接続し、サーバー証明書の検証が必要です。

```
using SQLC = Microsoft.Data.SqlClient;

...

    static public void Main()
    {
        using (var connection = new SQLC.SqlConnection(
            "Server=tcp:dbinstance.rds.amazon.com;" +
            "Database=ExampleDB;User ID=LOGIN_NAME;" +
            "Password=YOUR_PASSWORD;" +
            "Encrypt=True;TrustServerCertificate=False;"
```

```
            ))
        {
            connection.Open();
            ...
        }
```

Java

次の例では、アプリケーションは SSL を使用して接続し、サーバー証明書の検証が必要です。

```
String connectionUrl =
    "jdbc:sqlserver://dbinstance.rds.amazon.com;" +
    "databaseName=ExampleDB;integratedSecurity=true;" +
    "encrypt=true;trustServerCertificate=false";
```

JDBC を使用して接続するクライアントの SSL 暗号化を有効にするには、Java CA 証明書ストアへの Amazon RDS 証明書の追加が必要になる場合があります。手順については、Microsoft SQL Server ドキュメントの「暗号化のためのクライアントの構成」を参照してください。`trustStore=`*`path-to-certificate-trust-store-file* を接続文字列に追加して、信頼された CA 証明書ファイル名を直接入力することもできます。

> **Note**
>
> 接続文字列で `TrustServerCertificate=true` (または同等のもの) を使用する場合、接続プロセスでは、信頼チェーンの検証をスキップします。この場合、証明書が確認できない場合でも、アプリケーションは接続します。`TrustServerCertificate=false` を使用すると、証明書の検証が強制され、これはベストプラクティスです。

アプリケーション信頼ストアの更新

Microsoft SQL Server を使用するアプリケーションの信頼ストアを更新できます。手順については、特定の接続の暗号化 を参照してください。また、Microsoft SQL Server ドキュメントの「暗号化のためのクライアントの構成」を参照してください。

Microsoft Windows 以外のオペレーティングシステムを使用している場合、新しいルート CA 証明書の追加については、SSL/TLS 実装のためのソフトウェアディストリビューションのドキュメントを

参照してください。例えば、OpenSSL や GnuTLS は人気のあるオプションです。この実装方法を使用して、RDS ルート CA 証明書に信頼を追加します。Microsoft では、一部のシステムで証明書を設定する手順を提供しています。

ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

証明書をインポートするサンプルスクリプトについては、証明書を信頼ストアにインポートするためのサンプルスクリプト を参照してください。

> **Note**
>
> 信頼ストアを更新するとき、新しい証明書を追加できるだけでなく、古い証明書を保持できます。

Microsoft SQL Server DB インスタンス用のコンプライアンスプログラムサポート

AWS対象範囲内の のサービスは、サードパーティーの監査人によって十分に評価され、認定、コンプライアンスの証明、または Authority to Operate (ATO) が与えられます。詳細については、コンプライアンスプログラムによる対象範囲内の AWS のサービスを参照してください。

Microsoft SQL Server の DB インスタンス用の HIPAA サポート

Amazon RDS for Microsoft SQL Server データベースを使用して、HIPAA 準拠アプリケーションを構築できます。AWS との事業提携契約 (BAA) に基づいて、保護されるべき医療情報 (PHI) を含め、医療関連の情報を保存できます。詳細については、「HIPAA コンプライアンス」を参照してください。

Amazon RDS for SQL Server では、以下のバージョンおよびエディションの HIPAA をサポートしています。

- SQL Server 2025 Enterprise および Standard Edition
- SQL Server 2022 Enterprise、Standard、および Web Edition
- SQL Server 2019 Enterprise、Standard、および Web Edition
- SQL Server 2017 Enterprise、Standard、および Web Edition

- SQL Server 2016 Enterprise、Standard、および Web Edition

DB インスタンスで HIPAA サポートを有効にするには、次の 3 つのコンポーネントを設定します。

コンポーネント	詳細
監査	監査を設定するには、`rds.sqlserver_audit` パラメータを `fedramp_hipaa` の値に設定します。DB インスタンスがカスタム DB パラメータグループを使用していない場合は、`rds.sqlserver_audit` パラメータを変更する前に、カスタムパラメータグループを作成し、DB インスタンスにアタッチする必要があります。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
送信の暗号化	送信の暗号化を設定するには、DB インスタンスへのすべての接続で Secure Sockets Layer (SSL) を強制的に使用するようにします。詳細については、「DB インスタンスへの接続に SSL を使用させる」を参照してください。
保管時の暗号化	保管時の暗号化を設定するには、2 つのオプションがあります。 1. SQL Server 2016–2022 Enterprise Edition または 2022 Standard Edition を実行している場合、Transparent Data Encryption (TDE) を使用して、保管時の暗号化を実現できます。詳細については、「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。 2. AWS Key Management Service (AWS KMS) 暗号化キーを使用して、保管時の暗号化を設定できます。詳細については、「Amazon RDS リソースの暗号化」を参照してください。

Amazon RDS での Microsoft SQL Server バージョン

新しい DB インスタンスを作成するときは、現在サポートされているいずれかの Microsoft SQL Server バージョンを指定できます。Microsoft SQL Server メジャーバージョン (Microsoft SQL Server 14.00 など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいず

れかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合は、Amazon RDS では、お客様が指定したメジャーバージョンの最新リリースにデフォルトで設定されます。

次の表は、すべての工ディションおよびすべての AWS リージョンでサポートされている SQL Server バージョンを示しています (例外については特記します)。

> **Note**
>
> また、describe-db-engine-versions AWS CLI コマンドを使用して、サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示することもできます。describe-db-major-engine-versions AWS CLI コマンドを実行するか、DescribeDBMajorEngineVersions RDS API オペレーションを使用して、SQL Server データベースのメジャーバージョンを表示できます。

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2025	17.00.4055.5 (CU6)	17.00.4055.5.v1
	17.00.4045.5 (CU5)	17.00.4045.5.v1
SQL Server 2022	16.00.4255.1 (CU25)	16.00.4255.1.v1
	16.00.4250.1 (CU24 GDR)	16.00.4250.1.v1
	16.00.4245.2 (CU24)	16.00.4245.2.v1
	16.00.4236.2 (CU23)	16.00.4236.2.v1
	16.00.4230.2 (CU22 GDR)	16.00.4230.2.v1
	16.00.4225.2 (CU22)	16.00.4225.2.v1
	16.00.4215.2 (CU21)	16.00.4215.2.v1
	16.00.4210.1 (CU20 GDR)	16.00.4210.1.v1
	16.00.4205.1 (CU20)	16.00.4205.1.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
	16.00.4195.2 (CU19)	`16.00.4195.2.v1`
	16.00.4185.3 (CU18)	`16.00.4185.3.v1`
	16.00.4175.1 (CU17)	`16.00.4175.1.v1`
	16.00.4165.4 (CU16)	`16.00.4165.4.v1`
	16.00.4150.1 (CU15)	`16.00.4150.1.v1`
	16.00.4140.3 (CU14 GDR)	`16.00.4140.3.v1`
	16.00.4135.4 (CU14)	`16.00.4135.4.v1`
	16.00.4131.2 (CU13)	`16.00.4131.2.v1`
	16.00.4125.3 (CU13)	`16.00.4125.3.v1`
	16.00.4120.1 (CU12 GDR)	`16.00.4120.1.v1`
	16.00.4115.5 (CU12)	`16.00.4115.5.v1`
	16.00.4105.2 (CU11)	`16.00.4105.2.v1`
	16.00.4095.4 (CU10)	`16.00.4095.4.v1`
	16.00.4085.2 (CU9)	`16.00.4085.2.v1`

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2019	15.00.4470.1 (CU32 GDR)	15.00.4470.1.v1
	15.00.4465.1 (CU32 GDR)	15.00.4465.1.v1
	15.00.4460.4 (CU32 GDR)	15.00.4460.4.v1
	15.00.4455.2 (CU32 GDR)	15.00.4455.2.v1
	15.00.4445.1 (CU32 GDR)	15.00.4445.1.v1
	15.00.4440.1 (CU32 GDR)	15.00.4440.1.v1
	15.00.4435.7 (CU32)	15.00.4435.7.v1
	15.00.4430.1 (CU32)	15.00.4430.1.v1
	15.00.4420.2 (CU31)	15.00.4420.2.v1
	15.00.4415.2 (CU30)	15.00.4415.2.v1
	15.00.4410.1 (CU29 GDR)	15.00.4410.1.v1
	15.00.4395.2 (CU28)	15.00.4395.2.v1
	15.00.4390.2 (CU28)	15.00.4390.2.v1
	15.00.4385.2 (CU28)	15.00.4385.2.v1
	15.00.4382.1 (CU27)	15.00.4382.1.v1
	15.00.4375.4 (CU27)	15.00.4375.4.v1
	15.00.4365.2 (CU26)	15.00.4365.2.v1
	15.00.4355.3 (CU25)	15.00.4355.3.v1
	15.00.4345.5 (CU24)	15.00.4345.5.v1
	15.00.4335.1 (CU23)	15.00.4335.1.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
	15.00.4322.2 (CU22)	`15.00.4322.2.v1`
	15.00.4316.3 (CU21)	`15.00.4316.3.v1`
	15.00.4312.2 (CU20)	`15.00.4312.2.v1`
	15.00.4236.7 (CU16)	`15.00.4236.7.v1`
	15.00.4198.2 (CU15)	`15.00.4198.2.v1`
	15.00.4153.1 (CU12)	`15.00.4153.1.v1`
	15.00.4073.23 (CU8)	`15.00.4073.23.v1`
	15.00.4043.16 (CU5)	`15.00.4043.16.v1`

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2017	14.00.3530.2 (CU31 GDR)	14.00.3530.2.v1
	14.00.3525.1 (CU31 GDR)	14.00.3525.1.v1
	14.00.3520.4 (CU31 GDR)	14.00.3520.4.v1
	14.00.3515.1 (CU31 GDR)	14.00.3515.1.v1
	14.00.3505.1 (CU31 GDR)	14.00.3505.1.v1
	14.00.3500.1.(CU31 GDR)	14.00.3500.1.v1
	14.00.3495.9 (CU31 GDR)	14.00.3495.9.v1
	14.00.3485.1 (CU31 GDR)	14.00.3485.1.v1
	14.00.3480.1 (CU31)	14.00.3480.1.v1
	14.00.3475.1 (CU31)	14.00.3475.1.v1
	14.00.3471.2 (CU31)	14.00.3471.2.v1
	14.00.3465.1 (CU31)	14.00.3465.1.v1
	14.00.3460.9 (CU31)	14.00.3460.9.v1
	14.00.3451.2 (CU30)	14.00.3451.2.v1
	14.00.3421.10 (CU27)	14.00.3421.10.v1
	14.00.3401.7 (CU25)	14.00.3401.7.v1
	14.00.3381.3 (CU23)	14.00.3381.3.v1
	14.00.3356.20 (CU22)	14.00.3356.20.v1
	14.00.3294.2 (CU20)	14.00.3294.2.v1
	14.00.3281.6 (CU19)	14.00.3281.6.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2016	13.00.6490.1 (GDR)	13.00.6490.1.v1
	13.00.6485.1 (GDR)	13.00.6485.1.v1
	13.00.6480.4 (GDR)	13.00.6480.4.v1
	13.00.6475.1 (GDR)	13.00.6475.1.v1
	13.00.6470.1 (GDR)	13.00.6470.1.v1
	13.00.6465.1 (GDR)	13.00.6465.1.v1
	13.00.6460.7 (GDR)	13.00.6460.7.v1
	13.00.6455.2 (GDR)	13.00.6455.2.v1
	13.00.6450.1 (GDR)	13.00.6450.1.v1
	13.00.6445.1 (GDR)	13.00.6445.1.v1
	13.00.6441.1 (GDR)	13.00.6441.1.v1
	13.00.6435.1 (GDR)	13.00.6435.1.v1
	13.00.6430.49 (GDR)	13.00.6430.49.v1
	13.00.6419.1 (SP3 + 修正プログラム)	13.00.6419.1.v1
	11.00.6020.0 (SP3)	13.00.6300.2.v1

Amazon RDS でのバージョン管理

Amazon RDS には、柔軟なバージョン管理機能があります。この機能を使用して、DB インスタンスでのパッチの適用やアップグレードのタイミングと方法を制御できます。そのため、DB エンジンに対して以下の操作を行うことができます。

- データベースエンジンのパッチバージョンとの互換性を維持する。

- 本番稼働環境にデプロイする前に、新しいパッチバージョンをテストして、アプリケーションで動作することを確認する。
- サービスレベルアグリーメント (SLA) とタイミング要件を満たすようにバージョンアップグレードを計画して実行する。

Amazon RDS での Microsoft SQL Server DB エンジンのパッチ適用

Amazon RDS では、Amazon RDS に固有の DB インスタンスエンジンバージョンに、Microsoft SQL Server データベースの公式パッチを定期的に統合します。各エンジンバージョンの Microsoft SQL Server パッチの詳細については、「Amazon RDS のバージョンと機能のサポート」を参照してください。

現在、エンジンのアップグレードはすべて、DB インスタンスで手動で実行しています。詳細については、「Microsoft SQL Server DB エンジンのアップグレード」を参照してください。

Amazon RDS の Microsoft SQL Server のメジャーエンジンバージョンの非推奨スケジュール

次の表は、Microsoft SQL Server のメジャーエンジンバージョンの計画された非推奨スケジュールを示しています。

日付	情報
2026 年 7 月 14 日	Microsoft は、SQL Server 2016 の重要なパッチの更新を停止します。詳細についてはントの Microsoft SQL Server 2016を参照してください。
2026 年 7 月 14 日	Amazon RDS は、SQL Server 用 RDS での Microsoft SQL Server 2016 のサポートをの時点で、残りのインスタンスは SQL Server 2019 (利用可能な最新のマイナーバー定です。詳細については、「Announcement: Amazon RDS for SQL Server ending su Server 2016」を参照してください。 Microsoft SQL Server 2016 から自動的にアップグレードされないように、都合の良グレードしてください。詳細については、「DB インスタンスのエンジンバージョン参照してください。

日付	情報
2026 年 1 月 15 日	Amazon RDS は、Microsoft SQL Server 2016 を使用する、SQL Server DB インスタ成の無効化を開始します。詳細については、「Announcement: Amazon RDS for SQ for Microsoft SQL Server 2016」を参照してください。
2024 年 7 月 9 日	Microsoft は、SQL Server 2014 の重要なパッチの更新を停止します。詳細について ントの Microsoft SQL Server 2014 を参照してください。
2024 年 6 月 1 日	Amazon RDS は、SQL Server 用 RDS での Microsoft SQL Server 2014 のサポートを の時点で、残りのインスタンスは SQL Server 2016 (利用可能な最新のマイナーバー 定です。詳細については、「Announcement: Amazon RDS for SQL Server ending of 2014 major versions」を参照してください。 Microsoft SQL Server 2014 から自動的にアップグレードされないように、都合の良 グレードしてください。詳細については、「DB インスタンスのエンジンバージョン 参照してください。
2022 年 7 月 12 日	Microsoft は、SQL Server 2012 の重要なパッチの更新を停止します。詳細について ントの Microsoft SQL Server 2012を参照してください。
2022 年 6 月 1 日	Amazon RDS は、SQL Server 用 RDS での Microsoft SQL Server 2012 のサポートを の時点で、残りのインスタンスは SQL Server 2014 (利用可能な最新のマイナーバー 予定です。詳細については、Announcement: Amazon RDS for SQL Server ending of 2012 major versions を参照してください。 Microsoft SQL Server 2012 から自動的にアップグレードされないように、都合の良 グレードしてください。(詳しくは、「DB インスタンスのエンジンバージョンのア てください。)
2021 年 9 月 1 日	Amazon RDS は、Microsoft SQL Server 2012 を使用する、SQL Server DB インスタ 作成の無効化を開始します。詳細については、Announcement: Amazon RDS for SQ support for SQL Server 2012 major versions を参照してください。

日付	情報
2019 年 7 月 12 日	Amazon RDS チームは、2019 年 6 月に Microsoft SQL Server 2008 R2 のサポートを soft SQL Server 2008 R2 の残りのインスタンスは、SQL Server 2012 (利用可能な最 ン) に移行されます。 Microsoft SQL Server 2008 R2 から自動的にアップグレードされないように、都合の プグレードします。詳細については、「DB インスタンスのエンジンバージョンのア してください。
2019 年 4 月 25 日	2019 年 4 月末、Microsoft SQL Server 2008R2 を使用して新しい Amazon RDS for S スインスタンスを作成することはできなくなります。

Amazon RDS for Microsoft SQL Server のバージョンポリシー

このトピックでは、サポートされているメジャーバージョンとマイナーバージョン、リリースタイムライン、廃止手順、アップグレードガイダンスなど、Amazon RDS for SQL Server のバージョンポリシーについて説明します。

Amazon RDS for SQL Server のメジャーバージョン

Amazon RDS は、SQL Server 2014、2016、2017、2019、2022、2025 など、Microsoft SQL Server のいくつかのメジャーバージョンをサポートしています。

Microsoft SQL Server メジャーバージョンのサポート

Microsoft SQL Server メジャーバージョン	RDS for SQL Server のサポート
2008	サポートは終了しました
2012	サポートは終了しました
2014	サポートは終了しました
2016	サポート対象
2017 年	サポート対象
2019	サポート対象

Microsoft SQL Server メジャーバージョン	RDS for SQL Server のサポート
2022	サポート対象
2025	サポート対象

Amazon RDS for SQL Server のマイナーバージョン

RDS for SQL Server のお客様は、新しい DB インスタンスを作成するときに、現在サポートされているいずれかの Microsoft SQL Server バージョンを指定できます。Microsoft SQL Server メジャーバージョン (Microsoft SQL Server 2022 など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいずれかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合は、Amazon RDS では、お客様が指定したメジャーバージョンの最新リリースにデフォルトで設定されます。

次の表は、すべての エディション とすべての AWS リージョンでサポートされているバージョンを示しています (例外については特記します)。サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示するには、`describe-db-engine-versions` CLI コマンドを使用することもできます。

Amazon RDS for SQL Server でサポートされているマイナーバージョン

メジャーバージョン	マイナーバージョン	RDS API **`EngineVersion`** および CLI **`engine-version`**
SQL Server 2025	17.00.4045.5 (CU5)	`17.00.4045.5.v1`
SQL Server 2022	16.00.4245.2 (CU24)	`16.00.4245.2.v1`
	16.00.4236.2 (CU23 v2)	`16.00.4236.2.v1`
	16.00.4230.2 (GDR)	`16.00.4230.2.v1`
	16.00.4225.2 (CU22)	`16.00.4225.2.v1`
	16.00.4215.2 (CU21)	`16.00.4215.1.v1`
	16.00.4210.1 (GDR)	

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
	16.00.4205.1 (CU20)	16.00.4210.1.v1
	16.00.4195.2 (CU19)	16.00.4205.1.v1
	16.00.4185.3 (CU18)	16.00.4195.2.v1
	16.00.4175.1 (CU17)	16.00.4185.3.v1
	16.00.4165.4 (CU16)	16.00.4175.1.v1
	16.00.4150.1 (GDR)	16.00.4165.4.v1
	16.00.4145.4 (CU15)	16.00.4150.1.v1
	16.00.4140.3 (GDR)	16.00.4145.4.v1
	16.00.4135.4 (CU14)	16.00.4140.3.v1
	16.00.4131.2 (CU13)	16.00.4135.4.v1
	16.00.4125.3 (CU13)	16.00.4131.2.v1
	16.00.4120.1 (GDR)	16.00.4125.3.v1
	16.00.4115.5 (CU12)	16.00.4120.1.v1
	16.00.4105.2 (CU11)	16.00.4115.5.v1
	16.00.4095.4 (CU10)	16.00.4105.2.v1
	16.00.4085.2 (CU9)	16.00.4095.4.v1
		16.00.4085.2.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2019	15.00.4460.4 (GDR)	15.00.4460.4.v1
	15.00.4455.2 (GDR)	15.00.4455.2.v1
	15.00.4445.1 (GDR)	15.00.4445.1.v1
	15.00.4440.1 (GDR)	15.00.4440.1.v1
	15.00.4435.7 (GDR)	15.00.4435.7.v1
	15.00.4430.1 (CU32)	15.00.4430.1.v1
	15.00.4420.2 (CU31)	15.00.4420.2.v1
	15.00.4415.2 (CU30)	15.00.4415.2.v1
	15.00.4410.1 (GDR)	15.00.4410.1.v1
	15.00.4405.4 (CU29)	15.00.4405.4.v1
	15.00.4395.2 (GDR)	15.00.4395.2.v1
	15.00.4390.2 (GDR)	15.00.4390.2.v1
	15.00.4385.2 (CU28)	15.00.4385.2.v1
	15.00.4382.1 (CU27)	15.00.4382.1.v1
	15.00.4375.4 (CU27)	15.00.4375.4.v1
	15.00.4365.2 (CU26)	15.00.4365.2.v1
	15.00.4355.3 (CU25)	15.00.4355.3.v1
	15.00.4345.5 (CU24)	15.00.4345.5.v1
	15.00.4335.1 (CU23)	15.00.4335.1.v1
	15.00.4322.2 (CU22)	15.00.4322.2.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
	15.00.4316.3 (CU21)	`15.00.4316.3.v1`
	15.00.4312.2 (CU20)	`15.00.4312.2.v1`
	15.00.4236.7 (CU16 SU)	`15.00.4236.7.v1`
	15.00.4198.2 (CU15)	`15.00.4198.2.v1`
	15.00.4153.1 (CU12)	`15.00.4153.1.v1`
	15.00.4073.23 (CU8)	`15.00.4073.23.v1`
	15.00.4043.16 (CU5)	`15.00.4043.16.v1`

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2017	14.00.3520.4 (GDR)	14.00.3520.4.v1
	14.00.3515.1 (GDR)	14.00.3515.1.v1
	14.00.3505.1 (GDR)	14.00.3505.1.v1
	14.00.3500.1 (GDR)	14.00.3500.1.v1
	14.00.3495.9 (GDR)	14.00.3495.9.v1
	14.00.3485.1 (GDR)	14.00.3485.1.v1
	14.00.3480.1 (GDR)	14.00.3480.1.v1
	14.00.3475.1 (GDR)	14.00.3475.1.v1
	14.00.3471.2 (GDR)	14.00.3471.2.v1
	14.00.3465.1 (GDR)	14.00.3465.1.v1
	14.00.3460.9 (CU31)	14.00.3460.9.v1
	14.00.3451.2 (CU30)	14.00.3451.2.v1
	14.00.3421.10 (CU27)	14.00.3421.10.v1
	14.00.3401.7 (CU25)	14.00.3401.7.v1
	14.00.3381.3 (CU23)	14.00.3381.3.v1
	14.00.3356.20 (CU22)	14.00.3356.20.v1
	14.00.3294.2 (CU20)	14.00.3294.2.v1
	14.00.3281.6 (CU19)	14.00.3281.6.v1

メジャーバージョン	マイナーバージョン	RDS API **EngineVersion** および CLI **engine-version**
SQL Server 2016	13.00.6480.4 (GDR)	13.00.6480.4.v1
	13.00.6475.1 (GDR)	13.00.6475.1.v1
	13.00.6470.1 (GDR)	13.00.6470.1.v1
	13.00.6465.1 (GDR)	13.00.6465.1.v1
	13.00.6460.7 (GDR)	13.00.6460.7.v1
	13.00.6455.2 (GDR)	13.00.6455.2.v1
	13.00.6450.1 (GDR)	13.00.6450.1.v1
	13.00.6445.1 (GDR)	13.00.6445.1.v1
	13.00.6441.1 (GDR)	13.00.6441.1.v1
	13.00.6435.1 (SP3 GDR)	13.00.6435.1.v1
	13.00.6430.49 (SP3 GDR)	13.00.6430.49.v1
	13.00.6419.1 (SP3 GDR)	13.00.6419.1.v1
	11.00.6020.0 (SP3)	13.00.6300.2.v1

Amazon RDS for SQL Server が新しいメジャーバージョンのサポートを導入するタイミング

Amazon RDS for SQL Server では、通常、Microsoft の一般提供リリース日から 6 ~ 12 か月以内に新しいメジャー SQL Server データベースバージョンのサポートが導入されます。Amazon RDS は、新しいメジャーバージョンのサポートを追加する際、安定性を確保するための包括的なテストを受けた最初のマイナーバージョンを選択します。

Amazon RDS for SQL Server メジャーバージョンが利用可能な期間

Amazon RDS for SQL Server は、Microsoft のサポート終了日 (EOS) までメジャーバージョンのサポートを継続します。(Microsoft のドキュメントを参照してください。) 以下は、テストとアップグレードのサイクルを計画する上で役立つ重要な日付です。これらの日付は、必要となる可能性のある最短のアップグレードタイムラインを示しており、Amazon によって後日延長される可能性があることに注意してください。

Amazon RDS for SQL Server メジャーバージョンのサポート終了日

SQL Server メジャーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
2025	2036/1/6	2036/1/6
2016	2026/7/14	2026/7/14
2017 年	2027/10/12	2027/10/12
2019	1/8/2030	1/8/2030
2022	1/11/2033	1/11/2033

Amazon RDS for SQL Server マイナーバージョンがリリースされる頻度

一般的に、Amazon RDS for SQL Server のマイナーバージョンは、Microsoft によって利用可能になってから 30 日以内にリリースされます。このリリーススケジュールは、追加の機能または修正を含めるために変更される場合があります。

Amazon RDS for SQL Server マイナーバージョンが利用可能な期間

一般的に、Amazon RDS for SQL Server はメジャーバージョンごとに最新の 3 つのマイナーバージョンをサポートしています。各メジャーバージョンに対応するマイナーバージョンの正確な数は、各マイナーバージョンの Microsoft のサポートタイムライン、Amazon RDS のメンテナンススケジュール、およびその他の要因によって異なる場合があります。

Amazon RDS for SQL Server マイナーバージョンのサポート終了日

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
SQL Server 2022	16.00.4245.2 (CU24)	1/11/2033	1/11/2033
	16.00.4236.2 (CU23 v2)		
	16.00.4230.2 (GDR)		
	16.00.4225.2 (CU22)		
	16.00.4215.2 (CU21)		
	16.00.4210.1 (GDR)		
	16.00.4205.1 (CU20)		
	16.00.4195.2 (CU19)		
	16.00.4185.3 (CU18)		
	16.00.4175.1 (CU17)		
	16.00.4165.4 (CU16)		
	16.00.4150.1 (GDR)		
	16.00.4145.4 (CU15)		
	16.00.4140.3 (GDR)		
	16.00.4135.4 (CU14)		
	16.00.4131.2 (CU13)		
	16.00.4125.3 (CU13)		
	16.00.4120.1 (GDR)		
	16.00.4115.5 (CU12)		

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
	16.00.4105.2 (CU11) 16.00.4095.4 (CU10) 16.00.4085.2 (CU9)		

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
SQL Server 2019	15.00.4460.4 (GDR) 15.00.4455.2 (GDR) 15.00.4445.1 (GDR) 15.00.4440.1 (GDR) 15.00.4435.7 (GDR) 15.00.4430.1 (CU32) 15.00.4420.2 (CU31) 15.00.4415.2 (CU30) 15.00.4410.1 (GDR) 15.00.4405.4 (CU29) 15.00.4395.2 (GDR) 15.00.4390.2 (GDR) 15.00.4385.2 (CU28) 15.00.4382.1 (CU27) 15.00.4375.4 (CU27) 15.00.4365.2 (CU26) 15.00.4355.3 (CU25) 15.00.4345.5 (CU24) 15.00.4335.1 (CU23) 15.00.4322.2 (CU22)	1/8/2030	1/8/2030

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
	15.00.4316.3 (CU21) 15.00.4312.2 (CU20) 15.00.4236.7 (CU16 SU) 15.00.4198.2 (CU15) 15.00.4153.1 (CU12) 15.00.4073.23 (CU8) 15.00.4043.16 (CU5)		

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
SQL Server 2017	14.00.3520.4 (GDR) 14.00.3515.1 (GDR) 14.00.3505.1 (GDR) 14.00.3500.1 (GDR) 14.00.3495.9 (GDR) 14.00.3485.1 (GDR) 14.00.3480.1 (GDR) 14.00.3475.1 (GDR) 14.00.3471.2 (GDR) 14.00.3465.1 (GDR) 14.00.3460.9 (CU31) 14.00.3451.2 (CU30) 14.00.3421.10 (CU27) 14.00.3401.7 (CU25) 14.00.3381.3 (CU23) 14.00.3356.20 (CU22) 14.00.3294.2 (CU20) 14.00.3281.6 (CU19)	2027/10/12	2027/10/12

メジャーバージョン	マイナーバージョン	Microsoft のサポート終了日	新しいバージョンへのアップグレード予定日
SQL Server 2016	13.00.6480.4 (GDR) 13.00.6475.1 (GDR) 13.00.6470.1 (GDR) 13.00.6465.1 (GDR) 13.00.6460.7 (GDR) 13.00.6455.2 (GDR) 13.00.6450.1 (GDR) 13.00.6445.1 (GDR) 13.00.6441.1 (GDR) 13.00.6435.1 (SP3 GDR) 13.00.6430.49 (SP3 GDR) 13.00.6419.1 (SP3 GDR) 11.00.6020.0 (SP3)	2026/7/14	2026/7/14

Amazon RDS for SQL Server データベースバージョンが廃止されるとどうなるか

Amazon RDS でデータベースエンジンのメジャーバージョンが非推奨になった場合、サポートされるメジャーバージョンへの手動アップグレードを開始するために、非推奨の発表から最低 6 か月の期間が設けられています。この期間が終了すると、スケジュールされたメンテナンスウィンドウ中に、非推奨バージョンを実行しているすべてのインスタンスに対して、次のメジャーバージョンへの自動アップグレードが適用されます。

Amazon RDS でデータベースエンジンのマイナーバージョンが廃止された場合、自動アップグレードを開始する前に、発表から 3 か月の期間が与えられます。この期間が終了すると、非推奨のマイナーバージョンを実行しているすべてのインスタンスは、スケジュールされたメンテナンスウィンドウ中に、サポートされている最新のマイナーバージョンへの自動アップグレードがスケジュールされます。

Amazon RDS でメジャーデータベースエンジンバージョンまたはマイナーデータベースエンジンバージョンが廃止されると、サポートされていないバージョンで作成された DB スナップショットから復元された DB インスタンスは、自動的にすぐに現在サポートされているバージョンにアップグレードされます。

Amazon RDS for SQL Server の必須アップグレード

Amazon RDS for SQL Server では、重要な修正が必要な場合、新しいマイナーバージョンへの必須アップグレードが必要になることがあります。これらのアップグレードを実装する前に、Amazon は主要なマイルストーンのタイミング、データベースインスタンスへの影響、推奨されるアクションを含む詳細な計画を通知します。これらの必須のアップグレードは自動化されており、インスタンスの指定されたメンテナンスウィンドウ中に開始するようにスケジュールされます。

アップグレード前に新しい SQL Server バージョンで DB インスタンスをテストする

アップグレードプロセスのテストや、新しいバージョンがアプリケーションとワークロードでどのように動作するかのテストを行うことができます。インスタンススナップショットから復元して、新しい RDS for SQL Server インスタンスを作成します。インスタンススナップショットは、既存の Amazon RDS インスタンスから自分で作成できます。Amazon RDS は、インスタンスの定期的なスナップショットも自動的に作成します。その後、新しいインスタンスに対するバージョンアップグレードをスタートします。元のインスタンスをアップグレードするかどうかを決定する前に、アップグレードされたインスタンスのコピーで試験することができます。

Amazon RDS での Microsoft SQL Server の機能

Amazon RDS でサポートされる SQL Server バージョンには、以下の機能が含まれます。一般に、Microsoft のドキュメントで特に明記されていない限り、バージョンには以前のバージョンの機能も含まれます。

トピック

- Microsoft SQL Server 2025 の機能

- Microsoft SQL Server 2022 の機能
- Microsoft SQL Server 2019 の機能
- Microsoft SQL Server 2017 の機能
- Microsoft SQL Server 2016 の機能
- Microsoft SQL Server 2014 が Amazon RDS でのサポートを終了
- Microsoft SQL Server 2012 が Amazon RDS でのサポートを終了
- Microsoft SQL Server 2008 R2 が Amazon RDS でのサポートを終了
- Microsoft SQL Server DB インスタンスの変更データキャプチャのサポート
- サポート対象外の機能とサポートが制限されている機能

Microsoft SQL Server 2025 の機能

SQL Server 2025 には以下のような多数の新機能があります。

- 外部 REST エンドポイントの呼び出し (sp_invoke_external_rest_endpoint) – T-SQL コードが追加のミドルウェアなしでデータベースエンジンから直接 AWS のサービスを呼び出すことを可能にする新しいシステムストアドプロシージャ。これを使用して、AWS Lambda や Amazon Bedrock など AWS のサービスを呼び出し、外部 REST API を呼び出し、AWS インフラストラクチャを使用してイベント駆動型ワークフローを構築します。DB インスタンスパラメータグループで sp_invoke_external_rest_endpoint を有効にできます。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
- AI とベクトルのサポート – ベクトル埋め込みを保存およびクエリするためのネイティブベクトルデータ型、ベクトル関数 (VECTOR_DISTANCE、VECTOR_NORM、VECTOR_NORMALIZE)、および REST 推論エンドポイントの外部 AI モデル管理。
- 正規表現 – パターンマッチングとテキスト操作のための新しい T-SQL 関数 (REGEXP_LIKE、REGEXP_REPLACE、REGEXP_SUBSTR、REGEXP_INSTR、REGEXP_COUNT、REGEXP_M
- JSON ネイティブバイナリ形式 – JSON データは最適化されたバイナリ形式で保存され、JSON を解析、読み取り、変更、フォーマットするための関数が組み込まれています。
- 最適化ロック – ブロックとロックのメモリ消費量を減らし、ロックのエスカレーションを回避します。
- Standard Edition 容量の増加 – SQL Server 2025 では、最大コンピューティング容量が 4 ソケットまたは 32 コアに、最大バッファプールメモリが 256 GB に増えます。
- Standard Edition のリソースガバナー – ワークロードあたりの TempDB スペース使用量を管理する機能など、Enterprise Edition と同じ機能を備えた Standard Edition でリソースガバナーが利用

可能になりました。詳細については、Microsoft ドキュメントの「リソースガバナー」を参照してください。

- SSRS を Power BI Report Server に置き換え – レポートサービスは Power BI Report Server (PBIRS) に統合されます。詳細については、「Amazon RDS for SQL Server での SQL Server Reporting Services と Power BI Report Server のサポート」を参照してください。
- Developer エディション – 以前のバージョンの単一の Developer Edition が、新しい個別の Developer EE Edition と Developer SE Edition に置き換わります。サポートされているインスタンスクラスについては、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。

SQL Server 2025 のすべての機能のリストについては、Microsoft のドキュメントの「SQL Server 2025 (17.x) の新機能」を参照してください。

サポートされない機能の一覧については、「サポート対象外の機能とサポートが制限されている機能」を参照してください。

Microsoft SQL Server 2022 の機能

SQL Server 2022 には以下のような多数の新機能があります。

- パラメータに依存するプランの最適化 — 1 つのパラメータ化されたステートメントに対して複数のキャッシュされたプランを使用できるため、パラメータのスニッフィングの問題が軽減される可能性があります。
- SQL Server Ledger – データが承認なしで変更されていないことを暗号的に証明する機能を提供します。
- トランザクションログファイルの拡張イベントのファイルの瞬時初期化 – TDE が有効になっているデータベースの場合を含め、最大 64MB までのログ拡張イベントをより速く実行できます。
- システムページラッチのコンカレンシーの機能強化 — データページとエクステントの割り当てと割り当て解除中のページのラッチの競合が軽減され、tempdb の負荷の高いワークロードのパフォーマンスが大幅に強化されます。

SQL Server 2022 のすべての機能のリストについては、Microsoft のドキュメントの「SQL Server 2022 (16.x) の新機能」を参照してください。

サポートされない機能の一覧については、「サポート対象外の機能とサポートが制限されている機能」を参照してください。

Microsoft SQL Server 2019 の機能

SQL Server 2019 には以下のような多数の新機能があります。

- 高速データベース復旧 (ADR) – 再起動後または長時間実行されているトランザクションロールバック後のクラッシュ復旧時間が短くなります。
- インテリジェントクエリ処理 (IQP):
  - 行モードメモリ許可フィードバック – 過剰な許可が自動的に修正されます。これにより、無駄なメモリ使用や同時実行が減ります。
  - 行ストアバッチモード – 列ストアインデックスを必要とせずに、分析ワークロードのバッチモード実行が可能になります。
  - テーブル変数の遅延コンパイル – テーブル変数を参照するクエリのプラン品質と全体的なパフォーマンスが向上します。
- インテリジェントなパフォーマンス:
  - `OPTIMIZE_FOR_SEQUENTIAL_KEY` インデックスオプション – インデックスへの同時実行挿入のスループットが向上します。
  - 間接チェックポイントのスケーラビリティの向上 – DML ワークロードが高いデータベースに役立ちます。
  - Concurrent Page Free Space (PFS) の更新 – 排他的ラッチではなく共有ラッチとしての処理が可能になります。
- モニタリングの改善点:
  - `WAIT_ON_SYNC_STATISTICS_REFRESH` 待機タイプ – 同期統計更新オペレーションにかかったインスタンスレベルの累積時間を示します。
  - データベーススコープの設定 – `LIGHTWEIGHT_QUERY_PROFILING` と `LAST_QUERY_PLAN_STATS` が含まれます。
  - 動的管理機能 (DMF) – `sys.dm_exec_query_plan_stats` と `sys.dm_db_page_info` が含まれます。
- 詳細な切り捨て警告 – データの切り捨てエラーメッセージに、デフォルトでテーブル名と列名、切り捨てられた値が含まれます。
- 再開可能なオンラインインデックスの作成 – SQL Server 2017 では、再開可能なオンラインインデックスの再構築のみがサポートされています。

SQL Server 2019 のすべての機能のリストについては、Microsoft のドキュメントの「SQL Server 2019 (15.x) の新機能」を参照してください。

サポートされない機能の一覧については、「サポート対象外の機能とサポートが制限されている機能」を参照してください。

Microsoft SQL Server 2017 の機能

SQL Server 2017 には次のような多数の新機能があります。

- 適応型クエリ処理
- 自動計画修正 (自動チューニング機能)
- GraphDB
- 再開可能なインデックスの再構築

SQL Server 2017 のすべての機能のリストについては、Microsoft のドキュメントの「SQL Server 2017 の新機能」を参照してください。

サポートされない機能の一覧については、「サポート対象外の機能とサポートが制限されている機能」を参照してください。

Microsoft SQL Server 2016 の機能

Amazon RDS は、SQL Server 2016 の以下の機能をサポートしています。

- 常時暗号化
- JSON サポート
- 運用の分析
- クエリの保存
- 一時テーブル

SQL Server 2016 の機能の詳細なリストについては、Microsoft ドキュメントの「SQL Server 2016 の新機能」を参照してください。

Microsoft SQL Server 2014 が Amazon RDS でのサポートを終了

SQL Server 2014 は Amazon RDS でのサポートが終了しました。

RDS では、SQL Server 2014 を使用している既存のインスタンスはすべて、最新のマイナーバージョンの SQL Server 2016 にアップグレードされます。詳細については、「Amazon RDS でのバージョン管理」を参照してください。

Microsoft SQL Server 2012 が Amazon RDS でのサポートを終了

SQL Server 2012 は Amazon RDS でのサポートが終了しました。

RDS では、SQL Server 2012 を使用している既存のインスタンスはすべて、最新のマイナーバージョンの SQL Server 2016 にアップグレードされます。詳細については、「Amazon RDS でのバージョン管理」を参照してください。

Microsoft SQL Server 2008 R2 が Amazon RDS でのサポートを終了

SQL Server 2008 R2 は Amazon RDS でのサポートが終了しました。

RDS では、SQL Server 2008 R2 を使用している既存のインスタンスはすべて、最新のマイナーバージョンの SQL Server 2012 にアップグレードされます。詳細については、「Amazon RDS でのバージョン管理」を参照してください。

Microsoft SQL Server DB インスタンスの変更データキャプチャのサポート

Amazon RDS は、Microsoft SQL Server で実行されている DB インスタンスの変更データキャプチャをサポートします。CDC は、テーブル内のデータに加えられた変更を取得し、後からアクセスできる各変更に関するメタデータを格納します。詳細については、Microsoft ドキュメントの「変更データキャプチャ」を参照してください。

Amazon RDS は、次の SQL Server のエディションおよびバージョンの CDC をサポートしています。

- Microsoft SQL Server Enterprise Edition (すべてのバージョン)
- Microsoft SQL Server Standard Edition:
  - 2022
  - 2019
  - 2017 年
  - 2016 バージョン 13.00.4422.0 SP1 CU2 以降

Amazon RDS DB インスタンスで CDC を使用するには、まず RDS 提供のストアドプロシージャを使用してデータベースレベルで CDC を有効または無効にします。その後で、そのデータベースの `db_owner` ロールを持つユーザーは、ネイティブの Microsoft ストアドプロシージャを使用して、そのデータベースの CDC を制御できます。詳細については、「Amazon RDS for SQL Server の変更データキャプチャの使用」を参照してください。

CDC および AWS Database Migration Service を使用すると、SQL Server DB インスタンスからの継続的なレプリケーションを有効にできます。

サポート対象外の機能とサポートが制限されている機能

次の Microsoft SQL Server 機能は、Amazon RDS でサポートされていません。

- Azure Arc 対応機能
- Microsoft Azure Blob ストレージへのバックアップ
- バッファプールの拡張
- 変更イベントのストリーミング
- カスタムパスワードポリシー
- Data Quality Services
- データベースのログ配布
- データベーススナップショット (Amazon RDS は DB インスタンスナップショットのみをサポートします)
- 拡張ストアドプロシージャ (xp_cmdshell を含む)
- FILESTREAM のサポート
- ファイルテーブル
- 機械学習と R サービス (インストールするための OS アクセス許可が必要です)
- メンテナンスプラン
- パフォーマンスデータコレクター
- ポリシーベースの管理
- PolyBase
- レプリケーション
- サーバーレベルのトリガー
- サービスブローカーエンドポイント
- Stretch database
- 信頼できるデータベースプロパティ (sysadmin ロールが必要)
- T-SQL エンドポイント (CREATE ENDPOINT を使用するオペレーションはいずれも使用できません)
- WCF Data Services

以下の Microsoft SQL Server 機能は、Amazon RDS でのサポートが制限されています。

- 分散クエリ/リンクサーバー。詳細については、Amazon RDS for Microsoft SQL Server を使用してリンクされたサーバーを実装するを参照してください。
- 共通ランタイム言語 (CLR)。RDS for SQL Server 2016 以前のバージョンでは、CLR はSAFE モードでサポートされ、アセンブリビットだけを使用しています。CLR は、RDS for SQL Server 2017 以降のバージョンでサポートされていません。詳細については、Microsoft のドキュメントの「Common Runtime Language Integration」を参照してください。
- Amazon RDS for SQL Server での Oracle OLEDB によるリンクされたサーバー 詳細については、「Amazon RDS for SQL Server での Oracle OLEDB によるリンクされたサーバーのサポート」を参照してください。

次の機能は、Amazon RDS for SQL Server 2022 でサポートされていません。

- スナップショット作成のためのデータベースの一時停止
- 外部データソース
- S3 互換オブジェクトストレージへのバックアップと復元
- オブジェクトストアの統合
- TLS 1.3 および MS-TDS 8.0
- QAT によるバックアップ圧縮オフロード
- SQL Server Analysis Services (SSAS)
- マルチ AZ 配置によるデータベースミラーリング。SQL Server Always On は、マルチ AZ 配置でサポートされている唯一の方法です。

次の機能は、Amazon RDS for SQL Server 2025 でサポートされていません。さらに、SQL Server 2022 でサポートされていないとして表示されているすべての機能は、SQL Server 2025 にも適用されます。

- ZSTD バックアップの圧縮
- SQL Server Integration Services (SSIS)

Microsoft SQL Server のデータベースミラーリングまたは Always On 可用性グループを使用したマルチ AZ 配置

Amazon RDS は、Microsoft SQL Server を実行する DB インスタンスで SQL Server データベースミラーリング (DBM) または Always On 可用性グループ (AG) によるマルチ AZ 配置をサポートしています。マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。予定されたデータベースメンテナンスまたは予期しないサービス障害時に、Amazon RDS は自動的に最新のセカンダリレプリカにフェイルオーバーするため、データベースオペレーションを手動の介入なしで速やかに再開できます。プライマリインスタンスおよびセカンダリインスタンスは、同じエンドポイントを使用します。このエンドポイントの物理的なネットワークアドレスは、フェイルオーバープロセスの一環としてパッシブなセカンダリレプリカに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。

Amazon RDS は、アクティブにマルチ AZ をモニタリングして、プライマリで問題が発生したときにフェイルオーバーを開始することで、フェイルオーバーを管理します。スタンバイとプライマリが完全に同期しない限り、フェイルオーバーが開始することはありません。Amazon RDS は、異常のある DB インスタンスを自動的に修正し、同期レプリケーションを再確立することで、マルチ AZ デプロイをアクティブに維持します。何も管理する必要はありません。Amazon RDS がプライマリインスタンス、監視インスタンス、およびスタンバイインスタンスを処理します。SQL Server マルチ AZ をセットアップすると、RDS はインスタンスのすべてのデータベースに対してパッシブなセカンダリインスタンスを設定します。

詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。

Transparent Data Encryption を使用した保管時のデータの暗号化

Amazon RDS は、格納されているデータを透過的に暗号化する Microsoft SQL Server の透過的なデータ暗号化 (TDE) をサポートしています。Amazon RDS は、オプショングループを使用してこれらの機能の有効化と設定を行います。TDE オプションの詳細については、「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。

Amazon RDS for Microsoft SQL Server 用の関数とストアドプロシージャ

次の表に、SQL Server タスクの自動化に役立つ Amazon RDS 関数とストアドプロシージャの一覧を示します。

タスクタイプ	プロシージャまたは関数	使用場所
管理タスク	rds_drop_database	Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースの削除
	rds_failover_time	Amazon RDS for SQL Server の最終フェイルオーバー時間の決定
	rds_manage_view_db_permission	Amazon RDS for SQL Server のデータベース名の表示を拒否または許可する
	rds_modify_db_name	マルチ AZ 配置の Amazon RDS for Microsoft SQL Server データベースの名前の変更
	rds_read_error_log	エラーログとエージェントログの表示
	rds_set_configuration	このオペレーションは、さまざまな DB インスタンスの設定を設定するために使用されます。 • マルチ AZ インスタンスの変更データキャプチャ • トレースファイルおよびダンプファイルの保持期間を設定する • バックアップファイルの圧縮
	rds_set_database_online	Amazon RDS for SQL Server データベースのオフラインからオンラインへの切り替え
	rds_set_system_database_syn	SQL Server エージェントジョブレプリケーションをオンにする

タスクタイプ	プロシージャまたは関数	使用場所
	c_objects rds_fn_get_system_database_sync_objects rds_fn_server_object_last_sync_time	
	rds_show_configuration	rds_set_configuration を使用して設定された値を表示するには、これらのトピックを参照してください。 • マルチ AZ インスタンスの変更データキャプチャ • トレースファイルおよびダンプファイルの保持期間を設定する
	rds_shrink_tempdbfile	tempdb データベースの圧縮
変更データキャプチャ (CDC)	rds_cdc_disable_db	CDC の無効化
	rds_cdc_enable_db	CDC の有効化

タスクタイプ	プロシージャまたは関数	使用場所
データベースメール	rds_fn_sysmail_allitems	メッセージ、ログ、添付ファイルの表示
	rds_fn_sysmail_event_log	メッセージ、ログ、添付ファイルの表示
	rds_fn_sysmail_mailattachments	メッセージ、ログ、添付ファイルの表示
	rds_sysmail_control	このオペレーションは、メールキューの開始と停止に使用されます。 • メールキューの開始 • メールキューの停止
	rds_sysmail_delete_mailitems_sp	メッセージの削除
ネイティブバックアップおよび復元	rds_backup_database	データベースのバックアップ
	rds_cancel_task	タスクのキャンセル
	rds_finish_restore	データベースの復元を終了する

タスクタイプ	プロシージャまたは関数	使用場所
	`rds_restore_database`	データベースの復元
	`rds_restore_log`	ログの復元
Amazon S3 ファイル転送	`rds_delete_from_filesystem`	RDS DB インスタンスでのファイル削除
	`rds_download_from_s3`	Amazon S3 バケットから SQL Server DB インスタンスにファイルをダウンロードする
	`rds_gather_file_details`	RDS DB インスタンスでのファイル一覧表示
	`rds_upload_to_s3`	SQL Server DB インスタンスから Amazon S3 バケットにファイルをアップロードする
Microsoft 分散トランザクションコーディネーター (MSDTC)	`rds_msdtc_transaction_tracing`	トランザクションの追跡の使用
SQL Server Audit	`rds_fn_get_audit_file`	監査ログの表示

タスクタイプ	プロシージャまたは関数	使用場所

透過的なデータ暗号化	`rds_backup_tde_certificate`
	
	`rds_drop_tde_certificate`
	
	`rds_restore_tde_certificate`
	
	`rds_fn_list_user_tde_certificates`
		SQL サーバーの透過的なデータの暗号化サポート

タスクタイプ	プロシージャまたは関数	使用場所
マイクロソフトビジネスインテリジェンス (MSBI)	`rds_msbi_task`	このオペレーションは SQL Server Analysis Services (SSAS) で使用されます。 • Amazon RDS への SSAS プロジェクトのデプロイ • データベース管理者としてのドメインユーザーの追加 • SSAS データベースのバックアップ • SSAS データベースの復元 このオペレーションは SQL Server Integration Services (SSIS) でも使用されます。 • SSISDB の管理権限 • SSIS プロジェクトのデプロイ SQL Server Reporting Services (SSRS) および Power BI Report Server (PBIRS) もこのオペレーションを使用します。 • ドメインユーザーへのアクセスの許可 • システムレベルのアクセス許可の取り消し
	`rds_fn_task_status`	このオペレーションは MSBI タスクのステータスを表示します。 • SSAS: デプロイタスクのステータスのモニタリング • SSIS: デプロイタスクのステータスのモニタリング • SSRS: タスクのステータスのモニタリング
SSIS	`rds_drop_ssis_database`	SSISDB データベースの削除

タスクタイプ	プロシージャまたは関数	使用場所
	rds_sqlagent_proxy	SSIS プロキシの作成
SSRS および PBIRS	rds_drop_ssrs_databases	SSRS または PBIRS データベースの削除
リソースガバナー	rds_create_resource_pool	リソースプールの作成
	rds_alter_resource_pool	リソースプールの変更
	rds_drop_resource_pool	リソースプールの削除
	rds_create_workload_group	ワークロードグループを作成する
	rds_alter_workload_group	ワークロードグループの変更
	rds_drop_workload_group	ワークロードグループの削除

タスクタイプ	プロシージャまたは関数	使用場所
	rds_create_classifier_function	分類子関数を作成して登録する
	rds_drop_classifier_function	分類子関数の削除
	rds_alter_resource_governor_configuration	リソースガバナーの設定変更
	rds_bind_tempdb_metadata_to_resource_pool	TempDB をリソースプールにバインドする
	rds_unbind_tempdb_metadata_from_resource_pool	リソースプールから TempDB をバインド解除する
	rds_cleanup_resource_governor	リソースガバナーのクリーンアップ

Microsoft SQL Server DB インスタンスのローカルタイムゾーン

Microsoft SQL Server を実行している Amazon RDS DB インスタンスのタイムゾーンは、デフォルトに設定されています。現在のデフォルトは協定世界時 (UTC) です。DB インスタンスのタイムゾーンをローカルタイムゾーンに設定して、アプリケーションのタイムゾーンと一致させることも可能です。

DB インスタンスを最初に作成するときにタイムゾーンを設定します。AWS マネジメントコンソール、Amazon RDS API の CreateDBInstance アクション、または AWS CLI の create-db-instance コマンドを使用して、DB インスタンスを作成できます。

DB インスタンスがマルチ AZ 配置 (SQL Server の DBM または AG を使用) の一部である場合にフェイルオーバーが行われても、タイムゾーンは設定したローカルタイムゾーンを維持します。詳細については、「Microsoft SQL Server のデータベースミラーリングまたは Always On 可用性グループを使用したマルチ AZ 配置」を参照してください。

特定の時点への復元をリクエストする場合には、復元を行う時間を指定します。時間は、ローカルタイムゾーンで表示されます。詳細については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

DB インスタンスにローカルタイムゾーンを設定する際の制限事項を以下に示します。

- SQL Server の既存の DB インスタンスのタイムゾーンを変更することはできません。
- あるタイムゾーンの DB インスタンスのスナップショットを、タイムゾーンの異なる DB インスタンスに復元することはできません。
- あるタイムゾーンのバックアップファイルを、別のタイムゾーンに復元しないことを強くお勧めします。バックアップファイルを別のタイムゾーンに復元した場合は、タイムゾーンの変更によるクエリとアプリケーションへの影響を精査する必要があります。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

サポートされているタイムゾーン

ローカルタイムゾーンは以下の表に示されているいずれかの値に設定できます。

タイムゾーン	標準の時間オフセット	説明	コメント
アフガニスタン標準時	(UTC+04:30)	カブール	このタイムゾーンは夏時間を考慮しません。
アラスカ標準時	(UTC–09:00)	アラスカ	
アリューシャン標準時	(UTC–10:00)	アリューシャン列島	
アルタイ標準時	(UTC+07:00)	バルナウル (ゴルノ＝アルタイスク)	
アラブ標準時	(UTC+03:00)	クウェート (リヤド)	このタイムゾーンは夏時間を考慮しません。
アラビア標準時	(UTC+04:00)	アブダビ、マスカット	
アラビア標準時	(UTC+03:00)	バグダッド	このタイムゾーンは夏時間を考慮しません。
アルゼンチン標準時	(UTC–03:00)	ブエノスアイレス市	このタイムゾーンは夏時間を考慮しません。
アストラハン標準時	(UTC+04:00)	アストラハン (ウリヤノフスク州)	
大西洋標準時	(UTC–04:00)	大西洋標準時 (カナダ)	
中部オーストラリア標準時	(UTC+09:30)	ダーウィン	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
オーストラリア中西部標準時	(UTC+08:45)	ユークラ	
東部オーストラリア標準時	(UTC+10:00)	キャンベラ、メルボルン、シドニー	
アゼルバイジャン標準時	(UTC+04:00)	バクー	
アゾレス諸島標準時	(UTC–01:00)	アゾレス諸島	
バイア州標準時	(UTC–03:00)	サルバドル	
バングラデシュ標準時	(UTC+06:00)	ダッカ	このタイムゾーンは夏時間を考慮しません。
ベラルーシ標準時	(UTC+03:00)	ミンスク	このタイムゾーンは夏時間を考慮しません。
ブーゲンビル標準時	(UTC+11:00)	ブーゲンビル島	
中部カナダ標準時	(UTC–06:00)	サスカチュワン	このタイムゾーンは夏時間を考慮しません。
カーボベルデ標準時	(UTC–01:00)	カーボベルデ島	このタイムゾーンは夏時間を考慮しません。
コーカサス標準時	(UTC+04:00)	エレバン	
中部 オーストラリア標準時	(UTC+09:30)	アデレード	
アメリカ中部標準時	(UTC–06:00)	中米	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
中央アジア標準時	(UTC+06:00)	アスターナ	このタイムゾーンは夏時間を考慮しません。
中部ブラジル標準時	(UTC–04:00)	クヤバ	
中央ヨーロッパ標準時	(UTC+01:00)	ベオグラード、ブラティスラヴァ、ブダペスト、リュブヤナ、プラハ	
中央ヨーロッパ標準時	(UTC+01:00)	サラエボ、スコピエ、ワルシャワ、ザグレブ	
中央太平洋標準時	(UTC+11:00)	ソロモン諸島、ニューカレドニア	このタイムゾーンは夏時間を考慮しません。
中部標準時	(UTC–06:00)	中部標準時 (米国およびカナダ)	
中部標準時 (メキシコ)	(UTC–06:00)	グアダラハラ、メキシコシティ、モンテレイ	
チャタム諸島標準時	(UTC+12:45)	チャタム諸島	
中国標準時	(UTC+08:00)	北京、重慶、香港特別行政区、ウルムチ	このタイムゾーンは夏時間を考慮しません。
キューバ標準時	(UTC–05:00)	ハバナ	

タイムゾーン	標準の時間オフセット	説明	コメント
日付変更線標準時	(UTC–12:00)	国際日付変更線西	このタイムゾーンは夏時間を考慮しません。
アフリカ東部標準時	(UTC+03:00)	ナイロビ	このタイムゾーンは夏時間を考慮しません。
オーストラリア東部標準時	(UTC+10:00)	ブリスベン	このタイムゾーンは夏時間を考慮しません。
欧州東部標準時	(UTC+02:00)	キシナウ	
南米東部標準時	(UTC–03:00)	ブラジリア	
イースター島標準時	(UTC–06:00)	イースター島	
東部標準時	(UTC–05:00)	東部標準時 (米国およびカナダ)	
東部標準時 (メキシコ)	(UTC–05:00)	チェトゥマル	
エジプト標準時	(UTC+02:00)	カイロ	
エカテリンブルク標準時	(UTC+05:00)	エカテリンブルク	
フィジー標準時	(UTC+12:00)	フィジー	
FLE 標準時	(UTC+02:00)	ヘルシンキ、キエフ、リガ、ソフィア、タリン、ビリニュス	
ジョージア標準時	(UTC+04:00)	トビリシ	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
GMT 標準時	(UTC)	ダブリン エジンバラ、リスボン、ロンドン	このタイムゾーンはグリニッジ標準時と同じではありません。このタイムゾーンは夏時間を考慮します。
グリーンランド標準時	(UTC–03:00)	グリーンランド	
グリニッジ標準時	(UTC)	モンロビア、レイキャビク	このタイムゾーンは夏時間を考慮しません。
GTB 標準時	(UTC+02:00)	アテネ、ブカレスト	
ハイチ標準時	(UTC–05:00)	ハイチ	
ハワイ標準時	(UTC–10:00)	ハワイ	
インド標準時	(UTC+05:30)	チェンナイ、カルカッタ、ムンバイ、ニューデリー	このタイムゾーンは夏時間を考慮しません。
イラン標準時	(UTC+03:30)	テヘラン	
イスラエル標準時	(UTC+02:00)	エルサレム	
ヨルダン標準時	(UTC+02:00)	アンマン	
カリーニングラード標準時	(UTC+02:00)	カリーニングラード	
カムチャツカ標準時	(UTC+12:00)	ペトロパブロフスク・カムチャツキー – オールド	

タイムゾーン	標準の時間オフセット	説明	コメント
韓国標準時	(UTC+09:00)	ソウル	このタイムゾーンは夏時間を考慮しません。
リビア標準時	(UTC+02:00)	トリポリ	
ライン諸島標準時	(UTC+14:00)	キリティマティ島	
ロード・ハウ標準時	(UTC+10:30)	ロード・ハウ島	
マガダン標準時	(UTC+11:00)	マガダン	このタイムゾーンは夏時間を考慮しません。
マガジャネス標準時	(UTC–03:00)	プンタ・アレーナス	
マルケサス標準時	(UTC–09:30)	マルケサス諸島	
モーリシャス標準時	(UTC+04:00)	ポートルイス	このタイムゾーンは夏時間を考慮しません。
中東標準時	(UTC+02:00)	ベイルート	
モンテビデオ標準時	(UTC–03:00)	モンテビデオ	
モロッコ標準時	(UTC+01:00)	カサブランカ	
山岳部標準時	(UTC–07:00)	山地標準時 (米国およびカナダ)	
山岳部標準時 (メキシコ)	(UTC–07:00)	チワワ、ラパス、マサトラン	
ミャンマー標準時	(UTC+06:30)	ヤンゴン (ラングーン)	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
中央アジア北部標準時	(UTC+07:00)	ノヴォシビルスク	
ナミビア標準時	(UTC+02:00)	ウィントフック	
ネパール標準時	(UTC+05:45)	カトマンズ	このタイムゾーンは夏時間を考慮しません。
ニュージーランド標準時	(UTC+12:00)	オークランド、ウェリントン	
ニューファンドランド標準時	(UTC–03:30)	ニューファンドランド	
ノーフォーク標準時	(UTC+11:00)	ノーフォーク島	
北アジア東部標準時	(UTC+08:00)	イルクーツク	
北アジア標準時	(UTC+07:00)	クラスノヤルスク	
北朝鮮標準時	(UTC+09:00)	平壌	
オムスク標準時	(UTC+06:00)	オムスク	
太平洋南アメリカ標準時	(UTC–03:00)	サンティアゴ	
太平洋標準時	(UTC–08:00)	太平洋標準時 (米国およびカナダ)	
太平洋標準時 (メキシコ)	(UTC–08:00)	バハ・カリフォルニア	
パキスタン標準時	(UTC+05:00)	イスラマバード (カラチ)	このタイムゾーンは夏時間を考慮しません。
パラグアイ標準時	(UTC–04:00)	アスンシオン	

タイムゾーン	標準の時間オフセット	説明	コメント
ロマンス標準時	(UTC+01:00)	ブリュッセル、コペンハーゲン、マドリード、パリ	
ロシアタイムゾーン 10	(UTC+11:00)	チョクルダフ	
ロシアタイムゾーン 11	(UTC+12:00)	ペトロパブロフスク・カムチャツキー・アナディル	
ロシアタイムゾーン 3	(UTC+04:00)	イジェフスク	
ロシア標準時	(UTC+03:00)	モスクワ、サンクト・ペテルブルク、ヴォルゴグラード	このタイムゾーンは夏時間を考慮しません。
SA 東部標準時	(UTC–03:00)	フォルタレザカイエンヌ	このタイムゾーンは夏時間を考慮しません。
南アメリカ太平洋標準時	(UTC–05:00)	ボゴタ、リマ、キト、リオブランコ	このタイムゾーンは夏時間を考慮しません。
SA 西部標準時	(UTC–04:00)	ジョージタウン、ラパス、マナウス、サンフアン	このタイムゾーンは夏時間を考慮しません。
サンピエール標準時	(UTC–03:00)	サンピエール・ミクロン	
サハリン標準時	(UTC+11:00)	サハリン	
サモア標準時	(UTC+13:00)	サモア	
サントメ標準時	(UTC+01:00)	サントメ	

タイムゾーン	標準の時間オフセット	説明	コメント
サラトフ標準時	(UTC+04:00)	サラトフ	
東南アジア標準時	(UTC+07:00)	バンコク、ハノイ、ジャカルタ	このタイムゾーンは夏時間を考慮しません。
シンガポール標準時	(UTC+08:00)	クアラルンプール、シンガポール	このタイムゾーンは夏時間を考慮しません。
南アフリカ標準時	(UTC+02:00)	ハラレ (プレトリア)	このタイムゾーンは夏時間を考慮しません。
スリランカ標準時	(UTC+05:30)	スリ・ジャヤワルデネプラ	このタイムゾーンは夏時間を考慮しません。
スーダン標準時	(UTC+02:00)	ハルツーム	
シリア標準時	(UTC+02:00)	ダマスカス	
台北標準時	(UTC+08:00)	台北	このタイムゾーンは夏時間を考慮しません。
タスマニア標準時	(UTC+10:00)	ホバート	
トカンティンス標準時	(UTC–03:00)	アラグアイナ	
日本標準時	(UTC+09:00)	大阪、札幌、東京	このタイムゾーンは夏時間を考慮しません。
トムスク標準時	(UTC+07:00)	トムスク	

タイムゾーン	標準の時間オフセット	説明	コメント
トンガ標準時	(UTC+13:00)	ヌクアロファ	このタイムゾーンは夏時間を考慮しません。
トランスバイカル標準時	(UTC+09:00)	チタ	
トルコ標準時	(UTC+03:00)	イスタンブール	
タークス・カイコス標準時	(UTC–05:00)	タークス・カイコス島	
ウランバートル標準時	(UTC+08:00)	ウランバートル	このタイムゾーンは夏時間を考慮しません。
米国東部標準時	(UTC–05:00)	インディアナ東部	
山地標準時 (米国)	(UTC–07:00)	アリゾナ	このタイムゾーンは夏時間を考慮しません。
UTC	UTC	協定世界時	このタイムゾーンは夏時間を考慮しません。
UTC–02	(UTC–02:00)	協定世界時–02	このタイムゾーンは夏時間を考慮しません。
UTC–08	(UTC–08:00)	協定世界時–08	
UTC–09	(UTC–09:00)	協定世界時–09	
UTC–11	(UTC–11:00)	協定世界時–11	このタイムゾーンは夏時間を考慮しません。

タイムゾーン	標準の時間オフセット	説明	コメント
UTC+12	(UTC+12:00)	協定世界時+12	このタイムゾーンは夏時間を考慮しません。
UTC+13	(UTC+13:00)	協定世界時+13	
ベネズエラ標準時	(UTC–04:00)	カラカス	このタイムゾーンは夏時間を考慮しません。
ウラジオストク標準時	(UTC+10:00)	ウラジオストク	
ヴォルゴグラード標準時	(UTC+04:00)	ヴォルゴグラード	
オーストラリア西部標準時	(UTC+08:00)	パース	このタイムゾーンは夏時間を考慮しません。
中部アフリカ西部標準時	(UTC+01:00)	西部・中部アフリカ	このタイムゾーンは夏時間を考慮しません。
ヨーロッパ西部標準時	(UTC+01:00)	アムステルダム、ベルリン、ベルン、ローマ、ストックホルム、ウィーン	
モンゴル西部標準時	(UTC+07:00)	ホヴド	
西アジア標準時	(UTC+05:00)	アシガバート (タシュケント)	このタイムゾーンは夏時間を考慮しません。
西岸標準時	(UTC+02:00)	ガザ (ヘブロン)	

タイムゾーン	標準の時間オフセット	説明	コメント
西太平洋標準時	(UTC+10:00)	グアム、ポートモレスビー	このタイムゾーンは夏時間を考慮しません。
ヤクーツク標準時	(UTC+09:00)	ヤクーツク	

Amazon RDS での Microsoft SQL Server のライセンス

RDS for SQL Server は、ライセンス込みおよび Bring Your Own Media の 2 つのライセンスモデルをサポートしています。

ライセンス込み

ライセンス込み (LI) モデルでは、別の SQL Server ライセンスを購入する必要はありません。Amazon RDS の料金には、SQL Server ソフトウェアライセンス、基盤となるハードウェアリソース、および Amazon RDS 管理機能が含まれています。

Amazon RDS は、次の SQL Server エディションでライセンス込みをサポートしています。

- Enterprise
- Standard
- Web
- Express

Bring Your Own Media

独自の Microsoft SQL Server ライセンスを使用する場合、Amazon RDS は Bring Your Own Media (BYOM) をサポートしています。BYOM の場合、Amazon RDS では SQL Server ライセンス料金を請求されません。代わりに、ソフトウェアアシュアランスによるライセンスモビリティで既存の SQL Server ライセンスを使用します。

AWS インフラストラクチャ (コンピューティング、ストレージ、I/O、データ転送)、Windows OS ライセンス料金、および有効にした追加の RDS 機能に対して料金が発生します。有効な SQL Server ライセンスを所有し、ソフトウェアアシュアランスおよびライセンスモビリティが有効である必要があります。「AWS ライセンスモビリティ」を参照してください。

Amazon RDS は、以下の Microsoft SQL Server エディションで BYOM をサポートしています。

- Enterprise
- 標準
- 開発者

> **Note**
>
> Developer Edition の BYOM は、開発およびテストのみを目的とした無料エディションであり、本番環境では使用できません。詳細については、以下を参照してください。

BYOM のセットアップおよび使用の詳細については、「RDS for SQL Server の Bring Your Own Media (BYOM)」を参照してください。

開発とテスト

開発シナリオとテストシナリオには、次の 2 つのオプションがあります。

- Express Edition 開発、テスト、本番稼働用以外の多くのワークロードに適した無料エディション。ライセンス込みモデルで使用できます。
- Developer Edition SQL Server Enterprise Edition のすべての機能が含まれていますが、本番稼働用以外の用途にのみライセンスされます。BYOM モデルで使用できます。

BYOM エンジンバージョンを使用して、RDS for SQL Server に SQL Server Developer Edition をダウンロードしてインストールできます。詳細については、「RDS for SQL Server での SQL Server Developer Edition の使用」を参照してください。

SQL Server エディション間の相違点の詳細については、Microsoft ドキュメントの「SQL Server 2019 の各エディションとサポートされている機能」を参照してください。

マルチ AZ 配置

Amazon RDS は、Microsoft SQL Server を実行する DB インスタンスで、SQL Server データベースミラーリング (DBM)、Always On 可用性グループ (AG)、SQL Server Web Edition のブロックレベルレプリケーションによるマルチ AZ 配置をサポートしています。マルチ AZ 配置への追加ライセンスは必要ではありません。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。

SQL Server Web Edition の使用要件

SQL Server Web Edition は、公開されインターネットからアクセス可能なウェブページ、ウェブサイト、ウェブアプリケーション、ウェブサービスをホストするウェブホスティングプロバイダーおよびウェブ付加価値プロバイダー (VAP) 向けに設計されています。これらの使用制限は、AWS のサービス条件のセクション 10.5 に基づく Microsoft のライセンス条項により課されています。

ライセンス終了した DB インスタンスの復元

Amazon RDS は、ライセンス終了した DB インスタンスのスナップショットを作成します。インスタンスがライセンスの問題で終了した場合は、スナップショットから新しい DB インスタンスに復元できます。新しい DB インスタンスは、ライセンス込みモデルを使用します。

詳細については、「Amazon RDS for SQL Server のライセンス終了した DB インスタンスの復元」を参照してください。

SQL Server DB インスタンスへの接続

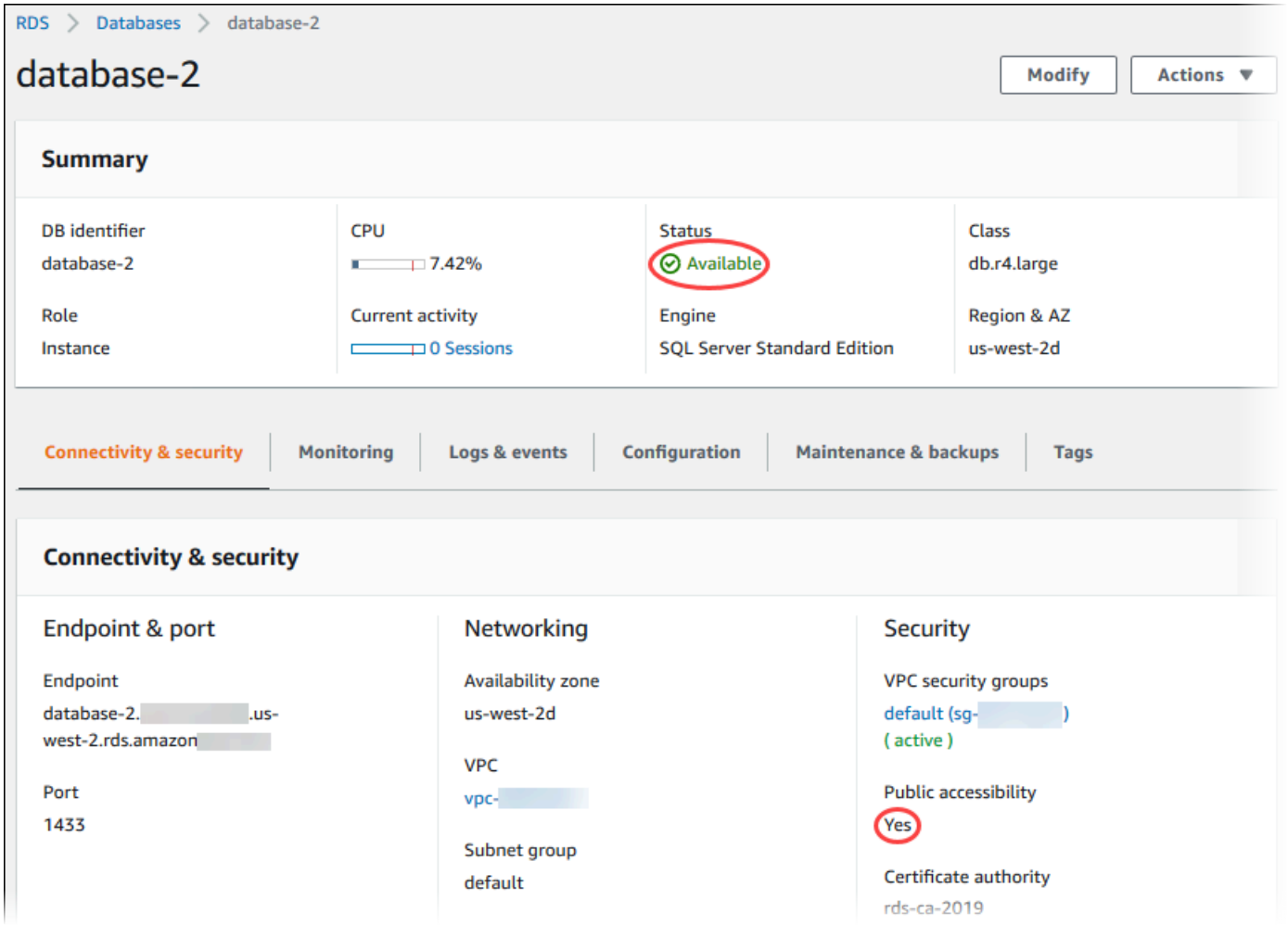
Amazon RDS によって DB インスタンスがプロビジョニングされると、標準の SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。このトピックでは、Microsoft SQL Server Management Studio (SSMS) または SQL Workbench/J を使用して DB インスタンスに接続します。

サンプルの DB インスタンスの作成と接続のプロセスを示す手順の例は、「Microsoft SQL Server DB インスタンスを作成して接続する」を参照してください。

接続する前に

DB インスタンスに接続する前に、そのインスタンスが使用可能かつアクセス可能である必要があります。

1. ステータスが `available` であることを確認してください。これは、AWS マネジメントコンソール のインスタンスの詳細ページで、または describe-db-instances AWS CLI コマンドを使用して確認できます。



2. ソースからアクセスできることを確認してください。シナリオによっては、公開する必要がない場合もあります。詳細については、「Amazon VPC と Amazon RDS」を参照してください。
3. VPC セキュリティグループのインバウンドルールで DB インスタンスへのアクセスが許可されていることを確認してください。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

DB インスタンスのエンドポイントとポート番号の検索

DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。

エンドポイントとポートを検索するには

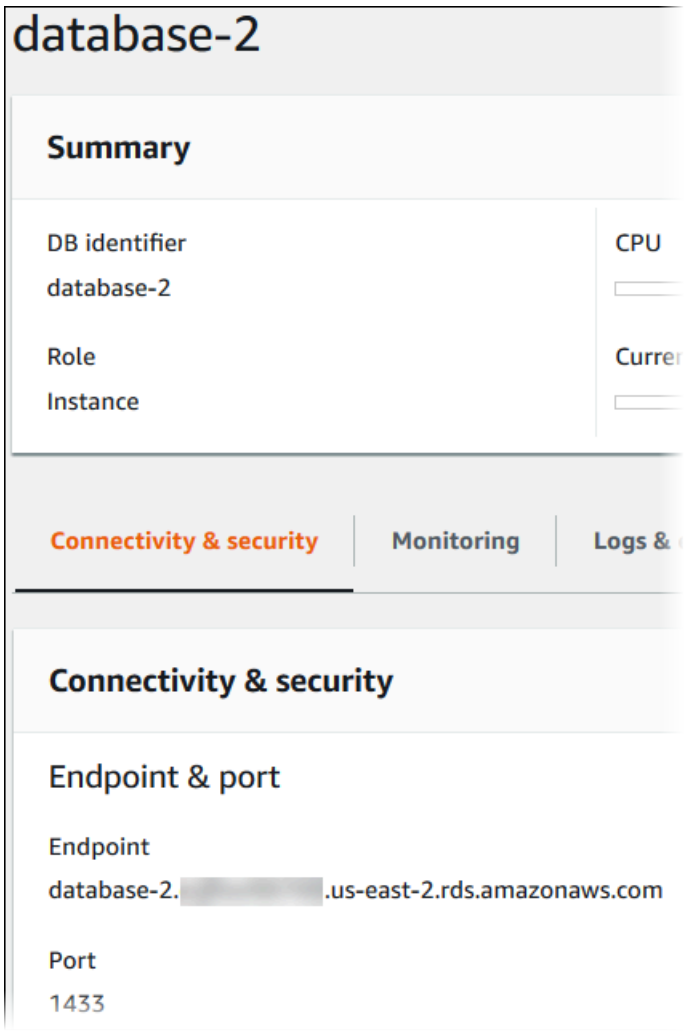
1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスの AWS リージョンを選択します。

3. DB インスタンスのドメインネームシステム (DNS) 名 (エンドポイント) とポート番号を見つけます。

   a. RDS コンソールを開き、[データベース] を選択して、DB インスタンスを一覧表示します。

   b. SQL Server DB インスタンスの名前を選択して詳細を表示します。

   c. [接続とセキュリティ] タブで、エンドポイントをコピーします。



   d. ポート番号を書き留めます。

Microsoft SQL Server Management Studio を使用して DB インスタンスに接続する

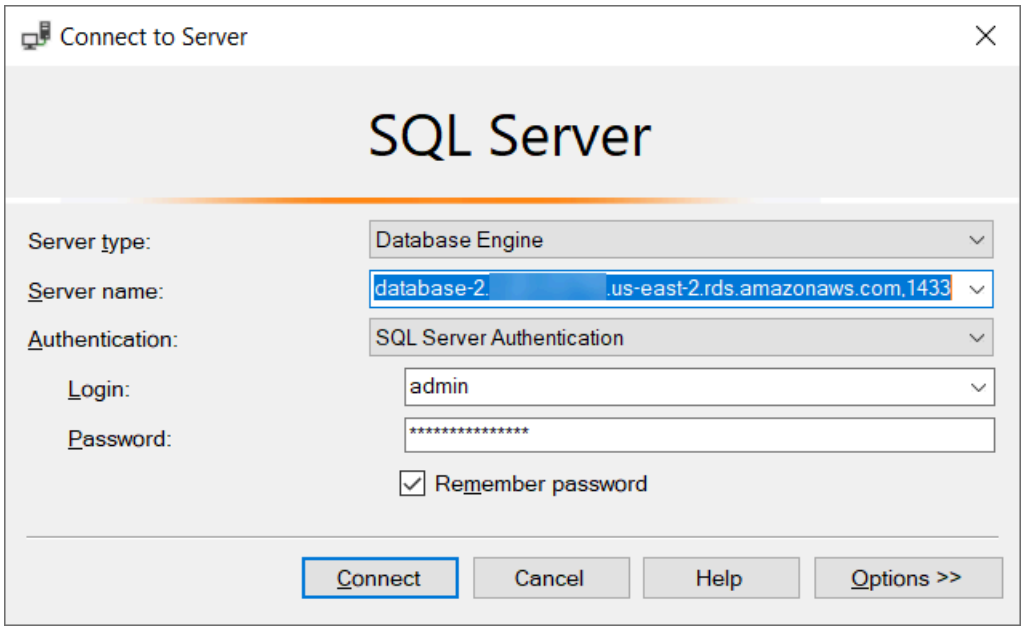
この手順では、Microsoft SQL Server Management Studio (SSMS) を使用してサンプル DB インスタンスに接続します。このユーティリティのスタンドアロンバージョンをダウンロードするに

は、Microsoft ドキュメントの「Download SQL Server Management Studio (SSMS)」を参照してください。

SSMS を使用して DB インスタンスに接続するには

1. SQL Server Management Studio を起動します。

   [サーバーに接続] ダイアログボックスが表示されます。



2. DB インスタンスの情報を指定します。

   a. [サーバーの種類] で、[データベース エンジン] を選択します。

   b. [サーバー名] に、DB インスタンスの DNS 名 (エンドポイント) とポート番号をカンマで区切って入力します。

      > **Important**
      >
      > エンドポイントとポート番号の間にあるコロンをカンマに置き換えます。

      サーバー名は次の例のようになります。

      ```
      database-2.cg034itsfake.us-east-1.rds.amazonaws.com,1433
      ```

   c. [認証] で、[SQL Server 認証] を選択します。

   d. [ログイン] に、DB インスタンスのマスターユーザー名を入力します。

e. [パスワード] に、DB インスタンスのパスワードを入力します。

3. [接続] を選択します。

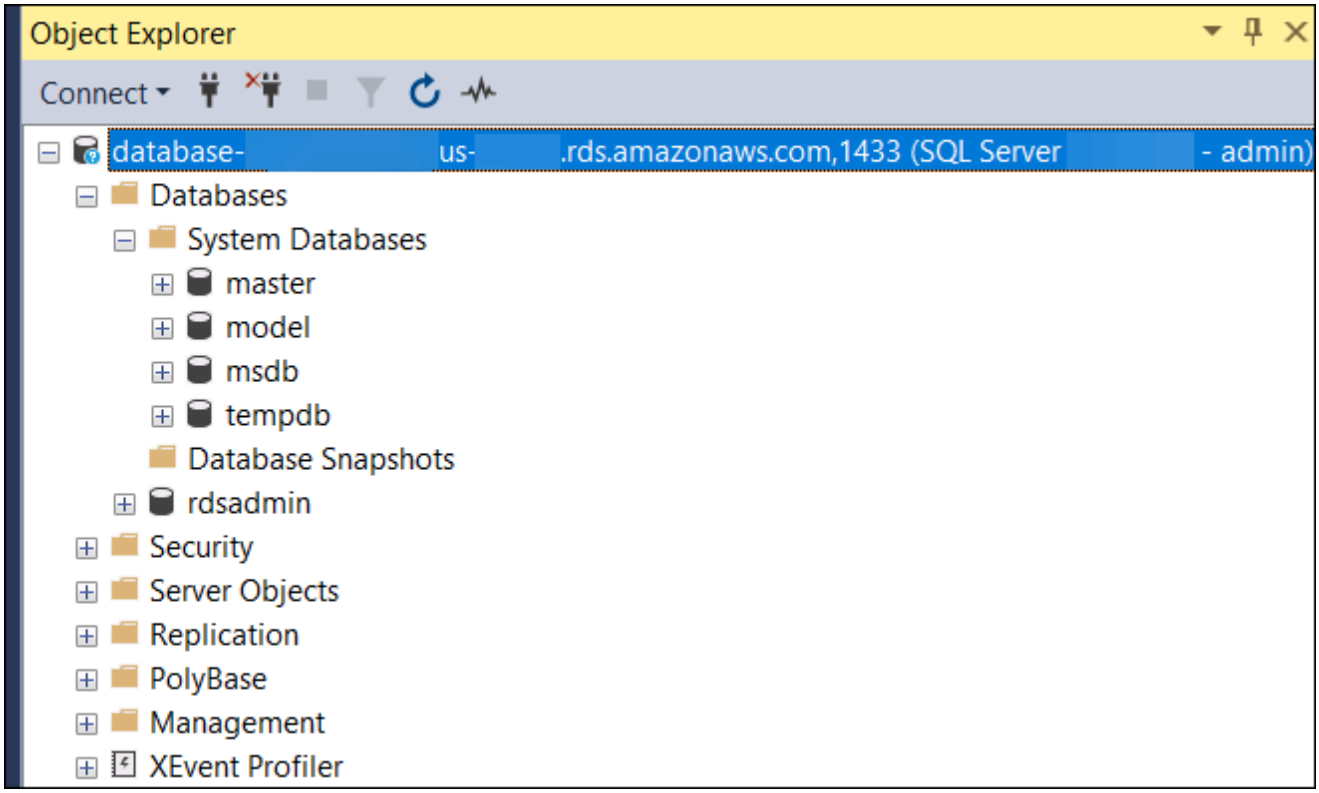
   しばらくすると、SSMS が DB インスタンスに接続されます。

   DB インスタンスに接続できない場合は、「セキュリティグループに関する考慮事項」および「SQL Server DB インスタンスへの接続のトラブルシューティング」を参照してください。

4. SQL Server DB インスタンスには、SQL Server の標準内蔵システムデータベース (`master`、`model`、`msdb`、`tempdb`) が含まれています。システムデータベースを閲覧するには、次を実行します。

   a. SSMS の [ビュー] メニューで、[オブジェクト エクスプローラー] を選択します。

   b. DB インスタンスを展開し、[データベース] を展開します。その後、[システムデータベース] を展開します。



5. SQL Server DB インスタンスには、`rdsadmin` という名前のデータベースも含まれています。Amazon RDS では、このデータベースを使用して、データベースを管理するために使用するオブジェクトを保存します。`rdsadmin` データベースには、詳細なタスクを実行するために保存された手順も含まれます。詳細については、「Amazon RDS for Microsoft SQL Server の一般的な DBA タスク」を参照してください。

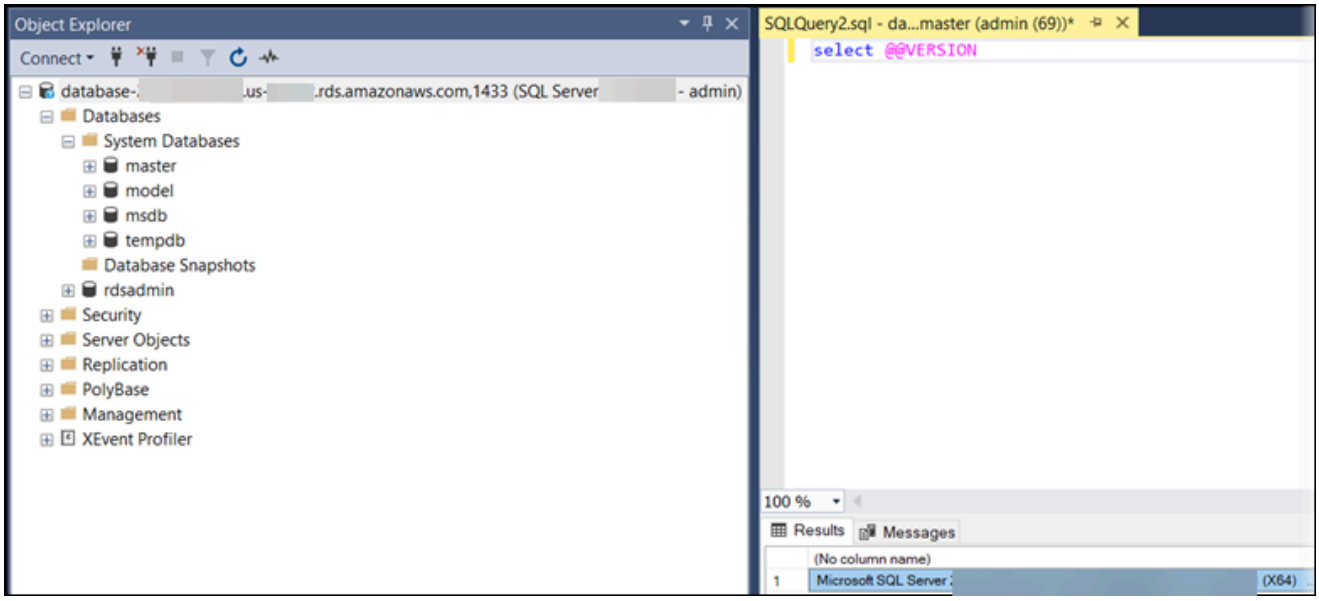
6. 独自のデータベースを作成し、通常のデータベースに加え、DB インスタンスに対しクエリを実行できるようになりました。DB インスタンスに対してテストクエリを実行するには、次を実行します。

   a. SSMS の [ファイル] メニューで [新規] をポイントし、[クエリを現在の接続で実行] を選択します。

   b. 次の SQL クエリを入力します。

   ```
   select @@VERSION
   ```

   c. クエリを実行します。SSMS は、Amazon RDS DB インスタンスの SQL Server のバージョンを返します。



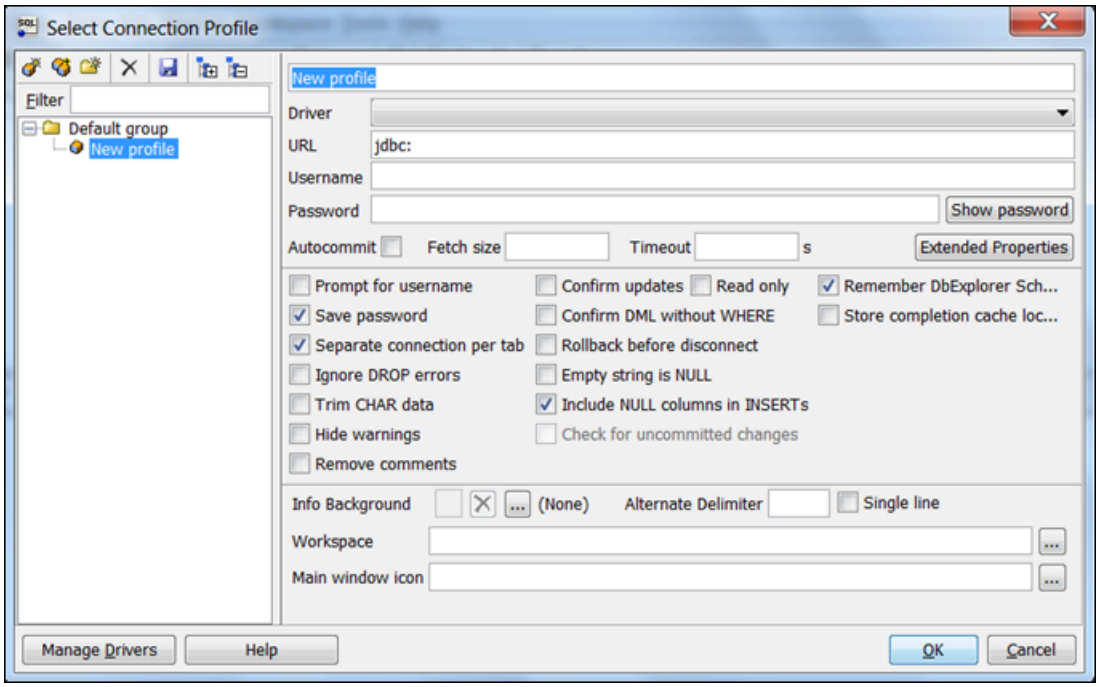
SQL Workbench/J を使用して DB インスタンスに接続する

この例では、SQL Workbench/J データベースツールを使用して Microsoft SQL Server データベースエンジンを実行する DB インスタンスに接続する方法を示します。SQL Workbench/J をダウンロードするには、「SQL Workbench/J」を参照してください。

SQL Workbench/J は DB インスタンスの接続に JDBC を使用します。さらに、SQL Server 用の JDBC ドライバーも必要です。このドライバーをダウンロードするには、「Microsoft SQL Server 用 JDBC Driver のダウンロード」を参照してください。

SQL Workbench/J を使用して DB インスタンスに接続するには

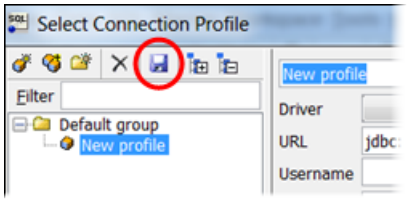
1. 次のように [接続プロファイルの選択] ダイアログボックスを開きます。



2. ダイアログボックスの最初のボックスにプロファイルの名前を入力します。
3. [ドライバー] で、[**SQL JDBC 4.0**] を選択します。
4. [URL] に「**jdbc:sqlserver://**」と入力し、さらに DB インスタンスのエンドポイントを入力します 例えば、URL の値は次のようになります。

```
jdbc:sqlserver://sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com:1433
```

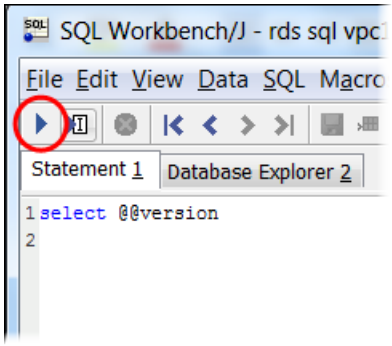
5. [ユーザー名] に、DB インスタンスのマスターユーザー名を入力します。
6. [パスワード] に、マスターユーザーのパスワードを入力します。
7. 以下に示すように、ダイアログのツールバーで保存アイコンを選択します。



8. [OK] を選択します。しばらくすると、SQL Workbench/J が DB インスタンスに接続されます。DB インスタンスに接続できない場合は、「セキュリティグループに関する考慮事項」および「SQL Server DB インスタンスへの接続のトラブルシューティング」を参照してください。
9. クエリペインに、次の SQL クエリを入力します。

```
select @@VERSION
```

10. 以下に示すように、ツールバーで Execute アイコンを選択します。



クエリは、次のような DB インスタンスのバージョン情報を返します。

```
Microsoft SQL Server 2017 (RTM-CU22) (KB4577467) - 14.0.3356.20 (X64)
```

セキュリティグループに関する考慮事項

DB インスタンスに接続するには、DB インスタンスがセキュリティグループに関連付けられている必要があります。このセキュリティグループには、DB インスタンスへのアクセスに使用する IP アドレスとネットワーク設定が含まれています。DB インスタンスを作成する際、DB インスタンスを適切なセキュリティグループと関連付けている可能性があります。DB インスタンスの作成時に、設定されていない、デフォルトのセキュリティグループを割り当てた場合は、DB インスタンスファイアウォールによって接続が禁止されます。

場合によっては、アクセスを可能にするために、新しいセキュリティグループを作成する必要があります。新しいセキュリティグループの作成手順については、「セキュリティグループによるアクセス制御」を参照してください。VPC セキュリティグループにルールを設定するプロセスについて順を追って説明するトピックは、「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。

新しいセキュリティグループを作成したら、そのセキュリティグループと関連付けるように DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

SSL を使用して DB インスタンスへの接続を暗号化することで、セキュリティを高めることができます。詳細については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。

SQL Server DB インスタンスへの接続のトラブルシューティング

次の表は、SQL Server DB インスタンスに接続しようとしたときに発生する可能性があるエラーメッセージを示しています。

問題	トラブルシューティングの推奨事項
SQL Server への接続を開けませんでした – Microsoft SQL Server、エラー: 53	正しいサーバー名を指定していることを確認します。[サーバー名] に、サンプル DB インスタンスの DNS 名とポート番号をカンマで区切って入力します。 **Important** DNS 名とポート番号の間にコロンがある場合は、コロンをコンマに変更します。 サーバー名は次の例のようになります。 `sample-instance.cg034itsfake.us-east-1.rds.amazonaws.com,1433`
対象のコンピュータによって能動的に拒否されたため、接続できません – Microsoft SQL Server、エラー: 10061	DB インスタンスにはリーチできましたが、接続が拒否されました。通常、この問題は、正しくないユーザー名やパスワードが指定されたときに発生します。ユーザー名とパスワードを確認し、再度試します。
SQL Server への接続を確立中に、ネットワーク関連またはインスタンス固有のエラーが発生しました。サーバーが見つからないか、アクセスできませんでした..。待機オペレーションがタイムアウトしました	ローカルファイアウォールによって適用されるアクセスルールと、DB インスタンスへのアクセスが許可された IP アドレスが一致していない可能性があります。問題の原因として最も多いのは、セキュリティグループのインバウンドルールです。詳細については、「Amazon RDS でのセキュリティ」を参照してください。
	データベースインスタンスはパブリックアクセスが可能である必要があります。VPC の外部から接続するには、インスタンスにパブリック IP アドレスが割り当てられている必要があります。

問題	トラブルシューティングの推奨事項
– Microsoft SQL Server、エラー: 258	

> **Note**
>
> 接続の問題の詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

RDS for SQL Server での SQL Server Developer Edition の使用

RDS for SQL Server は SQL Server Developer Edition をサポートとしています。Developer Edition には SQL Server Enterprise Edition のすべての機能が含まれていますが、本番稼働用以外の用途にのみライセンスされます。RDS for SQL Server Developer Edition インスタンスは、カスタムエンジンバージョン (CEV) 機能を使用して独自のインストールメディアを使用して作成できます。Amazon RDS SQL Server は、Standard Edition と Enterprise Edition の Bring Your Own Media (BYOM) もサポートしていますが、機能にはいくつかの違いがあります。詳細については、以下を参照してください。the section called “Developer Edition との比較”

利点

RDS for SQL Server Developer Edition を使用すると、次のことができます。

- 本番環境データベースと同等の機能を維持しながら、開発環境とテスト環境のコストを削減します。
- 非本番環境の Enterprise Edition 機能には、Enterprise ライセンス料金なしでアクセスできます。
- バックアップ、パッチ適用、モニタリングなど、Amazon RDS の自動管理機能を使用します。

> **Note**
>
> SQL Server Developer Edition は、開発およびテストのみを目的としてライセンスされており、本番環境では使用できません。

リージョンの可用性

RDS for SQL Server Developer Edition は、次の AWS リージョンで使用できます。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (台北)

- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (マレーシア)
- アジアパシフィック (メルボルン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ニュージーランド)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (タイ)
- アジアパシフィック (東京)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- イスラエル (テルアビブ)
- メキシコ (中部)
- 南米 (サンパウロ)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

ライセンスと使用

SQL Server Developer Edition は、Microsoft によって開発環境とテスト環境に対してのみライセンスされます。Developer Edition を本番サーバーとして使用することはできません。Amazon RDS で

SQL Server Developer Edition を使用する場合、Microsoft の SQL Server Developer Edition ライセンス条項を遵守する責任があります。AWS インフラストラクチャコストに対してのみお支払いいただきます。追加の SQL Server ライセンス料金はかかりません。料金の詳細については、「RDS for SQL Server の料金」を参照してください。

前提条件

RDS for SQL Server で SQL Server Developer Edition を使用する前に、次の要件が満たされていることを確認してください。

- インストールバイナリは Microsoft から直接取得し、Microsoft のライセンス条項に準拠していることを確認する必要があります。
- Developer Edition DB インスタンスを作成するには、次のリソースを使用するためのアクセス権が必要です。
  - `AmazonRDSFullAccess` および `s3:GetObject` アクセス許可を持つ AWS アカウント。
- インストールメディアを保存するには、Amazon S3 バケットが必要です。CEV 作成の一環として Amazon S3 バケットにアップロードするには、ISO ファイルと累積更新ファイルが必要です。詳細については、「Amazon S3 バケットへのインストールメディアのアップロード」を参照してください。
- すべてのインストールメディアファイルは、カスタムエンジンバージョンが作成される同じリージョン内の同じ Amazon S3 バケット内と、その Amazon S3 バケット内の同じフォルダパス内に存在する必要があります。

サポートバージョン

RDS for SQL Server の Developer Edition は、次のバージョンをサポートしています。

- SQL Server 2025 CU5 (17.00.4045.5) – Developer Edition (Enterprise Edition 機能)
- SQL Server 2025 CU5 (17.00.4045.5) – Developer Edition (Standard Edition 機能)
- SQL Server 2022 CU 21 (16.00.4215.2)
- SQL Server 2019 CU 32 GDR (15.00.4455.2)

> **Note**
>
> SQL Server 2025 以降、Amazon RDS は次の 2 つの Developer Edition エンジンタイプをサポートしています。

- `sqlserver-dev-ee` – Enterprise Edition のすべての機能が含まれていますが、本番稼働用以外の用途にのみライセンスされます。
- `sqlserver-dev-se` – SQL Server 2025 の新機能で、非本番稼働専用にライセンスされたすべての Standard Edition 機能が含まれています。

SQL Server Developer Edition バリアントの詳細については、Microsoft ドキュメントで「SQL Server 2025 の各エディションとサポートされている機能」を参照してください。

Developer Edition (Enterprise Edition 機能) の CEV 作成でサポートされているすべてのエンジンバージョンを一覧表示するには、次の AWS CLI コマンドを使用します。

```
aws rds describe-db-engine-versions --engine sqlserver-dev-ee --output json --query
 "{DBEngineVersions: DBEngineVersions[?Status=='requires-custom-engine-version'].
{Engine: Engine, EngineVersion: EngineVersion, Status: Status, DBEngineDescription:
 DBEngineDescription, DBEngineVersionDescription: DBEngineVersionDescription}}"
```

コマンドは次の例のような出力を返します。

```
{
    "DBEngineVersions": [
        {
            "Engine": "sqlserver-dev-ee",
            "EngineVersion": "16.00.4215.2.v1",
            "Status": "requires-custom-engine-version",
            "DBEngineDescription": "Microsoft SQL Server Enterprise Developer Edition",
            "DBEngineVersionDescription": "SQL Server 2022 16.00.4215.2.v1"
        }
    ]
}
```

Developer Edition (Standard Edition 機能) の CEV 作成でサポートされているすべてのエンジンバージョンを一覧表示するには、次の AWS CLI コマンドを使用します。

```
aws rds describe-db-engine-versions --engine sqlserver-dev-se --output json --query
 "{DBEngineVersions: DBEngineVersions[?Status=='requires-custom-engine-version'].
{Engine: Engine, EngineVersion: EngineVersion, Status: Status, DBEngineDescription:
 DBEngineDescription, DBEngineVersionDescription: DBEngineVersionDescription}}"
```

コマンドは次の例のような出力を返します。

```
{
    "DBEngineVersions": [
        {
            "Engine": "sqlserver-dev-se",
            "EngineVersion": "17.00.4045.5.v1",
            "Status": "requires-custom-engine-version",
            "DBEngineDescription": "Microsoft SQL Server Standard Developer Edition",
            "DBEngineVersionDescription": "SQL Server 2025 17.00.4045.5.v1"
        }
    ]
}
```

エンジンバージョンステータス (`requires_custom_engine_version`) は、サポートされているテンプレートエンジンバージョンを識別します。これらのテンプレートには、インポートできる SQL Server バージョンが表示されます。

制限事項

Amazon RDS の Developer Edition には、次の制限が適用されます。

- サポートされているインスタンスクラスは、SQL Server のバージョンによって異なります。現在のリストについては、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照するか、describe-orderable-db-instance-options AWS CLI コマンドを使用します。
- Amazon RDS は、このエディションのマルチ AZ 配置またはリードレプリカをサポートしていません。
- 独自の SQL Server インストールメディアを提供および管理する必要があります。
- SQL Server Developer Edition (`sqlserver-dev-ee` および `sqlserver-dev-se`) のカスタムエンジンバージョンをリージョンまたはアカウント間で共有することはできません。

RDS for SQL Server の CEV の準備

前提条件

カスタムエンジンバージョンを作成する前に、次の前提条件を満たしていることを確認してください。

SQL Server Developer Edition インストールメディアを準備する

Microsoft から SQL Server Developer Edition インストールメディアを取得し、S3 にアップロードする準備をする必要があります。

Microsoft からインストールメディアをダウンロードするには

1. オプション A: Visual Studio サブスクリプションを使用して Developer Edition ISO をダウンロードします。英語版のみサポートされています。
2. オプション B: SQL Server インストーラを使用する

   a. Microsoft SQL Server のダウンロードページから、対象エディションの SQL Server Developer Edition インストーラをダウンロードします。

      - Developer Edition (Enterprise Edition 機能) – SQL Server 2025 セクションから Developer Edition インストーラをダウンロードします。ISO ファイルの名前は `SQLServer2025-x64-ENU-EntDev.iso` です。
      - Developer Edition (Standard Edition 機能) – SQL Server 2025 セクションから Developer (Standard) Edition インストーラをダウンロードします。ISO ファイルの名前は `SQLServer2025-x64-ENU-StdDev.iso` です。

      英語版のみサポートされています。

   b. インストーラを実行し、[メディアのダウンロード] を選択して ISO 全体をダウンロードします。

   c. 優先言語として [英語] を選択します。

   d. メディアタイプとして [ISO] を選択します。

   e. [ダウンロード] を選択します。

累積更新をダウンロードするには

1. Microsoft Catalog Update ページにアクセスします。
2. RDS for SQL Server でサポートされている SQL Server Developer Edition (例: SQL Server 2025 累積更新) を見つけます。
3. サポートされている最新の CU 実行可能ファイルをダウンロードし、マシンに保存します。

4. ファイル例: `SQLServer2025-KB5084896-x64.exe` (CU5 for SQL Server 2025)。SQL Server の各バージョンに必要な CU ファイル名については、以下のサポートされているバージョンの表を参照してください。

> **Important**
>
> RDS for SQL Server は、特定の累積更新 (CU) バージョンのみをサポートします。次の表に示すバージョンを正確に使用する必要があります。新しい CU バージョンは、RDS と互換性がない可能性があるため、Microsoft から入手できる場合でも使用しないでください。

または、以下から必要な累積更新 (CU) ファイルを直接ダウンロードすることもできます。

次の表に、RDS で使用するためにサポートされている SQL Server Developer Edition のバージョンとそれに対応する累積更新を示します。

SQL Server バージョン	サポートされている CU	KB Article	ダウンロードファイル名
SQL Server 2025 Enterprise Edition	CU5	KB5084896	SQLServer2025-KB5084896-x64.exe
SQL Server 2025 (Standard Edition 機能)	CU5	KB5084896	SQLServer2025-KB5084896-x64.exe
SQL Server 2022	CU21	KB5065865	SQLServer2022-KB5065865-x64.exe
SQL Server 2019	CU32 GDR	KB5068404	SQLServer2019-KB5068404-x64.exe

RDS for SQL Server のカスタムエンジンバージョンの作成

RDS for SQL Server のカスタムエンジンバージョン (CEV) は、Amazon RDS にインポートされた SQL Server Developer Edition インストールメディアで構成されます。ベース ISO インストーラと累積更新ファイル (.exe) を Amazon S3 バケットにアップロードする必要があります。アップロードし

たら、Amazon S3 の場所を RDS に提供して、CEV をダウンロード、検証、作成する必要があります。

命名に関する制限

CEV を作成するときは、特定の命名規則に従う必要があります。

- CEV 名はパターン `major-version.minor-version.customized-string` に従う必要があります。
- `customized-string` には、1～50 文字の英数字、アンダースコア、ダッシュ、ピリオドを含めることができます。例: `17.00.4045.5.my-dev-cev` SQL Server 2025 用。

Developer Edition (Enterprise Edition 機能) でサポートされているすべてのエンジンバージョンを一覧表示するには、次の AWS CLI コマンドを使用します。

```
aws rds describe-db-engine-versions --engine sqlserver-dev-ee --output json --query
 "{DBEngineVersions: DBEngineVersions[?Status=='requires-custom-engine-version'].
{Engine: Engine, EngineVersion: EngineVersion, Status: Status, DBEngineDescription:
 DBEngineDescription, DBEngineVersionDescription: DBEngineVersionDescription}}"

{
    "DBEngineVersions": [
        {
            "Engine": "sqlserver-dev-ee",
            "EngineVersion": "17.00.4045.5.v1",
            "Status": "requires-custom-engine-version",
            "DBEngineDescription": "Microsoft SQL Server Enterprise Developer Edition",
            "DBEngineVersionDescription": "SQL Server 2025 17.00.4045.5.v1"
        }
    ]
}
```

Developer Edition (Standard Edition 機能) でサポートされているすべてのエンジンバージョンを一覧表示するには、次の AWS CLI コマンドを使用します。

```
aws rds describe-db-engine-versions --engine sqlserver-dev-se --output json --query
 "{DBEngineVersions: DBEngineVersions[?Status=='requires-custom-engine-version'].
{Engine: Engine, EngineVersion: EngineVersion, Status: Status, DBEngineDescription:
 DBEngineDescription, DBEngineVersionDescription: DBEngineVersionDescription}}"
```

```
{
    "DBEngineVersions": [
        {
            "Engine": "sqlserver-dev-se",
            "EngineVersion": "17.00.4045.5.v1",
            "Status": "requires-custom-engine-version",
            "DBEngineDescription": "Microsoft SQL Server Standard Developer Edition",
            "DBEngineVersionDescription": "SQL Server 2025 17.00.4045.5.v1"
        }
    ]
}
```

AWS CLI

Developer Edition (Enterprise Edition 機能) のカスタムエンジンバージョンを作成するには

- create-custom-db-engine-version コマンドを使用します。

  以下のオプションは必須です。

  - `--engine`
  - `--engine-version`
  - `--database-installation-files-s3-bucket-name`
  - `--database-installation-files`
  - `--region`

  また、以下のオプションを指定することもできます。

  - `--database-installation-files-s3-prefix`
  - `--description`
  - `--tags`

```
aws rds create-custom-db-engine-version \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.cev-dev-ss2025-cu5 \
--region us-west-2 \
--database-installation-files-s3-bucket-name my-s3-installation-media-bucket \
--database-installation-files-s3-prefix sqlserver-dev-media \
```

```
--database-installation-files "SQLServer2025-x64-ENU-EntDev.iso" "SQLServer2025-
KB5084896-x64.exe"
```

Developer Edition (Standard Edition 機能) のカスタムエンジンバージョンを作成するには

- create-custom-db-engine-version コマンドを --engine sqlserver-dev-se と共に使用します。

```
aws rds create-custom-db-engine-version \
--engine sqlserver-dev-se \
--engine-version 17.00.4045.5.cev-dev-se-ss2025-cu5 \
--region us-west-2 \
--database-installation-files-s3-bucket-name my-s3-installation-media-bucket \
--database-installation-files-s3-prefix sqlserver-dev-media \
--database-installation-files "SQLServer2025-x64-ENU-StdDev.iso" "SQLServer2025-
KB5084896-x64.exe"
```

CEV の作成には通常 15～30 分かかります。CEV 作成の進行状況をモニタリングするには、次のコマンドを使用します。

```
# For Developer Edition (Enterprise Edition capabilities)
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2

# For Developer Edition (Standard Edition capabilities)
aws rds describe-db-engine-versions \
--engine sqlserver-dev-se \
--engine-version 17.00.4045.5.my-dev-se-cev \
--region us-west-2
```

RDS for SQL Server CEV のライフサイクル

RDS for SQL Server で SQL Server Developer Edition を使用する場合、カスタムエンジンバージョンはさまざまなライフサイクル状態に移行します。

ライフサイクル状態	説明	いつ発生するか	使用可能なアクション
検証保留中	CEV 作成時の初期状態	これは、`create-custom-db-engine-version` コマンドを使用して作成した後の最初の状態です。	`describe-db-engine-version` を介してステータスをモニタリングします。
検証しています	CEV 検証状態	Amazon RDS はカスタムエンジンバージョン (CEV) を検証しています。この非同期プロセスが完了するまでに時間がかかる場合があります。	検証が完了するまでステータスをモニタリングします。
利用可能	カスタムエンジンバージョン (CEV) の検証が正常に完了しました。	カスタムエンジンバージョン (CEV) が利用可能になりました。Amazon RDS は SQL Server ISO と累積更新ファイルの検証に成功しました。この CEV を使用して DB インスタンスを作成できるようになりました。	この CEV を使用して DB インスタンスを作成する
失敗	検証チェックに失敗したため、RDS for SQL Server はカスタムエンジンバージョン (CEV) を作成できません。	ISO と累積メディアの検証に失敗しました。	ISO 検証に失敗しました。`describe-db-engine-version` で障害の理由を確認し、ハッシュの不一致やコン

ライフサイクル状態	説明	いつ発生するか	使用可能なアクション
			テンツの破損などのファイルの問題を修正してから、カスタムエンジンバージョン (CEV) を再作成します。
削除中	カスタムエンジンバージョン (CEV) は削除中です	顧客が `delete-custom-db-engine-version` を呼び出した後、削除ワークフローが完了するまで。	`describe-db-engine-version` を介してステータスをモニタリングします。
incompatible-installation-media	Amazon RDS は、カスタムエンジンバージョン (CEV) 用に提供されたインストールメディアを検証できませんでした	カスタムエンジンバージョン (CEV) の検証に失敗しました。これは終了状態です。	検証が失敗した理由については、`describe-db-engine-versions` の failureReason を参照してください。CEV を削除します。

CEV ステータスの説明

AWS CLI を使用して CEV の状態を確認できます。

```
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2 \
--query 'DBEngineVersions[0].{Version:EngineVersion,Status:Status}'
```

Developer Edition (Standard Edition 機能) の場合は、`sqlserver-dev-ee` を `sqlserver-dev-se` に置き換えます。

サンプル出力

```
| DescribeDBEngineVersions                          |
+------------+--------------------------------------+
| Status | Version                                  |
+------------+--------------------------------------+
| available | 17.00.4045.5.cev-dev-ss2025-cu5         |
+------------+--------------------------------------+
```

CEV が `failed` ステータスを表示している場合、次のコマンドを使用して理由を特定できます。

```
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2 \
--query 'DBEngineVersions[0].
{Version:EngineVersion,Status:Status,FailureReason:FailureReason}'
```

RDS for SQL Server Developer Edition DB インスタンスの作成

RDS for SQL Server で Developer Edition インスタンスを起動するには、まず `create-custom-db-engine-version` で CEV を作成します。カスタムエンジンバージョンが使用可能な状態になったら、CEV を使用して Amazon RDS データベースインスタンスを作成できます。

Developer Edition インスタンス作成の主な違い

パラメータ	Developer Edition
`--engine`	`sqlserver-dev-ee` (Enterprise Edition 機能) または `sqlserver-dev-se` (Standard Edition 機能、SQL Server 2025 のみ)
`--engine-version`	カスタムエンジンバージョン (例: `17.00.4045.5.cev-dev-ss2025-cu5`)
`--license-model`	自分のライセンス使用

コンソール

AWS AWS マネジメントコンソール を使用して SQL Server Developer Edition DB インスタンスを作成するには:

1. AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。
2. ナビゲーションペインで [データベース] を選択してから、[データベースの作成] を選択します。
3. [データベース作成方法を選択] で、[フル設定] を選択します。
4. [エンジンのオプション] で、[Microsoft SQL Server] を選択します。
5. [エディション] で、次のいずれかを選択します。
   - SQL Server Developer Edition – Enterprise Edition 機能を提供します。`sqlserver-dev-ee` エンジンに対応します。
   - SQL Server Standard Developer Edition – Standard Edition 機能を提供します。`sqlserver-dev-se` エンジンに対応します。SQL Server 2025 でのみ使用できます。
6. ライセンスモデルの場合、Developer Edition を選択すると、コンソールは自動的に [ライセンス: 該当なし] を選択します。この値を変更することはできません。
7. [カスタムエンジンバージョン] で、作成した CEV (例: *`17.00.4045.5.cev-dev-ss2025-cu5`*) を選択します。

   > **Note**
   >
   > `available` 状態の CEV のみが一覧に表示されます。CEV が一覧表示されていない場合は、そのステータスを確認します。詳細については、「RDS for SQL Server のカスタムエンジンバージョンの作成」を参照してください。

8. [DB インスタンス識別子] として、DB インスタンスの一意の名前を入力します。
9. 必要に応じて、[DB インスタンスクラス]、[ストレージ]、[接続]、[データベース認証]、およびその他の設定を指定します。詳細については、「DB インスタンスの作成」を参照してください。
10. [データベースの作成] を選択します。

AWS CLI

SQL Server Developer Edition DB を作成するには、以下のパラメータを指定して create-db-instance コマンドを使用します。

以下のオプションは必須です。

- `--db-instance-identifier`
- `--db-instance-class`
- `--engine` – (`sqlserver-dev-ee` Enterprise Edition 機能) または `sqlserver-dev-se` (Standard Edition 機能、SQL Server 2025 のみ)
- `--region`

例:

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
--db-instance-identifier my-dev-sqlserver \
--db-instance-class db.m6i.xlarge \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.cev-dev-ss2025-cu5 \
--allocated-storage 200 \
--master-username admin \
--master-user-password changeThisPassword \
--license-model bring-your-own-license \
--vpc-security-group-ids sg-xxxxxxxxx \
--db-subnet-group-name my-db-subnet-group \
--backup-retention-period 7 \
--region us-west-2
```

Windows の場合:

```
aws rds create-db-instance ^
--db-instance-identifier my-dev-sqlserver ^
--db-instance-class db.m6i.xlarge ^
--engine sqlserver-dev-ee ^
--engine-version 17.00.4045.5.cev-dev-ss2025-cu5 ^
--allocated-storage 200 ^
--master-username admin ^
--master-user-password master_user_password ^
--license-model bring-your-own-license ^
--vpc-security-group-ids sg-xxxxxxxxx ^
--db-subnet-group-name my-db-subnet-group ^
--backup-retention-period 7 ^
--region us-west-2
```

> **Note**
>
> Developer Edition (Standard Edition 機能) の場合は、`--engine sqlserver-dev-se` を `sqlserver-dev-se` CEV バージョンで使用します。

データベースのマイナーバージョンアップグレードの適用

RDS for SQL Server Developer Edition では、データベースのマイナーバージョンアップグレードを適用するために、最新の累積更新を含む新しいカスタムエンジンバージョン (CEV) を作成する必要があります。SQL Server Developer Edition のデータベースマイナーバージョンアップグレードには、次のステップが含まれます。

1. アップグレードする前に、インスタンスの現在のエンジンバージョンを確認し、Amazon RDS でサポートされているバージョンからターゲットデータベースエンジンのバージョンを特定します。Amazon RDS で使用できる SQL Server のバージョンについての詳細は、「RDS for SQL Server での SQL Server Developer Edition の使用」を参照してください。
2. インストールメディア (ISO および CU) を取得してアップロードし、新しいカスタムエンジンバージョンを作成します。
3. 新しい CEV で Amazon RDS `modify-db-instance` を使用してデータベースのマイナーバージョンアップグレードを適用します。

```
aws rds modify-db-instance \
--db-instance-identifier <instance-id> \
--engine-version <new-cev-version> \
--no-apply-immediately ## use --apply-immediately for immediate update
```

> **Note**
>
> `--no-apply-immediately` (デフォルト) を使用して、次のメンテナンスウィンドウ中に変更を適用します。

カスタムエンジンバージョンの表示と管理

すべての RDS for SQL Server Developer Edition CEV を表示するには、Enterprise Edition 機能の場合は `--engine sqlserver-dev-ee`、Standard Edition 機能の場合は `--engine sqlserver-`

dev-se (SQL Server 2025 のみ) を指定して describe-db-engine-versions コマンドを使用します。

```
# Developer Edition (Enterprise Edition capabilities)
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--include-all \
--region us-west-2

# Developer Edition (Standard Edition capabilities, SQL Server 2025 only)
aws rds describe-db-engine-versions \
--engine sqlserver-dev-se \
--include-all \
--region us-west-2
```

特定の CEV の詳細を表示するには、次のコマンドを使用します。

```
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2
```

> **Note**
>
> このコマンドは、available ステータスの CEV のみを返します。検証または他の状態の CEV を表示するには、--include-all フラグを含めます。Standard Edition 機能の CEV を表示するときは、sqlserver-dev-ee を sqlserver-dev-se に置き換えます。

カスタムエンジンバージョンの削除

CEV を削除する前に、CEV が次のいずれかで使用中ではないことを確認してください。

- Amazon RDS DB インスタンス
- Amazon RDS DB インスタンスのスナップショット
- Amazon RDS DB インスタンスの自動バックアップ

> **Note**
>
> 関連付けられたリソースがある場合、CEV を削除することはできません。

カスタムエンジンバージョンを削除するには、delete-custom-db-engine-version コマンドを使用します。

- `--engine`: `sqlserver-dev-ee` (Enterprise Edition 機能) または `sqlserver-dev-se` (Standard Edition 機能、SQL Server 2025 のみ) を指定します。
- `--engine-version`: 削除する正確な CEV バージョン識別子
- `--region`: CEV が存在する AWS リージョン

```
aws rds delete-custom-db-engine-version \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2
```

CEV 削除プロセスをモニタリングするには、`describe-db-engine-versions` コマンドを使用して RDS for SQL Server CEV エンジンのバージョンを指定します。

```
aws rds describe-db-engine-versions \
--engine sqlserver-dev-ee \
--engine-version 17.00.4045.5.my-dev-cev \
--region us-west-2
```

ステータス値:

- `deleting`: CEV 削除を実行中
- 結果は返されません: CEV が正常に削除されました

RDS for SQL Server の SQL Server Developer Edition のトラブルシューティング

次の表に、SQL Server Developer Edition for RDS for SQL Server を使用する際の一般的なエラーと対応する解決策を示します。

一般的なエラーと解決策

エラーコード	説明	ソリューション
InvalidParameterValue	無効な CEV パラメータまたはファイル参照	ファイル名、パス、パラメータ構文を検証する
DBSubnetGroupNotFound	サブネットグループが存在しません	サブネットグループの作成または名前の検証
InvalidVPCNetworkState	VPC 設定の問題	VPC、サブネット、ルーティングテーブルを確認する
InvalidEngineVersion	CEV が使用できないか無効です	CEV のステータスと名前を確認する
InvalidDBInstanceClass	インスタンスクラスはサポートされていません	サポートされているインスタンスクラスの選択
CustomDBEngineVersionQuotaExceededFault	カスタムエンジンバージョンの最大数に達しました	必要に応じてサービスクォータを引き上げるか、必要に応じて未使用の CEV を削除する
CreateCustomDBEngineVersionFault	Amazon RDS は、Amazon S3 バケット内の指定されたインストーラファイルにアクセスできませんでした。	Amazon RDS は、指定された Amazon S3 の場所にある SQL Server インストールファイルにアクセスできません。Amazon S3 バケットポリシーで Amazon RDS サービスプリンシパル (rds.amazonaws.com) s3:GetObject アクセス許可を付与します。Amazon S3 バケットリージョンが CEV を作成するリージョンと同じであることを確認します。

RDS for SQL Server の Bring Your Own Media (BYOM)

RDS for SQL Server の Bring Your Own Media (BYOM) では、Microsoft のソフトウェアアシュアランスによるライセンスモビリティのメリットを活かして、Amazon RDS で既存の SQL Server ライセンスを使用できるようになります。

Amazon RDS の Microsoft SQL Server のライセンスは、次の 2 つの方法で取得できます。

- ライセンス込み AWS が SQL Server ライセンスを提供します。ライセンス料金は RDS 時間料金に含まれています。
- Bring Your Own Media (BYOM) Microsoft のソフトウェアアシュアランスによるライセンスモビリティのメリットを活かして、既存の SQL Server ライセンスを持ち込むことができます。AWS は SQL Server ライセンス料金を請求しません。

ユースケースと既存のライセンス契約によって、ワークロードに対してコスト効率の高いモデルが決まります。BYOM を使用するには、アクティブなソフトウェアアシュアランスのある SQL Server ライセンスが必要です。

SQL Server の Release To Manufacturing (RTM) インストールメディアを ISO ファイルとして Microsoft からダウンロードし、Amazon S3 バケットにアップロードします。Amazon RDS は、そのメディアを使用して SQL Server をインストールし、必要な累積更新を自動的に適用します。

Amazon RDS は、インスタンスを作成した後、ライセンス込みインスタンスと同じインフラストラクチャを使用して、パッチ適用、自動バックアップ、高可用性、モニタリングなど、進行中のすべてのデータベースオペレーションを管理します。BYOM は、SQL Server Enterprise Edition または Standard Edition をサポートしています。

Amazon RDS SQL Server は Developer Edition の BYOM もサポートしていますが、機能にはいくつかの違いがあります。詳細については、以下を参照してください。the section called “Developer Edition との比較”

利点

- 既存のライセンスを使用する AWS インフラストラクチャ (コンピューティング、ストレージ、I/O、データ転送) および Windows OS の料金が発生します。既存の Microsoft ライセンスで、SQL Server ライセンスコストがカバーされます。AWS は SQL Server のライセンス料金を請求しません。

- フルマネージド型オペレーション Amazon RDS は、インフラストラクチャ管理、自動バックアップ、ポイントインタイムリカバリ、マルチ AZ フェイルオーバー、エンジンバージョンアップグレードなど、進行中のすべてのデータベースオペレーションを管理します。
- 完全な機能同等性 BYOM インスタンスは、ライセンス込みインスタンスと同じ RDS for SQL Server 機能セットをサポートします。例外については、「the section called “制限事項”」を参照してください。
- ライセンス使用状況の可視性 Amazon RDS は AWS License Manager と統合されているため、AWS 環境全体の SQL Server ライセンス消費量を追跡し、ライセンス契約への準拠を維持できます。

BYOM の仕組み

RDS for SQL Server で BYOM を使用するためのエンドツーエンドのワークフローは、次のステップで構成されます。

1. インストールメディアを準備する Microsoft から SQL Server RTM をダウンロードし、Amazon S3 バケットにアップロードします。詳細については、「the section called “開始する前に”」を参照してください。
2. BYOM エンジンバージョンを作成する アップロードしたインストールメディアを参照する BYOM エンジンバージョンを作成します。Amazon RDS はファイルを検証し、インスタンスを起動するための再利用可能な BYOM エンジンバージョンを作成します。詳細については、「the section called “BYOM エンジンバージョンの作成と管理”」を参照してください。
3. BYOM インスタンスを起動する `bring-your-own-media` オプションとして `license-model` を使用して RDS for SQL Server インスタンスを起動します。詳細については、「the section called “BYOM DB インスタンスの作成”」を参照してください。

> **Note**
>
> AWS マネジメントコンソールを使用する場合は、ステップ 2 をスキップできます。コンソールから BYOM インスタンスを起動するときに、ターゲットのメジャーエンジンバージョンとマイナーエンジンバージョンを選択すると、Amazon RDS によって BYOM エンジンバージョンが自動的に作成されます。ステップ 2 は、AWS CLI または API を使用する場合にのみ必要です。

> **Note**
>
> BYOM を使用する場合、お客様には以下の責任があります。
>
> 1. 有効な SQL Server インストールメディア (RTM ISO ファイル) の提供
> 2. ライセンスモビリティによる Microsoft ライセンスコンプライアンスの維持
>
> Amazon RDS は、ライセンス込みインスタンスの場合と同様に、その他のすべてのデータベースオペレーションを管理します。

BYOM とライセンス込みの違い

側面	ライセンス込み (LI)	Bring Your Own Media (BYOM)
SQL Server ライセンス	含まれる	Microsoft のライセンスモビリティプログラムを通じて独自のライセンスを提供する
SQL Server のインストールメディア	AWS が提供および管理する	お客様が SQL Server RTM を Amazon S3 にアップロードする
エンジンのバージョン管理	AWS がすべてのエンジンバージョンを管理します。お客様が使用するバージョンを選択します。	ターゲットのマイナーバージョンごとに BYOM エンジンバージョンを作成し、それを使用するようにインスタンスを作成または変更します。RTM ファイルはエンジンのマイナーバージョン間で再利用されます。
ライセンスコンプライアンス	AWS が管理する	お客様がライセンスモビリティを通じてコンプライアンスを維持します。AWSLicense Manager は使用状況の追跡を提供します。

サポートされているエディションとメジャーバージョン

Edition	メジャーバージョン
Enterprise Edition (EE)	SQL Server 2019、SQL Server 2022、SQL Server 2025
Standard Edition (SE)	SQL Server 2019、SQL Server 2022、SQL Server 2025

サポートされるインスタンスタイプ

現在の BYOM インスタンスのサポートについては、Amazon RDS for SQL Server の料金ページを参照してください。

リージョンの可用性

RDS for SQL Server の BYOM は、次の AWS リージョンで使用できます。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (マレーシア)
- アジアパシフィック (メルボルン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ニュージーランド)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (台北)
- アジアパシフィック (タイ)

- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- 欧州 (パリ)
- 欧州 (スペイン)
- 欧州 (ストックホルム)
- メキシコ (中部)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

料金

お客様のリージョンにおける現在の料金については、Amazon RDS for SQL Server の料金ページを参照してください。

AWS License Manager 統合

Amazon RDS は AWS License Manager と統合されており、SQL Server のライセンス使用状況の追跡と管理に役立ちます。License Manager を設定すると、Amazon RDS は BYOM インスタンスを自動的に検出し、vCPU の使用状況を報告します。License Manager は以下を提供します。

- すべての BYOM インスタンスでの SQL Server ライセンス消費の可視性
- 独自のライセンスを使用するインスタンスとライセンス込みのインスタンスの追跡
- Microsoft ライセンスコンプライアンス監査に対応可能なレポート

License Manager を設定しない場合でも、BYOM インスタンスは正常に機能します。ただし、ライセンスの使用状況をモニタリングし、コンプライアンスを個別に維持する責任はお客様にあります。

AWS License Manager に追加料金はかかりません。料金情報については、「AWS License Manager の料金」を参照してください。

現在の制限事項

- メジャーバージョンアップグレード インプレースメジャーバージョンアップグレードはサポートされていません。例えば、BYOM インスタンスを SQL Server 2019 から SQL Server 2022 または SQL Server 2025 に直接アップグレードすることはできません。新しいメジャーバージョンに移行するには、ターゲットバージョンで新しい BYOM DB インスタンスを作成し、ネイティブバックアップと復元を使用してデータを移行します。
- BYOM エンジンのバージョンは、お客様の AWS アカウントとリージョンに固有のものです。各アカウントは、独自の BYOM エンジンバージョンを個別に作成する必要があります。現在、クロスアカウントオペレーションはサポートされていません。
- クロスリージョンオペレーションの BYOM エンジンバージョンはリージョン固有です。クロスリージョンオペレーションを実行するには、まずターゲットリージョンで同じエンジンバージョンの BYOM エンジンバージョンを作成する必要があります。これは、以下に適用されます。
  - クロスリージョンリードレプリカ
  - クロスリージョンスナップショットのコピー
  - クロスリージョン自動バックアップレプリケーション
- BYOM は、次の機能をサポートしていません。
  - SQL Server Analysis Services (SSAS)
  - SQL Server Reporting Services (SSRS)
- BYOM は現在、SQL Server 2019 エンジンバージョン `15.00.4043.16.v1` をサポートしていません。SQL Server 2019 で BYOM を使用するには、サポートされている別のエンジンバージョンを選択します。
- BYOM は、バージョン `17.00.4045.5.v1` 以降、SQL Server 2025 Enterprise Edition および Standard Edition をサポートしています。

RDS for SQL Server の BYOM エンジンバージョンの作成と管理

BYOM エンジンバージョンを作成するには、1 回限りのセットアップ (ライセンスモビリティの検証、RTM メディアの取得、S3 へのアップロード) が必要です。セットアップ後、Amazon RDS コンソールまたは AWS CLI を使用してエンジンバージョンを作成できます。

開始する前に

ステップ 1: Microsoft ライセンスモビリティの検証を完了する

BYOM には、ソフトウェアアシュアランスによるライセンスモビリティが必要です。AWS 上に SQL Server ライセンスをデプロイする前に、SQL Server ライセンスには有効なソフトウェアアシュアランスが適用されている必要があります。

検証プロセスを完了するには、以下を実行します。

1. 適格性の確認 AWS ライセンスモビリティページにアクセスして、要件と資格基準を確認してください。
2. 検証フォームを送信する Microsoft ライセンスモビリティ検証フォームをダウンロードして入力します。ライセンスパートナーを通じて、または直接 Microsoft にフォームを送信します。
3. 承認を待つ検証が承認されると Microsoft から確認 E メールが送信されます。

製品を AWS に持ち込んでから 10 日以内に、直接、または Microsoft 販売代理店を通じてフォームを Microsoft に提出する必要があります。

> **Important**
>
> AWS は、Microsoft ライセンスコンプライアンスを検証または適用しません。お客様には、Microsoft ライセンス契約への準拠を維持する責任があります。

ステップ 2: SQL Server インストールメディアを取得する

使用する SQL Server メジャーバージョンの Release to Manufacturing (RTM) ファイルをダウンロードします。RTM ファイルは、SQL Server メジャーバージョンとエディション (SQL Server 2019 Enterprise Edition など) の基本インストールメディアです。

RTM ファイルは、次のいずれかの Microsoft ソースから取得できます。

- Visual Studio サブスクリプション アクティブな Visual Studio サブスクリプションがある場合は、サブスクライバーのダウンロードページから RTM ファイルをダウンロードします。
- Microsoft 365 管理センター ボリュームライセンスで製品を購入した場合は、Microsoft 365 管理センターのダウンロードページから RTM ファイルをダウンロードします。

> **Important**
>
> 英語の Core ベースの ISO ファイルのみをダウンロードします。Server+CAL ベースの ISO ファイルを使用しないでください。

> **Important**
>
> SQL Server 2019 Release to Manufacturing (RTM) インストールメディア (Standard および Enterprise Core エディション) の場合は、Microsoft 365 管理センターからのみダウンロードしてください。Visual Studio サブスクリプションからダウンロードされた SQL Server 2019 メディアはサポートされていません。SQL Server 2022 以降では、Microsoft 365 管理センターまたは Visual Studio サブスクリプションから RTM インストールメディアをダウンロードできます。

ステップ 3: Amazon S3 にインストールメディアをアップロードする

BYOM エンジンバージョンを作成する予定の AWS リージョンとアカウントの Amazon S3 バケットに SQL Server RTM ファイルをアップロードします。AWS CLI を使用して RTM ファイルをアップロードするには、以下のコマンドを実行します。

```
aws s3 cp SQLServer2022-x64-ENU-Enterprise.iso s3://my-sqlserver-media/ISOs/
```

> **Note**
>
> BYOM エンジンの作成時に累積更新 (CU) ファイルを指定しない場合、Amazon RDS はエンジンバージョンの作成時に Microsoft から必要な CU を自動的にダウンロードします。

BYOM エンジンバージョンの作成

BYOM エンジンバージョンの作成対象となるエンジンバージョンを確認するには、`describe-db-engine-versions` を使用します。サポートされているバージョンについては、「the section called “Bring Your Own Media (BYOM)”」を参照してください。

コンソール

1. AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。
2. ナビゲーションペインで、[カスタムエンジンバージョン] を選択します。
3. ドロップダウンから [RDS for SQL Server] を選択します。
4. [カスタムエンジンバージョンの作成] を選択します。
5. [エンジンのタイプ] に [SQL Server] を選択します。
6. [データベース管理のタイプ] で、[Amazon RDS] を選択します。
7. [エディション] で、次のいずれかを選択します。
   - SQL Server Enterprise Edition
   - SQL Server Standard Edition
8. [エンジンバージョン] では、インストールファイルに一致する SQL Server バージョン (例: SQL Server 2022 `16.00.4175.1.v1` または SQL Server 2025 `17.00.4045.5.v1`) を選択します。
9. [インストールメディア] で、SQL Server RTM メディアを含む S3 URI を入力するか、[S3 をブラウズ] を選択してファイルを見つけて選択します。
10. [カスタムエンジンバージョンの作成] を選択します。

コンソールには、初期ステータスが [保留中の検証] の新しい BYOM エンジンバージョンが表示されます。ステータスは自動的に進行します。詳細については、「the section called “ライフサイクル状態”」を参照してください。

CLI

`create-custom-db-engine-version` コマンドを使用して、インストールメディアから BYOM エンジンバージョンを作成します。Amazon RDS は、必要な累積更新プログラムを Microsoft から自動的にダウンロードします。

```
aws rds create-custom-db-engine-version \
    --engine sqlserver-ee \
```

```
    --engine-version 16.00.4175.1.v1 \
    --database-installation-files-s3-bucket-name my-sqlserver-media \
    --database-installation-files-s3-prefix "ISOs/" \
    --database-installation-files "SQLServer2022-x64-ENU-Enterprise.iso"
```

応答

```
{
    "Engine": "sqlserver-ee",
    "MajorEngineVersion": "16.00",
    "EngineVersion": "16.00.4175.1.v1",
    "DatabaseInstallationFiles": [
        "SQLServer2022-x64-ENU-Enterprise.iso"
    ],
    "DBEngineDescription": "Microsoft SQL Server Enterprise Edition",
    "DBEngineVersionArn": "arn:aws:rds:us-east-1:123456789012:cev:sqlserver-
ee/16.00.4175.1.v1/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    ....
    "Status": "pending-validation",
    .....
}
```

CLI パラメータ

[Parameter] (パラメータ)	必須	説明
--engine	はい	データベースエンジンのタイプ: sqlserver-ee、sqlserver-se 。
--engine-version	はい	ターゲットエンジンのバージョン (16.00.4175.1.v1 など)。
--database-installation-files-s3-bucket-name	はい	インストールメディアが保存されている S3 バケットの名前。バケットは BYOM エンジンバージョンと同じ AWS リージョンにある必要があります。
--database-installation-files-s3-prefix	いいえ	インストールメディアファイルが保存されている S3 キープレフィックス (フォルダパス)。

[Parameter] (パラメータ)	必須	説明
--database-installation-files	はい	インストールメディアのファイル名。

> **Note**
>
> AWS CLI を使用する場合、--database-installation-files-s3-bucket-name、--database-installation-files-s3-prefix、および --database-installation-files パラメータは、メジャーバージョンとエディション用に作成する最初の BYOM エンジンバージョンにのみ必要です。同じメジャーバージョン内の後続のマイナーバージョンでは、--engine と --engine-version のみが必要です。

BYOM エンジンバージョンについて

BYOM エンジンバージョンを作成した後、describe-db-engine-versions を使用してそのステータスと詳細を確認します。出力は、同じエンジンバージョンに対して 2 つのレコードを返します。1 つはライセンス込み (DBEngineMediaType: None)、もう 1 つは BYOM (DBEngineMediaType: Customer Provided) です。

例:

```
aws rds describe-db-engine-versions \
    --engine sqlserver-ee \
    --engine-version 16.00.4175.1.v1 \
    --include-all \
    --output table \
    --query "DBEngineVersions[].{Engine: Engine, Version: EngineVersion, Status:
 Status, MediaType: DBEngineMediaType}"
```

レスポンス:

```
-------------------------------------------------------------------
|                     DescribeDBEngineVersions                    |
+--------------+---------------------+------------+-----------------+
|    Engine    |      MediaType      |  Status    |      Version    |
+--------------+---------------------+------------+-----------------+
|  sqlserver-ee|  None               |  available |  16.00.4175.1.v1  |
```

```
|  sqlserver-ee|  Customer Provided  |  available |  16.00.4175.1.v1  |
+--------------+---------------------+------------+-------------------+
```

BYOM エンジンバージョンのライフサイクル状態

状態	説明
pending-validation	RDS はリクエストを受け入れ、処理待ちのキューに入れました。
validating	RDS は RTM メディアファイルを検証しています。
creating	RDS は RTM メディアファイルから SQL Server をインストールしています。
available	BYOM エンジンバージョンはインスタンス作成の準備ができました。
incompatible_installation_media	インストールファイルが無効であるか、破損しています。これは終了状態です。BYOM エンジンのバージョンを削除し、正しいファイルを使用して再試行します。

BYOM エンジンバージョンの変更

BYOM エンジンバージョンは、AWS マネジメントコンソールまたは AWS CLI を使用して変更できます。BYOM エンジンバージョンの説明またはそのアベイラビリティーステータスを変更できます。BYOM エンジンバージョンには、次のいずれかのステータス値があります。

- 使用可能 – この BYOM エンジンバージョンを使用して、新しい DB インスタンスを作成したり、DB インスタンスをアップグレードしたりできます。これは、新規作成された BYOM エンジンバージョンのデフォルトのステータスです。
- 非アクティブ – この BYOM エンジンバージョンで DB インスタンスを作成またはアップグレードすることはできません。この BYOM エンジンバージョンを使用して新しい DB インスタンスを作成するために、DB スナップショットを復元することはできません。

BYOM エンジンバージョンのステータスは、`available` から `inactive` または `inactive` から `available` に変更できます。ステータスを `inactive` に変更することで、BYOM エンジンバー

ジョンの誤用を防ぎ、また中止された BYOM エンジンバージョンを再び使用できるようになります。

```
aws rds modify-custom-db-engine-version \
    --engine sqlserver-ee \
    --engine-version 16.00.4175.1.v1 \
    --status inactive
```

応答

```
{
    "Engine": "sqlserver-ee",
    "MajorEngineVersion": "16.00",
    "EngineVersion": "16.00.4175.1.v1",
    ....
    "Status": "inactive",
    ....
}
```

> **Note**
>
> 非アクティブに設定した BYOM エンジンバージョンで実行されている既存のインスタンスは、引き続き正常に動作します。新しいインスタンスの作成および復元オペレーションのみが影響を受けます。

BYOM エンジンバージョンの削除

`delete-custom-db-engine-version` を使用して BYOM エンジンバージョンを完全に削除します。削除後、エンジンバージョンは LI のみの単一エントリに戻ります。

削除の前提条件

BYOM エンジンバージョンを削除する前に、以下を確認してください。

- BYOM バージョンを使用している DB インスタンスはありません。
- BYOM バージョンを参照する DB スナップショットはありません。

RDS リソースがいずれか 1 つでも存在する場合、API は削除リクエストに失敗します。

BYOM エンジンバージョンの削除 (CLI)

```
aws rds delete-custom-db-engine-version \
    --engine sqlserver-ee \
    --engine-version 16.00.4175.1.v1
```

応答

```
{
    "Engine": "sqlserver-ee",
    "MajorEngineVersion": "16.00",
    "EngineVersion": "16.00.4175.1.v1",
    .....
    "Status": "deleting",
    ....
}
```

考慮事項

- メジャーバージョンとエディションごとに 1 回、RTM ISO ファイルを提供します。同一のメジャーバージョンとエディション内のすべてのマイナーバージョンは、同一のインストールメディアを再利用します。
- Bring Your Own Media で SQL Server RTM を提供する場合、サーバー + CAL ベースの RTM ISO ファイルではなく、Core ベースの RTM ISO のみを使用する必要があります。

次のステップ

BYOM エンジンバージョンが `available` ステータスになったら、それを使用して RDS for SQL Server インスタンスを起動できます。「the section called “BYOM DB インスタンスの作成”」を参照してください。

RDS for SQL Server の BYOM DB インスタンスの作成

BYOM インスタンスは、Amazon RDS コンソールまたは AWS CLI を使用して作成できます。

コンソール: [データベースの作成] ページから、ライセンス込み (LI) インスタンスを作成するのと同じ方法で BYOM インスタンスを作成できます。コンソールは、お客様に代わってバックグラウンドで BYOM エンジンバージョンを作成します。提供されたインストールメディアからエンジンバージョンを構築すると、インスタンスの作成の所要時間が全体で約 20 分長くなることに注意してください。

この追加の待ち時間を回避するには、[カスタムエンジンバージョン] ページから BYOM エンジンバージョンを事前に作成できます。エンジンバージョンが作成されると、それに対するインスタンスの起動は LI インスタンスと同じタイムラインに従います。

CLI: AWS CLI を使用して、まず `create-custom-db-engine-version` を呼び出して BYOM エンジンバージョンを作成し、次に `create-db-instance` を呼び出してデータベースを起動します。この 2 段階のプロセスは、既存の BYOM インスタンスを新しいマイナーバージョンに変更する場合にも適用されます。まずターゲット BYOM エンジンバージョンを作成し、`Available` ステータスになるまで待機してから、`modify-db-instance` を呼び出してアップグレードします。

前提条件

BYOM DB インスタンスを作成する前に、以下を確認してください。

- 「the section called “BYOM エンジンバージョンの作成と管理”」のステップを完了していること。

BYOM DB インスタンスの作成 (コンソール)

1. AWS マネジメントコンソールにサインインして、Amazon RDS コンソールを開きます。
2. ナビゲーションペインで [データベース] を選択してから、[データベースの作成] を選択します。
3. [データベース作成方法を選択] で、[フル設定] を選択します。
4. [エンジンのオプション] で、[Microsoft SQL Server] を選択します。
5. [テンプレート] では、ユースケースに基づいて [本番稼働用] または [開発/テスト] を選択します。
6. [データベース管理のタイプ] で、[Amazon RDS] を選択します。
7. [エディション] で、[SQL Server Standard Edition] または [SQL Server Enterprise Edition] を選択します。
8. [ライセンスモデル] には、[BYOM (Bring Your Own Media)] を選択します。

9. [メジャーエンジンバージョン] で、インストールメディアに一致する SQL Server バージョン (例: 2022 または 2025) を選択します。
10. [マイナーエンジンバージョン] で、ターゲットのマイナーバージョン (例: SQL Server 2022 の場合は 16.00.4175.1.v1、SQL Server 2025 の場合は 17.00.4045.5.v1) を選択します。
11. [DB インスタンス識別子] として、DB インスタンスの一意の名前を入力します。
12. ライセンス込みインスタンスの場合と同様に、残りの設定 (DB インスタンスクラス、ストレージ、接続、認証、バックアップ、メンテナンス) を設定します。
13. [データベースの作成] を選択します。

Amazon RDS は BYOM エンジンバージョンを自動的に作成し、インスタンスを起動します。

BYOM DB インスタンスの作成 (AWS CLI)

`--license-model bring-your-own-media` を指定して `create-db-instance` コマンドを使用します。

```
aws rds create-db-instance \
    --engine sqlserver-ee \
    --engine-version 17.00.4045.5.v1 \
    --license-model bring-your-own-media \
    --db-instance-identifier my-byom-instance \
    --db-instance-class db.m7i.xlarge \
    --master-username admin \
    --master-user-password <password> \
    --allocated-storage 200 \
    --db-subnet-group-name my-subnet-group \
    --vpc-security-group-ids sg-0123456789abcdef0
```

BYOM DB インスタンスの検証

インスタンスを作成したら、BYOM ライセンスモデルを使用していることを確認します。

```
aws rds describe-db-instances \
    --db-instance-identifier my-byom-instance
```

レスポンスで、次の値を確認します。

- `"LicenseModel": "bring-your-own-media"` – インスタンスは独自の SQL Server ライセンスを使用します。

- `"Engine": "sqlserver-ee"` – 正しいエンジンエディション。
- `"EngineVersion": "17.00.4045.5.v1"` – ターゲットエンジンのバージョン。

考慮事項

- `--license-model` を指定せずに DB インスタンスを作成する既存のオートメーションがある場合、そのエンジンバージョンに対して BYOM エンジンバージョンを作成すると、それらのワークフローは失敗します。BYOM エンジンバージョンをアクティブ化する前に、オートメーションを更新して `--license-model` パラメータを含めます。

RDS for SQL Server の BYOM DB インスタンスをアップグレードする

RDS for SQL Server の BYOM DB インスタンスにマイナーエンジンバージョンアップグレードを適用できます。マイナーバージョンアップグレードには、セキュリティパッチ、バグ修正、パフォーマンスの向上を含む Microsoft の累積更新が含まれます。

ライセンス込みインスタンスとは異なり、BYOM インスタンスをアップグレードするには、アップグレードを開始する前に BYOM エンジンバージョンが使用可能である必要があります。

有効なアップグレードターゲットの特定

アップグレードする前に、現在のバージョンで有効なアップグレードターゲットとして利用可能なエンジンバージョンを特定します。

`describe-db-engine-versions` コマンドを使用して、有効なマイナーバージョンアップグレードターゲットを一覧表示します。

例:

```
aws rds describe-db-engine-versions \
    --engine sqlserver-ee \
    --engine-version 15.00.4440.1.v1 \
    --include-all \
    --query "DBEngineVersions[?DBEngineMediaType=='Customer
 Provided'].ValidUpgradeTarget[] | [?IsMajorVersionUpgrade==\`false\`]"
```

サンプル出力。

```
[
    {
        "Engine": "sqlserver-ee",
        "EngineVersion": "15.00.4455.2.v1",
        "AutoUpgrade": false,
        "IsMajorVersionUpgrade": false
    },
    .....
    {
        "Engine": "sqlserver-ee",
        "EngineVersion": "15.00.4465.1.v1",
        "AutoUpgrade": false,
        "IsMajorVersionUpgrade": false
```

```
    }
]
```

BYOM バージョンアップグレードの適用 (AWS CLI)

バージョンアップグレードを開始する前に、手動スナップショットを作成します。Amazon RDS は自動スナップショットを作成しますが、手動スナップショットを使用することで、既知の復元ポイントを確保できます。

例:

```
aws rds modify-db-instance \
    --db-instance-identifier my-byom-instance \
    --engine-version 15.00.4465.1.v1 \
    --no-apply-immediately
```

> **Note**
>
> `--no-apply-immediately` を使用すると、Amazon RDS は次のメンテナンスウィンドウ中に変更を適用します。今すぐ変更を適用するには、`--apply-immediately` を使用します。詳細については、「スケジュール変更設定の使用」を参照してください。

アップグレードを検証する

アップグレードが完了したら、DB インスタンスの新しいエンジンバージョンを確認します。

例:

```
aws rds describe-db-instances \
    --db-instance-identifier my-byom-instance \
    --query "DBInstances[*].
[DBInstanceIdentifier,EngineVersion,DBInstanceStatus,LicenseModel]"
```

```
[
    [
        "my-byom-instance",
        "15.00.4465.1.v1",
        "available",
        "bring-your-own-media"
```

```
    ]
]
```

次の値を確認します。

- `EngineVersion` はターゲットバージョンと一致します。
- `DBInstanceStatus` は `available`。
- `LicenseModel` は `bring-your-own-media`。

ライセンス込みのインスタンスを Bring Your Own Media に変換する

既存の RDS for SQL Server インスタンスをライセンス込みモデルから BYOM に変換できます。これにより、AWS に含まれるライセンス料金を支払う代わりに、独自の SQL Server ライセンスを使用できるようになります。

> **Note**
>
> インスタンスの現在のエンジンバージョンに対応する BYOM エンジンバージョンが、`available` ステータスである必要があります。

`modify-db-instance` コマンドを使用してライセンスモデルを変更します。

```
aws rds modify-db-instance \
    --db-instance-identifier DB_INSTANCE_IDENTIFIER \
    --license-model bring-your-own-media \
    --no-apply-immediately
```

> **Note**
>
> `--no-apply-immediately` を使用すると、Amazon RDS は次のメンテナンスウィンドウ中に変更を適用します。今すぐ変更を適用するには、`--apply-immediately` を使用します。詳細については、「スケジュール変更設定の使用」を参照してください。

DB インスタンスの現在のライセンスモデルを確認するには、`describe-db-instances` コマンドを使用してレスポンスの `LicenseModel` フィールドを確認します。変更が進行中の場合は、保留中のライセンスモデルの `PendingModifiedValues` フィールドを確認します。変更が完了する

と、LicenseModel フィールドには新しい値が反映されます。詳細については、DB インスタンスの説明についてのページを参照してください。

考慮事項

- BYOM からライセンス込みへの変換はサポートされていません。
- ライセンス込み (LI) インスタンスにリードレプリカがある場合は、ライセンスモデルを BYOM に変換する前に削除し、後で再作成します。新しいレプリカは BYOM ライセンスモデルを自動的に継承します。
- LI DB インスタンスが BYOM でサポートされていないインスタンスクラス (db.m5 や db.r6i など) で実行されている場合、BYOM に直接変換することはできません。まず、インスタンスをサポートされているインスタンスクラス (第 7 世代以降) に変更し、ライセンスモデルを別の変更オペレーションとして変換します。

BYOM DB インスタンスと AWS License Manager の統合

BYOM モデルで SQL Server のライセンス使用状況を簡単にモニタリングできるように、AWS License Manager は RDS for SQL Server と統合されています。License Manager は、仮想コア (vCPU) に基づいた RDS for SQL Server エンジンエディションの追跡をサポートしています。また AWS Organizations で License Manager を使用して、すべての組織アカウントを一元管理することもできます。

RDS for SQL Server の製品情報フィルターは、次の表のとおりです。

フィルタ	名前	説明
エンジンのエディション	`sqlserver-ee`	SQL Server Enterprise Edition (EE)
エンジンのエディション	`sqlserver-se`	SQL Server Standard Edition (SE)

SQL Server DB インスタンスのライセンス使用状況を追跡するには、AWS License Manager を使用してセルフマネージドライセンスを作成できます。この場合、製品情報フィルターに一致する SQL Server リソースは、セルフマネージドライセンスに自動的に関連付けられます。SQL Server DB インスタンスの検出には最長で 24 時間かかることがあります。AWS Resource Access Manager を使用して、アカウント間でライセンスを追跡することもできます。

AWS License Manager でセルフマネージドライセンスを作成するには

SQL Server DB インスタンスのライセンス使用状況を追跡するセルフマネージドライセンスを AWS License Manager で作成するには

1. https://console.aws.amazon.com/license-manager/ に移動します。
2. [セルフマネージドライセンスを作成] を選択します。

   手順については、「AWS License Manager ユーザーガイド」の「セルフマネージドライセンスを作成する」を参照してください。

   [Product Information (製品情報)] パネルで [RDS Product Information Filter (RDS 製品情報フィルター)] のルールを追加します。リソースタイプを選択するときは、[SQL Server データベース] を選択します。

詳細については、「AWS License Manager API リファレンス」の「ProductInformation」を参照してください。

3. (クロスアカウント追跡のみ) セルフマネージドライセンスを任意の AWS アカウントと共有する場合、または AWS Organizations を通じて共有する場合に AWS Resource Access Manager を使用します。詳細については、「AWS リソースの共有」を参照してください。

AWS CLI を使用してセルフマネージドライセンスを作成するには

AWS CLI を使用してセルフマネージドライセンスを作成するには、`create-license-configuration` コマンドを呼び出します。`--cli-input-json` または `--cli-input-yaml` パラメータを使用して、コマンドにパラメータを渡すことができます。

例

次の例は、SQL Server Enterprise Edition のセルフマネージドライセンスを作成します。

```
aws license-manager create-license-configuration --cli-input-json file://rds-sqlserver-ee.json
```

次に、この例で使用されているサンプルの `rds-sqlserver-ee.json` ファイルを示します。

```
{
    "Name": "rds-sqlserver-ee",
    "Description": "RDS SQL Server Enterprise Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["sqlserver-ee"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

製品情報の詳細については、「AWS License Manager ユーザーガイド」の「リソースインベントリの自動検出」を参照してください。

`--cli-input` パラメータの詳細については、「AWS CLI ユーザーガイド」の「JSON または YAML 入力ファイルからの AWS CLI スケルトンと入力パラメータの生成」を参照してください。

Developer Edition と BYOM の違い

次の表は、RDS for SQL Server の Developer Edition の BYOM と Standard Edition および Enterprise Edition の BYOM を比較したものです。

	Developer Edition BYOM	Standard/Enterprise Edition BYOM
ライセンス要件	ライセンスモビリティは必要ありません。Developer Edition の準備についてのページを参照してください。	ソフトウェアアシュアランスによるライセンスモビリティが必要です。「the section called “BYOM エンジンバージョンの作成と管理”」を参照してください。
累積的な更新のダウンロード	顧客が提供する必要がある	CU ファイルにアクションは必要ありません。CEV の作成時に目的のマイナーエンジンバージョンを選択するだけで、Amazon RDS は適切な CU を自動的にダウンロードします。
マルチ AZ	サポート外	サポート対象
リードレプリカ	サポート外	サポート対象
SQL Server Integration Service (SSIS)	サポート外	サポート対象
SQL Server Reporting Service (SSRS)、SQL Server Analysis Service (SSAS)	サポートされません	サポートされません

BYOM エンジンのバージョンと DB インスタンスのトラブルシューティング

次の表は、RDS for SQL Server の Bring Your Own Media (BYOM) エンジンバージョンと DB インスタンスを作成および管理する際の一般的な問題の診断と解決に役立ちます。

エラーメッセージ / 症状	原因	推奨される修正
`InvalidParameterValue: RDS can't create the custom engine version because the provided <Name>.ISO file doesn't match the expected installer file for engine version 15.00.4435.7.v1`	アップロードされた RTM ファイルが、コマンドで指定された SQL Server メジャーバージョンまたはエディションと一致しません。例えば、2019 エンジンバージョン (15.xx) をターゲットにしながら SQL Server 2022 ISO をアップロードする場合などです。	1. エンジンバージョンのプレフィックスが ISOと一致していることを確認します。(15.xx = SQL Server 2019、16.xx = SQL Server 2022)。2. エンジンパラメータがエディションと一致することを確認します。(`sqlserver-ee` = Enterprise、`sqlserver-se` = Standard)。3. 正しい RTM ファイルを S3 にアップロードします。4. 失敗した BYOM エンジンバージョンが作成された場合は削除します。5. 一致する ISO を使用して `create-custom-db-engine-version` を再試行します。
`CreateCustomDBEngineVersionFault: RDS was unable to access the specified installer file in the S3 bucket.`	RDS が S3 バケットにアクセスできなかったか、指定された S3 ロケーションに ISO ファイルが見つかりませんでした。	1. S3 バケットに RDS サービスアクセスをブロックするポリシーがないことを確認します。2. `--database-installation-files-s3-bucket-name` の S3 バケット名を確認します。3. `--database-installation-files` のファイル名が完全に一致することを確認します。4. バケットが BYOM エンジンバージョン

エラーメッセージ / 症状	原因	推奨される修正
		と同じリージョンにあることを確認します。5. アクセスをブロックする可能性のある SCP または VPC エンドポイントポリシーを確認します。6. 再試行。`create-custom-db-engine-version`
`InvalidParameterValue: RDS can't create the custom engine version because the provided file doesn't match the expected installer file for engine version`	CU ファイルがターゲットエンジンのバージョンと一致しません。	1. ターゲットのマイナーバージョンに適した CU KB 番号を特定します。2. 一致する CU バージョンで `create-custom-db-engine-version` を再試行します。
`InvalidParameterValue: The custom engine version isn't valid. Names can include only lowercase letters (a-z), dashes (-), underscores (_), and periods (.). Specify a valid CEV, refer to https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/custom-cev.html#custom-cev.create, and try again.`	CEV 名にサポートされている文字のみを使用し、BYOM でサポートされている SQL Server のエディションまたはバージョンを使用していることを確認します。	1. サポートされているエンジン `sqlserver-ee` または `sqlserver-se` を使用していることを確認します。2. エンジンのバージョンが SQL Server 2019 (15.xx) または SQL Server 2022 (16.xx) であることを確認します。3. サポートされているエディションとバージョンで再試行します。

エラーメッセージ / 症状	原因	推奨される修正
InvalidCustomDBEngineVersionStateFault: You can't delete custom engine version sqlserver-ee 15.00.4465.1.v1. At least one instance and one backup is using the CEV. Change the version or delete the instances and backups, and try again.	DB インスタンス、スナップショット、またはリードレプリカが、引き続き BYOM バージョンを使用しています。	1. このバージョンを使用しているリソースを特定します。aws rds describe-db-instances --query "DBInstances[?EngineVersion=='<version>']" 。2. スナップショットを特定します。aws rds describe-db-snapshots --query "DBSnapshots[?EngineVersion=='<version>']" 。3. インスタンスを別のバージョンに移行するか、削除します。4. 関連付けられたスナップショットと自動バックアップを削除します。5. 再試行。delete-custom-db-engine-version
InvalidParameterCombination: The LicenseModel parameter is required for configurations with multiple options.	BYOM はこのバージョンでアクティブ化されていますが、--license-model は除外されています。	コマンドに --license-model bring-your-own-media または --license-model license-included を追加します。

エラーメッセージ / 症状	原因	推奨される修正
The combination of engine sqlserver-se and version <version> isn't valid. Enter a valid custom engine version, and try again.	このエラーは、以下のいずれかの場合に発生します。(1) ターゲットエンジンバージョンに BYOM エンジンバージョンが存在しない。(2) ターゲット BYOM エンジンバージョンが incompatible-media または inactive ステータスである。(3) 指定されたエンジンバージョンがサポートされているメジャー/マイナーの組み合わせではない。	サポートされているエンジンバージョンを確認するには、まず describe-db-engine-versions を実行し、続いて create-custom-db-engine-version を使用して BYOM エンジンバージョンを作成します。
InvalidParameterCombination: RDS can't perform the operation because the custom engine version <version> is in incompatible-installation-media status. Choose a custom engine version in one of the following statuses and try again: available, creating, pending-validation, validating.	BYOM エンジンバージョンが検証に失敗した状態です。	失敗した BYOM エンジンバージョンを削除し、正しいインストールメディアを再アップロードして、新しい BYOM エンジンバージョンを作成します。

エラーメッセージ / 症状	原因	推奨される修正
InvalidParameterCombination: RDS does not support creating a DB instance with the following combination: DBInstanceClass=<instance class>, Engine=<Edition>, EngineVersion=<version>, LicenseModel=bring-your-own-media. For supported combinations of instance class and database engine version, see the documentation.	指定されたインスタンスタイプは BYOM ではサポートされていません。	インスタンスクラスが BYOM をサポートしているかどうかを確認するには、describe-orderable-db-instance-options API を使用します。aws rds describe-orderable-db-instance-options --engine sqlserver-ee --engine-version <version> --db-instance-class <instance-class> --region <region> --query "OrderableDBInstanceOptions[?LicenseModel=='bring-your-own-media']" 。または、サポートされているインスタンスクラスの Amazon RDS for SQL Server の料金ページにアクセスしてください。

エラーメッセージ / 症状	原因	推奨される修正
InvalidParameterCombination: The combination of engine and version isn't valid. Enter a valid custom engine version, and try again. の。ModifyDBInstance	BYOM インスタンスをアップグレードするには、ターゲットバージョンの BYOM エンジンバージョンが存在し、少なくとも available ステータスである必要があります。以下のいずれかの場合、modify-db-instance 呼び出しは拒否されます。ターゲットエンジンバージョンに BYOM エンジンバージョンが存在しない場合、ターゲット BYOM エンジンバージョンが incompatible-media または inactive ステータスの場合、ターゲットバージョンが ValidUpgradeTarget 配列にない場合。	describe-db-engine-versions コマンドを使用して、有効なマイナーバージョンアップグレードターゲットを一覧表示します。BYOM インスタンスにマイナーバージョンアップグレードを適用する前に、ターゲットバージョンに有効な BYOM エンジンバージョンが存在することを確認します。存在しない場合は、作成して使用可能になるまで待機します。

RDS for SQL Server による Active Directory の操作

RDS for SQL Server DB インスタンスを Microsoft Active Directory (AD) ドメインに参加させることができます。AD ドメインは、AWS 内の AWS マネージド AD で、企業データセンターなど任意の場所のセルフマネージド AD で、AWS EC2 で、またはその他のクラウドプロバイダーでホストすることができます。

セルフマネージド Active Directory および AWS Managed Microsoft AD では、NTLM 認証と Kerberos 認証を使用してドメインユーザーを認証できます。

以下のセクションでは、Amazon RDS 上の Microsoft SQL Server について、セルフマネージド Active Directory と AWS マネージド Active Directory の使用方法について説明します。

トピック

- Amazon RDS for SQL Server DB インスタンスによるセルフマネージド Active Directory の操作
- RDS for SQL Server による AWS Managed Active Directory の操作

Amazon RDS for SQL Server DB インスタンスによるセルフマネージド Active Directory の操作

Amazon RDS for SQL Server は、データセンター内、Amazon EC2、またはその他のクラウドプロバイダーなど、AD がホストされている場所に関係なく、セルフマネージド Active Directory (AD) ドメインとシームレスに統合されます。この統合により、NTLM または Kerberos プロトコルを介した直接ユーザー認証が可能になり、複雑な中間ドメインやフォレストの信頼性が不要になります。RDS SQL Server DB インスタンスに接続すると、認証リクエストは指定された AD ドメインに安全に転送され、Amazon RDS のマネージドデータベース機能を活用しながら、既存の ID 管理構造が維持されます。

トピック

- 利用可能なリージョンとバージョン
- 要件
- 考慮事項
- セルフマネージド Active Directory の設定
- DB インスタンスをセルフマネージド Active Directory に結合する
- セルフマネージド Active Directory ドメイン内の DB インスタンスの管理
- セルフマネージド Active Directory ドメインメンバーシップについて
- セルフマネージド Active Directory のトラブルシューティング
- SQL Server DB インスタンスを復元してからセルフマネージド Active Directory ドメインに追加する

利用可能なリージョンとバージョン

Amazon RDS は、すべての商用 AWS リージョンおよび AWS GovCloud (US) Regions で NTLM を使用するセルフマネージド AD for SQL Server をサポートしています。

要件

RDS for SQL Server DB インスタンスをセルフマネージド AD ドメインに参加させる前に、次の要件を満たしていることを確認してください。

トピック

- オンプレミス AD の設定

- ネットワーク接続の設定
- AD ドメインサービスアカウントの設定
- LDAPS を介した安全な通信の設定

オンプレミス AD の設定

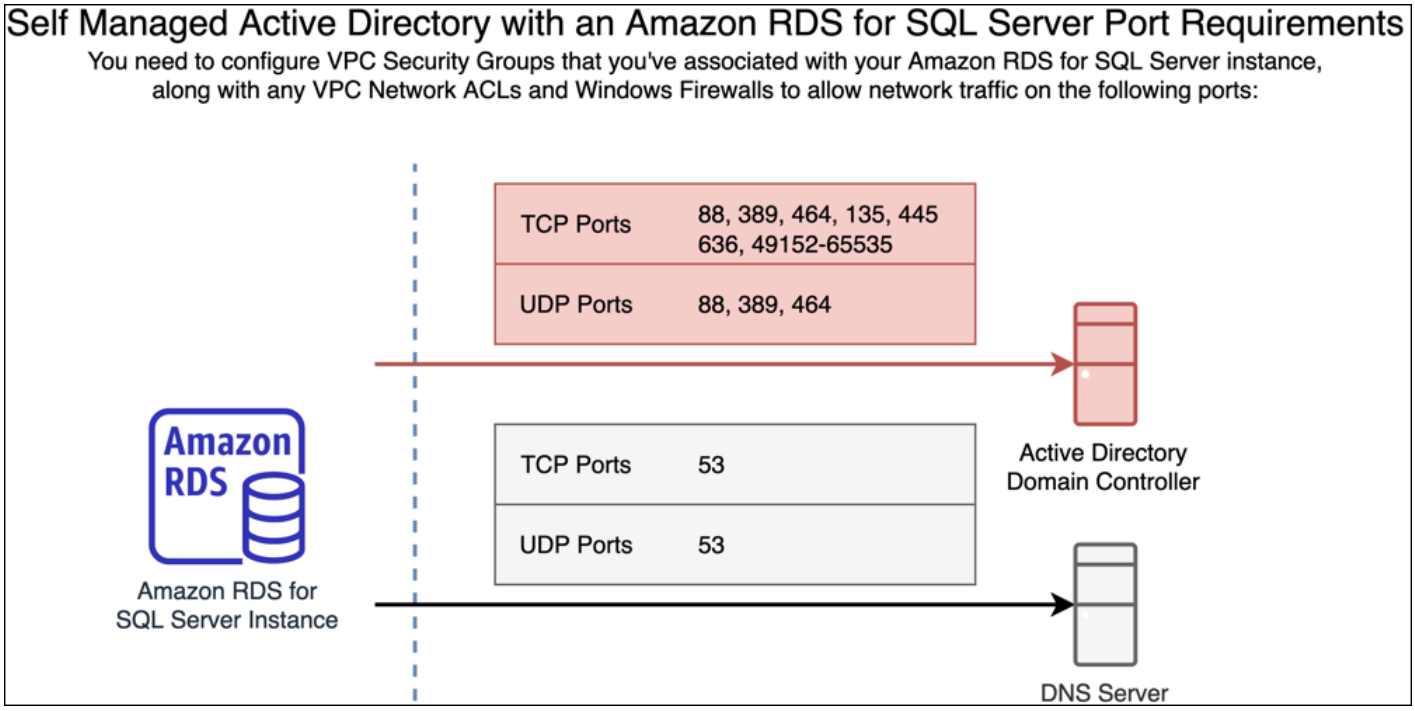
Amazon RDS for SQL Server インスタンスを参加させることができるオンプレミスまたはその他のセルフマネージド Microsoft AD があることを確認してください。オンプレミス AD には以下の設定が必要です。

- AD サイトが定義されている場合、RDS for SQL Server DB インスタンスに関連付けられている VPC 内のサブネットが AD サイトで定義されていることを確認してください。VPC 内のサブネットと他の AD サイトのサブネットの間に競合がないことを確認します。
- AD ドメインコントローラーは、Windows Server 2008 R2 以降のドメイン機能レベルを持ちます。
- AD ドメイン名をシングルラベルドメイン (SLD) 形式にすることはできません。RDS for SQL Server は、SLD ドメインをサポートしていません。
- AD の完全修飾ドメイン名 (FQDN) は 47 文字以内で指定します。

ネットワーク接続の設定

次のネットワーク設定を満たしていることを確認します。

- RDS for SQL Server DB インスタンスを作成する Amazon VPC とセルフマネージド AD の間の接続を設定します。接続は、AWS 直接接続、AWS VPN、VPC ピアリング、または AWS トランジットゲートウェイを使用して設定できます。
- VPC セキュリティグループ については、デフォルトの Amazon VPC のデフォルトセキュリティグループが、コンソールの RDS for SQL Server DB インスタンスに既に追加されています。RDS for SQL Server DB インスタンスを作成するサブネットのセキュリティグループと VPC ネットワーク ACL が、次の図に示す方向でのポート上のトラフィックを許可していることを確認します。



以下の表に、各ポートのロールを示します。

プロトコル	ポート	ロール
TCP / UDP	53	ドメインネームシステム (DNS)
TCP / UDP	88	Kerberos 認証
TCP / UDP	464	パスワードを変更 / 設定する
TCP / UDP	389	Lightweight Directory Access プロトコル (LDAP)
TCP	135	Distributed Computing Environment / End Point Mapper (DCE / EPMAP)
TCP	445	Directory Services SMB ファイル共有

プロトコル	ポート	ロール
TCP	636	TLS/SSL (LDAPS) を介した Lightweight Directory Access Protocol (LDAPS)
TCP	49152 - 65535	RPC 用のエフェメラルポート

- 通常、ドメイン DNS サーバーは AD ドメインコントローラーにあります。この機能を使用するために VPC DHCP オプションセットを設定する必要はありません。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。

> **Important**
>
> VPC ネットワーク ACL を使用している場合は、RDS for SQL Server DB インスタンスからのダイナミックポート (49152-65535) でのアウトバウンドトラフィックも許可する必要があります。これらのトラフィックルールが、各 ADドメインコントローラー、DNS サーバー、および RDS for SQL Server DB インスタンスにもミラーリングされていることを確認します。
>
> VPC セキュリティグループでは、ネットワークトラフィックが開始される方向でのみポートを開く必要がありますが、ほとんどの Windows ファイアウォールとおよび VPC ネットワーク ACL では両方向にポートを開く必要があります。

AD ドメインサービスアカウントの設定

AD ドメインサービスアカウントが次の要件を満たしていることを確認してください。

- コンピュータをドメインに参加させる委任アクセス許可のあるドメインサービスアカウントがセルフマネージド AD ドメインにあることを確認してください。ドメインサービスアカウントは、特定のタスクを実行するアクセス許可を委任されたセルフマネージド AD のユーザーアカウントです。
- ドメインサービスアカウントには、RDS for SQL Server DB インスタンスを参加させる組織単位 (OU) の以下のアクセス許可を委任する必要があります。
  - DNS ホスト名への書き込みを検証する機能
  - サービスプリンシパル名への書き込みを検証する機能
  - コンピュータオブジェクトを作成および削除する

これらは、コンピュータオブジェクトをセルフマネージド AD に接続させるために必要な最小限の許可セットです。詳細については、Microsoft Windows Server ドキュメントの「コンピューターをドメインに参加させようとするとエラーが発生する」を参照してください。

- Kerberos 認証を使用するには、AD ドメインサービスアカウントにサービスプリンシパルネーム (SPN) と DNS アクセス許可を提供する必要があります。
  - 書き込み SPN: RDS for SQL Server DB インスタンスに参加する必要がある OU の AD ドメインサービスアカウントに書き込み SPN アクセス許可を委任します。このアクセス許可は、検証済みの書き込み SPN とは異なります。
  - DNS アクセス許可: ドメインコントローラーのサーバーレベルで DNS マネージャーの AD ドメインサービスアカウントに次のアクセス許可を付与します。
    - コンテンツを一覧表示する
    - すべてのプロパティを読み取る
    - 読み取りアクセス許可

> **Important**
>
> DB インスタンスの作成後に RDS for SQL Server が組織単位に作成したコンピュータオブジェクトを移動しないでください。関連するオブジェクトを移動すると、RDS for SQL Server DB インスタンスが誤って設定される原因となります。Amazon RDS によって作成されたコンピュータオブジェクトを移動する必要がある場合は、ModifyDbInstance RDS API オペレーションを使用して、コンピュータオブジェクトの目的の場所にドメインパラメータを変更します。

LDAPS を介した安全な通信の設定

RDS がドメインコントローラー内のコンピュータオブジェクトと SPN をクエリしてアクセスするには、LDAPS 経由の通信が推奨されます。安全な LDAP を使用するには、安全な LDAPS の要件を満たす有効な SSL 証明書をドメインコントローラーで使用します。有効な SSL 証明書がドメインコントローラーに存在しない場合、RDS for SQL Server DB インスタンスはデフォルトで LDAP を使用します。証明書の有効性の詳細については、「Requirements for an LDAPS certificate」を参照してください。

考慮事項

RDS for SQL Server DB インスタンスをセルフマネージド AD に追加するときは、次の点を考慮してください。

- DB インスタンスは、AD ドメインのタイムサーバーではなく、AWS の NTP サービスと同期します。AD ドメイン内のリンクされた SQL Server インスタンス間のデータベース接続の場合、SQL 認証のみ実行でき、Windows 認証は実行できません。
- セルフマネージド AD ドメインのグループポリシーオブジェクト設定は RDS for SQL Server インスタンスには伝播されません。

セルフマネージド Active Directory の設定

セルフマネージド AD を設定するには、次の手順を実行してください。

トピック

- ステップ 1: AD に組織単位を作成する
- ステップ 2: AD に AD ドメインサービスアカウントを作成する
- ステップ 3: AD ドメインサービスアカウントに制御を委任する
- ステップ 4: AWS KMS キーを作成する
- ステップ 5: AWS シークレットを作成する

ステップ 1: AD に組織単位を作成する

> **Important**
>
> セルフマネージド AD ドメインに参加した RDS for SQL Server DB インスタンスを所有する AWS アカウントに、専用の OU とその OU を対象とするサービス認証情報を作成することをお勧めします。OU とサービス認証情報を専用にすることで、アクセス許可の競合を回避し、最小特権の原則に従うことができます。

AD に OU を作成するには

1. ドメイン管理者として AD ドメインに接続します。
2. [Active Directory ユーザーとコンピューター] を開き、OU を作成するドメインを選択します。

3. ドメインを右クリックして、[新規] を選択し、次に [組織単位] を選択します。
4. OU の名前を入力します。
5. [コンテナを誤って削除しないように保護する] ボックスはオンのままにしてください。
6. [OK] をクリックします。新しい OU がドメインの下に表示されます。

ステップ 2: AD に AD ドメインサービスアカウントを作成する

ドメインサービスアカウントの認証情報は AWS Secrets Manager のシークレットに使用されます。

AD に AD ドメインサービスアカウントを作成するには

1. [Active Directory ユーザーとコンピューター] を開き、ユーザーを作成するドメインと OU を選択します。
2. [ユーザー] オブジェクトを右クリックして、[新規] を選択し、[ユーザー] を選択します。
3. ユーザーの名、姓、およびログオン名を入力します。[次へ] をクリックします。
4. ユーザーのパスワードを入力します。[ユーザーは次回のログイン時にパスワードを変更する必要がある] を選択しないでください。[アカウントは無効です] を選択しないでください。[次へ] をクリックします。
5. [OK] をクリックします。新しいユーザーがドメインの下に表示されます。

ステップ 3: AD ドメインサービスアカウントに制御を委任する

ドメイン内の AD ドメインサービスアカウントに制御を委任するには

1. [Active Directory ユーザーとコンピューター] MMC スナップインを開き、ユーザーを作成するドメインを選択します。
2. 前に作成した OU を右クリックして、[制御を委任] を選択します。
3. [コントロールウィザードの委任] で、[次へ] を選択します。
4. [ユーザーまたはグループ] セクションで、[追加] をクリックします。
5. [ユーザー、コンピューター、またはグループの選択] セクションで、作成した AD ドメインサービスアカウントを入力し、[名前を確認] をクリックします。AD ドメインサービスアカウントのチェックが成功したら、[OK] をクリックします。
6. [ユーザーまたはグループ] セクションで、AD ドメインサービスアカウントが追加されたことを確認し、[次へ] をクリックします。

7. [委任するタスク] セクションで、[委任するカスタムタスクを作成] を選択し、[次へ] をクリックします。
8. [Active Directory オブジェクトタイプ] セクションで:
   a. [フォルダ内の次のオブジェクトのみ] を選択します。
   b. [コンピュータオブジェクト] を選択します。
   c. [このフォルダに選択したオブジェクトを作成] を選択します。
   d. [このフォルダ内の選択したオブジェクトを削除] を選択し、[次へ] をクリックします。
9. [アクセス許可] セクションで:
   a. [全般] を選択したままにします。
   b. [DNS ホスト名への検証済み書き込み] を選択します。
   c. [サービスプリンシパル名への検証済み書き込み] を選択し、[次へ] をクリックします。
   d. Kerberos 認証を有効にするには、[プロパティ固有] を選択し、リストから [書き込み servicePrincipalName] を選択します。
10. [コントロールウィザードの委任の完了] で設定を確認し、[完了] をクリックします。
11. Kerberos 認証の場合は、DNS Manager を開き、[サーバー] プロパティを開きます。
    a. Windows ダイアログボックスで、`dnsmgmt.msc` と入力します。
    b. [セキュリティ] タブの下に AD ドメインサービスアカウントを追加します。
    c. [読み取り] アクセス許可を選択し、変更を適用します。

ステップ 4: AWS KMS キーを作成する

KMS キーは、AWS シークレットの暗号化に使用されます。

AWS KMS キーを作成するには

> **Note**
>
> [暗号化キー] として、AWS デフォルトの KMS キーを使用しないでください。AWS KMS キーは、セルフマネージド AD に参加させる RDS for SQL Server DB インスタンスを含んでいるのと同じ AWS アカウントに作成してください。

1. AWS KMS コンソールで、[キーの作成] を選択します。

2. [キーの種類] として、[対称] を選択します。
3. [キーの使用方法] として、[暗号化と復号化] を選択します。
4. [Advanced options (詳細オプション)] の場合:
   a. [キーマテリアルのオリジン] として、[KMS] を選択します。
   b. [リージョナリティ] として、[単一リージョンキー] を選択し、[次へ] をクリックします。
5. [エイリアス] に、KMS キーの名前を指定します。
6. (オプション) [説明] に、KMS キーの説明を入力します。
7. (オプション) [タグ] に、KMS キーのタグを指定し、[次へ] をクリックします。
8. [キー管理者] として、IAM ユーザーの名前を入力して選択します。
9. [キーの削除] で、[キー管理者にこのキーの削除を許可する] のボックスをオンのままにして、[次へ] をクリックします。
10. [キーユーザー] として、前のステップと同じ IAM ユーザーを指定して選択します。[次へ] をクリックします。
11. 設定を確認します。
12. [キーポリシー] で、以下をポリシー [ステートメント] に含めます。

```
{
    "Sid": "Allow use of the KMS key on behalf of RDS",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "rds.amazonaws.com"
        ]
    },
    "Action": "kms:Decrypt",
    "Resource": "*"
}
```

13. [Finish] をクリックします。

ステップ 5: AWS シークレットを作成する

シークレットを作成する

> **Note**
>
> シークレットは、セルフマネージド AD に参加させる RDS for SQL Server DB インスタンスを含んでいるのと同じ AWS アカウントに作成してください。

1. AWS Secrets Manager で、[新しいシークレットを保存する] を選択します。
2. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
3. [キーと値のペア] として、次の 2 つのキーを追加します。
   a. 最初のキーには、`SELF_MANAGED_ACTIVE_DIRECTORY_USERNAME` と入力します。
   b. 最初のキーの値には、AD ユーザーのユーザー名 (ドメインプレフィックスなし) のみを入力します。ドメイン名を含めないでください。含めると、インスタンスの作成が失敗します。
   c. 2 番目のキーとして、`SELF_MANAGED_ACTIVE_DIRECTORY_PASSWORD` と入力します。
   d. 2 番目のキーの値には、ドメインの AD ユーザー用に作成したパスワードを入力します。
4. [暗号化キー] として、前のステップで作成した KMS キーを入力し、[次へ] をクリックします。
5. [シークレット名] として、後でシークレットを見つけやすい、わかりやすい名前を入力します。
6. (オプション) [説明] として、シークレット名の説明を入力します。
7. [リソースアクセス許可] として、[編集] をクリックします。
8. 以下のポリシーをアクセス許可ポリシーに追加します。

> **Note**
>
> Confused Deputy Problem (混乱した代理の問題) を回避するために、ポリシーの `aws:sourceAccount` および `aws:sourceArn` 条件を使用することをお勧めします。`aws:sourceAccount` の AWS アカウント と `aws:sourceArn` の RDS for SQL Server DB インスタンス ARN を使用します。詳細については、「サービス間での混乱した代理問題の防止」を参照してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Principal":
            {
                "Service": "rds.amazonaws.com"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*",
            "Condition":
            {
                "StringEquals":
                {
                    "aws:sourceAccount": "123456789012"
                },
                "ArnLike":
                {
                    "aws:sourceArn": "arn:aws:rds:us-
west-2:123456789012:db:*"
                }
            }
        }
    ]
}
```

9. [保存] をクリックし、[次へ] をクリックします。
10. [ローテーションの設定] は、デフォルト値のままにして、[次へ] を選択します。
11. シークレットの設定を確認し、[保存] をクリックします。
12. 作成したシークレットを選択し、[シークレット ARN] の値をコピーします。これを次のステップで使用して、セルフマネージド Active Directory をセットアップします。

DB インスタンスをセルフマネージド Active Directory に結合する

RDS for SQL Server DB インスタンスをセルフマネージド AD に結合するには、次の手順に従います。

ステップ 1: SQL Server DB インスタンスを作成または変更する

コンソール、CLI、または RDS API を使用して、RDS for SQL Server DB インスタンスをセルフマネージド AD ドメインに関連付けることができます。これは、次のいずれかの方法で行うことができます。

- コンソール、create-db-instance CLI コマンド、または CreateDBInstance RDS API オペレーションを使用して、新しい SQL Server DB インスタンスを作成します。

  手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance CLI コマンド、または ModifyDBInstance RDS API オペレーションを使用して、既存の SQL Server DB インスタンスを変更します。

  手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、restore-db-instance-from-db-snapshot CLI コマンド、または RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから SQL Server DB インスタンスを復元します。

  手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、または RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、SQL Server DB インスタンスをポイントインタイムに復元します。

  手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

AWS CLI を使用する場合は、DB インスタンスが、作成したセルフマネージド AD ドメインを使用できるように、以下のパラメータが必要です。

- `--domain-fqdn` パラメータには、セルフマネージド AD の完全修飾ドメイン名 (FQDN) を使用します。
- `--domain-ou` パラメータには、セルフマネージド AD で作成した OU を使用します。

- `--domain-auth-secret-arn` パラメータには、前のステップで作成した[シークレット ARN] の値を使用します。
- `--domain-dns-ips` パラメータには、セルフマネージド AD の DNS サーバーのプライマリ IPv4 アドレスとセカンダリ IPv4 アドレスを使用します。セカンダリ DNS サーバーの IP アドレスがない場合は、プライマリ IP アドレスを 2 回入力します。

次の CLI コマンドの例は、セルフマネージド AD ドメインを使用して RDS for SQL Server DB インスタンスを作成、変更、削除する方法を示しています。

> **Important**
>
> DB インスタンスを変更してセルフマネージド AD ドメインに参加させたり、削除したりする場合、変更を有効にするには DB インスタンスを再起動する必要があります。変更をすぐに適用するか、次のメンテナンスウィンドウまで待つかを選択できます。[すぐに適用] オプションを選択すると、シングル AZ DB インスタンスのダウンタイムが発生します。マルチ AZ DB インスタンスは、再起動を完了する前にフェイルオーバーを実行します。詳細については、「スケジュール変更設定の使用」を参照してください。

次の CLI コマンドは、新しい RDS for SQL Server DB インスタンスを作成し、それをセルフマネージド AD ドメインに参加させます。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier my-DB-instance \
    --db-instance-class db.m5.xlarge \
    --allocated-storage 50 \
    --engine sqlserver-se \
    --engine-version 15.00.4043.16.v1 \
    --license-model license-included \
    --master-username my-master-username \
    --master-user-password my-master-password \
    --domain-fqdn my_AD_domain.my_AD.my_domain \
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier my-DB-instance ^
    --db-instance-class db.m5.xlarge ^
    --allocated-storage 50 ^
    --engine sqlserver-se ^
    --engine-version 15.00.4043.16.v1 ^
    --license-model license-included ^
    --master-username my-master-username ^
    --master-user-password my-master-password ^
    --domain-fqdn my-AD-test.my-AD.mydomain ^
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \ ^
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

次の CLI コマンドは、セルフマネージド AD ドメインを使用するように既存の RDS for SQL Server DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-DB-instance \
    --domain-fqdn my_AD_domain.my_AD.my_domain \
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain \
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" \
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-DBinstance ^
    --domain-fqdn my_AD_domain.my_AD.my_domain ^
    --domain-ou OU=my-AD-test-OU,DC=my-AD-test,DC=my-AD,DC=my-domain ^
    --domain-auth-secret-arn "arn:aws:secretsmanager:region:account-number:secret:my-
AD-test-secret-123456" ^
    --domain-dns-ips "10.11.12.13" "10.11.12.14"
```

次の CLI コマンドは、セルフマネージド AD ドメインから RDS for SQL Server DB インスタンスを削除します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-DB-instance \
    --disable-domain
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-DB-instance ^
    --disable-domain
```

ステップ 2: Kerberos または NTLM 認証の使用

NTLM 認証

各 Amazon RDS DB インスタンスにはエンドポイントがあり、各エンドポイントには DB インスタンスの DNS 名とポート番号があります。SQL クライアントアプリケーションを使用して DB インスタンスに接続するには、DB インスタンスの DNS 名とポート番号が必要です。NTLM 認証を使用して認証するには、マルチ AZ 配置を使用している場合、RDS エンドポイントまたはリスナーエンドポイントに接続する必要があります。

計画されたデータベースメンテナンス時や計画外のサービス中断時に、Amazon RDS は自動的に最新のセカンダリデータベースにフェイルオーバーするため、オペレーションは手動の介入なしで速やかに再開できます。プライマリインスタンスおよびセカンダリインスタンスは、同じエンドポイントを使用し、エンドポイントの物理ネットワークアドレスは、フェイルオーバープロセスの一環としてセカンダリに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。

Kerberos 認証

RDS for SQL Server の Kerberos ベースの認証では、特定のサービスプリンシパル名 (SPN) に接続する必要があります。ただし、フェイルオーバーイベント後、アプリケーションは新しい SPN を認識しない場合があります。これに対処するために、RDS for SQL Server には Kerberos ベースのエンドポイントが用意されています。

Kerberos ベースのエンドポイントは、特定の形式に従います。RDS エンドポイントが *rds-instance-name*.*account-region-hash*.*aws-region*.rds.amazonaws.com である場合、対応する Kerberos ベースのエンドポイントは *rds-instance-name*.*account-region-hash*.*aws-region*.awsrds.*fully qualified domain name (FQDN)* になります。

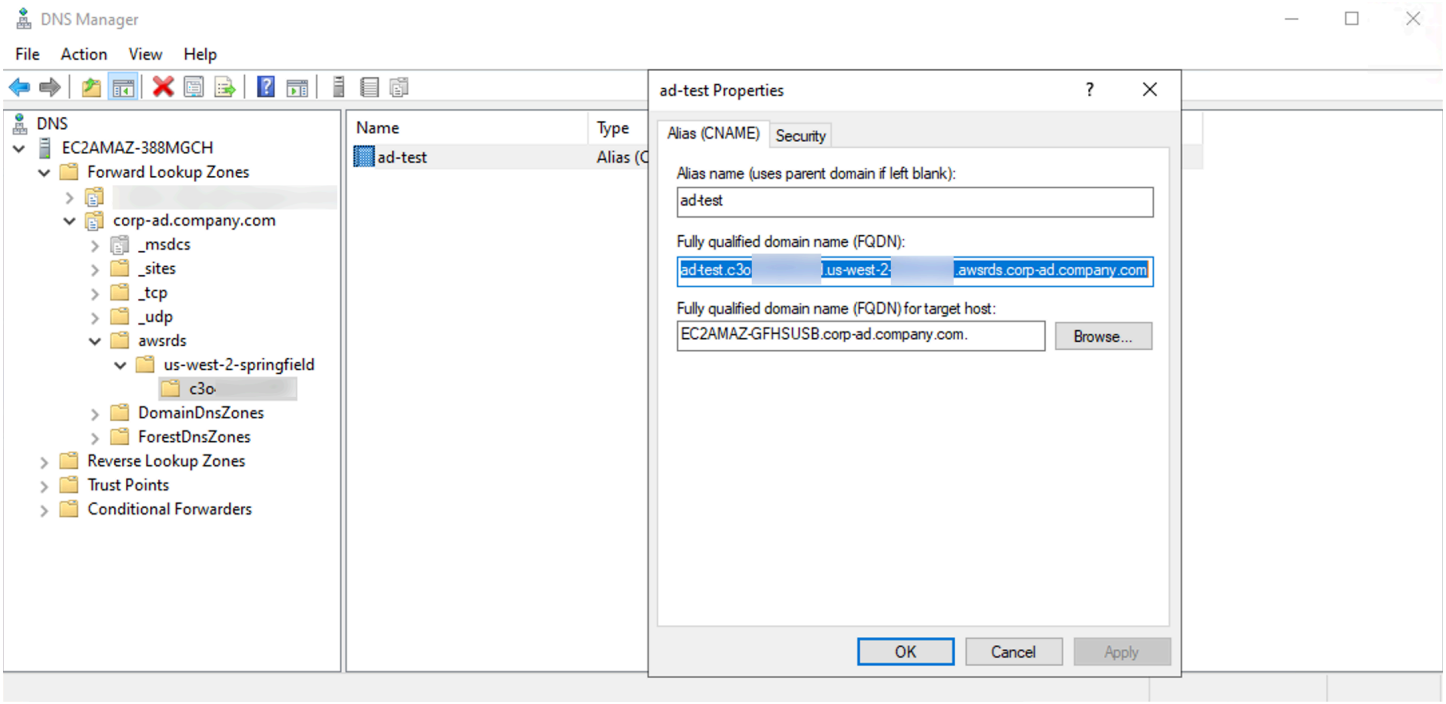
例えば、RDS エンドポイントが ad-test.cocv6zwtircu.us-east-1.rds.amazonaws.com で、ドメイン名が corp-ad.company.com である場合、Kerberos ベースのエンドポイントは ad-test.cocv6zwtircu.us-east-1.awsrds.corp-ad.company.com になります。

この Kerberos ベースのエンドポイントを使用すると、プライマリ SQL Server インスタンスの新しい SPN を指すようにエンドポイントが自動的に更新されるため、フェイルオーバーイベント後でも、Kerberos を使用して SQL Server インスタンスで認証できます。

CNAME の検索

CNAME を検索するには、ドメインコントローラーに接続し、[DNS マネージャー] を開きます。[前方参照ゾーン] と FQDN に移動します。

awsrds、aws-region、および [アカウントとリージョン固有のハッシュ] をナビゲートします。



リモートクライアントから CNAME を接続した後に NTLM 接続が返された場合は、必要なポートが許可リストに登録されているかどうかを確認します。

接続で Kerberos を使用しているかどうかを確認するには、次のクエリを実行します。

```
SELECT net_transport, auth_scheme
    FROM sys.dm_exec_connections
    WHERE session_id = @@SPID;
```

Kerberos エンドポイントに接続するときにインスタンスが NTLM 接続を返す場合は、ネットワーク設定とユーザー設定を確認します。「ネットワーク接続の設定」を参照してください。

ステップ 3: Windows 認証の SQL Server ログインを作成する

Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同じように SQL Server DB インスタンスに接続します。DB インスタンスはセルフマネージド AD ドメインに参加しているため、SQL Server のログインとユーザーをプロビジョニングできます。これは、セルフマネージド AD ドメインの AD ユーザーとグループユーティリティから行います。データベースへのアクセス許可は、これらの Windows ログインに付与され無効化されている標準の SQL サーバーのアクセス許可によって管理されています。

セルフマネージド AD ドメインサービスアカウントが SQL Server に認証するには、セルフマネージド AD ドメインサービスアカウント、またはそのユーザーが属するセルフマネージド AD グループに、SQL Server Windows ログインが存在する必要があります。これらの SQL Server ログインでアクセスを許可したり取り消したりして、細分化されたアクセスコントロールを処理します。SQL Server ログインを持たないか、またはそのようなログインを持つセルフマネージド AD グループに属していないセルフマネージド AD ドメインサービスアカウントは、SQL Server DB インスタンスにアクセスできません。

セルフマネージド AD SQL Server ログインを作成するには、ALTER ANY LOGIN アクセス許可が必要です。このアクセス許可を持つログインをまだ作成していない場合は、SQL Server 認証を使用して DB インスタンスのマスターユーザーとして接続し、マスターユーザーのコンテキストでセルフマネージド AD SQL Server ログインを作成してください。

次の例のようなデータ定義言語 (DDL) コマンドを実行して、セルフマネージド AD ドメインサービスアカウントまたはグループへの SQL Server ログインを作成できます。

> **Note**
>
> *my_AD_domain**my_AD_domain_user* の形式で Windows 2000 以前のログイン名を使用して、ユーザーまたはグループを指定します。ユーザープリンシパル名 (UPN) を *my_AD_domain_user* @ *my_AD_domain* の形式で使用することはできません。

```
USE [master]
GO
CREATE LOGIN [my_AD_domain\my_AD_domain_user] FROM WINDOWS WITH DEFAULT_DATABASE =
 [master], DEFAULT_LANGUAGE = [us_english];
```

```
GO
```

> **Note**
>
> Active Directory ユーザーを削除し、同じ sAMAccountName で新しいユーザーを作成し、RDS for SQL Server インスタンスで CREATE LOGIN [DOMAIN\username] FROM WINDOWS を実行すると、新しいユーザーの SID ではなく、削除されたユーザーの SID を使用してログインが作成され、Windows 認証が失敗する可能性があります。これは、RDS ホストが OS レベルで名前から SID へのマッピングをキャッシュし、このキャッシュにお客様が直接アクセスできないために発生します。この問題を回避するには、プロビジョニングサイクルごとに一意の sAMAccountName を使用します (タイムスタンプやバージョンサフィックスを追加するなど)。同じ名前を再利用する必要がある場合は、AD ユーザーを再作成した後に DBCC FREESYSTEMCACHE('TokenAndPermUserStore') を実行し、最大 10 分待ってから (この間隔は RDS マネージドインスタンスでは設定できないことから、OS レベルのキャッシュの更新を許可するため)、ログインを作成する前に SUSER_SID('DOMAIN\username') が予期される SID を返すことを確認します。

詳細については、Microsoft Developer Network ドキュメントの「ログインの作成 (Transact-SQL)」を参照してください。

ドメインのユーザー (人およびアプリケーション) は、Windows 認証を使用して、セルフマネージド AD ドメインに参加したクライアントマシンから RDS for SQL Server インスタンスに接続できるようになりました。

セルフマネージド Active Directory ドメイン内の DB インスタンスの管理

コンソール、AWS CLI、または Amazon RDS API を使用して、DB インスタンスおよびセルフマネージド AD ドメインとの関係を管理できます。例えば、DB インスタンスをドメイン内、ドメイン外、またはドメイン間で移動させることができます。

例えば、Amazon RDS API を使用して次を実行できます。

- メンバーシップが失敗したためにセルフマネージドドメインへの参加を再試行するには、ModifyDBInstance API オペレーションを使用して、同じパラメータセットを指定します。
  - --domain-fqdn
  - --domain-dns-ips
  - --domain-ou

  - `--domain-auth-secret-arn`
- セルフマネージドドメインから DB インスタンスを削除するには、`ModifyDBInstance` API オペレーションを使用し、ドメインパラメータとして `--disable-domain` を指定します。
- DB インスタンスを別のセルフマネージドドメインに移動するには、`ModifyDBInstance` API オペレーションを使用し、新しいドメインのドメインパラメータを指定します。
  - `--domain-fqdn`
  - `--domain-dns-ips`
  - `--domain-ou`
  - `--domain-auth-secret-arn`
- 各 DB インスタンスのセルフマネージド AD ドメインメンバーシップを一覧表示するには、DescribeDBInstances API オペレーションを使用します。

セルフマネージド Active Directory ドメインメンバーシップについて

AD の詳細を指定しながら DB インスタンスを作成または変更した後、そのインスタンスはセルフマネージド AD ドメインのメンバーになります。AWS コンソールは、DB インスタンスについて、セルフマネージド Active Directory ドメインメンバーシップのステータスを示します。DB インスタンスのステータスは、以下のいずれかです。

- joined – インスタンスは AD ドメインのメンバーです。
- Joining – インスタンスは、AD ドメインのメンバーになる途中です。
- pending-join (参加保留中) – インスタンスのメンバーシップは保留中です。
- pending-maintenance-join – AWS は、次に予定されているメンテナンスウィンドウ中に、インスタンスを AD ドメインのメンバーにできるよう試みます。
- pending-removal – AD ドメインからのインスタンスの削除は保留中です。
- pending-maintenance-removal – AWS は、次に予定されているメンテナンスウィンドウ中に、AD ドメインからのインスタンスの削除を試みます。
- failed – 設定の問題により、インスタンスは AD ドメインに参加できませんでした。インスタンスの変更コマンドを再発行する前に、設定を確認して修正してください。
- removing – インスタンスをセルフマネージド AD ドメインから削除しています。

> **Important**
>
> セルフマネージド AD ドメインのメンバーになるリクエストは、ネットワーク接続の問題が原因で失敗する場合があります。例えば、DB インスタンスを作成したか、既存のインスタンスを変更したが、DB インスタンスをセルフマネージド AD ドメインのメンバーにする試みが失敗することがあります。この場合、コマンドを再発行して DB インスタンスを作成または変更するか、新しく作成されたインスタンスを変更して、セルフマネージド AD ドメインに参加させます。

セルフマネージド Active Directory のトラブルシューティング

セルフマネージド AD をセットアップまたは変更する際に発生する可能性のある問題は次のとおりです。

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 2 / 0x2	指定されたファイルがシステムで見つかりません。	–domain-ou パラメータで指定された組織単位 (OU) の形式または場所が無効です。AWS Secrets Manager で指定されたドメインサービスアカウントには、OU への参加に必要なアクセス許可がありません。	–domain-ou パラメータを確認してください。ドメインサービスアカウントに OU に対する適切なアクセス許可があることを確認してください。詳細については、「AD ドメインサービスアカウントの設定」を参照してください。
エラー 5 / 0x5	アクセスが拒否されました。	ドメインサービスアカウントのアクセス許可が正しく設定されていないか、コンピュータアカウントがドメインに既に存在しています。	ドメイン内のドメインサービスアカウントのアクセス許可を確認し、RDS コンピュータアカウントがドメイン内で重複していないことを確認します。RDS コンピュータアカウント

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
			の名前は、RDS for SQL Server DB インスタンスで SELECT @@SERVERNAME を実行することで確認できます。マルチ AZ を使用している場合は、フェイルオーバーを使用して再起動し、RDS コンピュータアカウントを再度確認してください。詳細については、「DB インスタンスの再起動」を参照してください。
エラー 87 / 0x57	パラメータが間違っています。	AWS Secrets Manager で指定されたドメインサービスアカウントには、適切なアクセス許可がありません。ユーザープロファイルが壊れている可能性もあります。	ドメインサービスアカウントの要件を確認します。詳細については、「AD ドメインサービスアカウントの設定」を参照してください。
エラー 234 / 0xEA	指定された組織単位 (OU) は存在しません。	–domain-ou パラメータで指定された OU は、セルフマネージド AD に存在しません。	–domain-ou パラメータを確認して、指定した OU がセルフマネージド AD に存在することを確認します。

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 1326 / 0x52E	ユーザー名またはパスワードが正しくありません。	AWS Secrets Manager で指定されたドメインサービスアカウントの認証情報に、不明なユーザー名または不正なパスワードが含まれています。セルフマネージド AD でドメインアカウントが無効になっている場合もあります。	AWS Secrets Manager で指定した認証情報が正しく、ドメインアカウントがセルフマネージド AD で有効になっていることを確認します。
エラー 1355 / 0x54B	指定されたドメインが存在しないか、接続できませんでした。	ドメインがダウンしているか、指定された DNS IP セットにアクセスできないか、指定された FQDN にアクセスできません。	`–domain-dns-ips` および `–domain-fqdn` パラメータが正しいことを確認します。RDS for SQL Server DB インスタンスのネットワーク構成を確認し、セルフマネージド AD にアクセスできることを確認します。詳細については、「ネットワーク接続の設定」を参照してください。
エラー 1772 / 0x6BA	RPC サーバーは使用できません。	AD ドメインの RPC サービスへのアクセスに問題がありました。これはサービスまたはネットワークの問題かもしれません。	RPC サービスがドメインコントローラーで実行されていることと、RDS for SQL Server DB インスタンスから TCP ポート 135 および `49152-65535` にアクセスできることを確認します。

エラーコード	説明	一般的な原因	トラブルシューティングの推奨事項
エラー 1727 / 0x6BF	リモートプロシージャ呼び出しが失敗し、実行されませんでした。	ネットワーク接続の問題またはファイアウォールの制限により、ドメインコントローラーへの RPC 通信がブロックされます。	クロス VPC ドメイン結合を使用する場合は、クロス VPC 通信が VPC ピアリングまたは Transit Gateway のいずれかで正しく設定されていることを確認します。RDS for SQL Server DB インスタンスからドメインで TCP ハイポート 49152-65535 にアクセスできることを確認します。これには、ファイアウォールの潜在的な制限も含まれます。
エラー 2224 / 0x8B0	ユーザーアカウントは既に存在しています。	セルフマネージド AD に追加しようとしているコンピュータアカウントはすでに存在しています。	RDS for SQL Server DB インスタンスで `SELECT @@SERVERNAME` を実行してコンピュータアカウントを特定し、セルフマネージド AD から慎重に削除します。
エラー 2242 / 0x8c2	このユーザーのパスワードは有効期限が切れています。	AWS Secrets Manager で指定されたドメインサービスアカウントのパスワードは有効期限が切れています。	RDS for SQL Server DB インスタンスをセルフマネージド AD に参加させるために使用するドメインサービスアカウントのパスワードを更新します。

DB インスタンスをセルフマネージド Active Directory ドメインに結合すると、ドメインの状態に関連する RDS イベントが表示される場合があります。

```
Unhealthy domain state detected while attempt to verify or
configure your Kerberos endpoint in your domain on
node node_n. message
```

マルチ AZ インスタンスの場合、node1 と node2 の両方でエラーレポートが表示されることがあります。これは、インスタンスの Kerberos 設定がフェイルオーバーする準備ができていないことを示します。フェイルオーバーが発生した場合、Kerberos を使用した認証で問題が発生する可能性があります。設定の問題を解決して、Kerberos のセットアップが有効で最新の状態であることを確認します。マルチ AZ インスタンスの場合、すべてのネットワークとアクセス許可の設定が整っているため、新しいプライマリホストで Kerberos 認証を使用するためのアクションは必要ありません。

シングル AZ インスタンスの場合、node1 はプライマリノードです。Kerberos 認証が期待どおりに動作しない場合は、インスタンスイベントを確認し、設定の問題を解決して、Kerberos 設定が有効で最新の状態であることを確認します。

SQL Server DB インスタンスを復元してからセルフマネージド Active Directory ドメインに追加する

SQL Server DB インスタンスの DB スナップショットまたはポイントインタイムリカバリ (PITR) を復元して、セルフマネージド Active Directory ドメインに追加できます。DB インスタンスが復元されたら、「ステップ 1: SQL Server DB インスタンスを作成または変更する」で説明している手順に従ってインスタンスを変更し、DB インスタンスをセルフマネージド AD ドメインに追加します。

RDS for SQL Server による AWS Managed Active Directory の操作

ユーザーが RDS for SQL Server DB インスタンスに接続する際、AWS Managed Microsoft AD を使用して Windows 認証でユーザーを認証できます。DB インスタンスは、Windows 認証を有効にするために AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD とも呼ばれます) を使用します。ユーザーが、信頼性の高いドメインに接続された SQL Server DB インスタンスを使用して認証を実行すると、Directory Service を使用して作成したドメインディレクトリに認証リクエストが転送されます。

リージョンとバージョンの可用性

Amazon RDS では、Windows Authentication 向けに AWS Managed Microsoft AD のみの使用をサポートしています。RDS は AD Connector の使用をサポートしていません。詳細については次を参照してください:

- のアプリケーションの互換性ポリシーAWS Managed Microsoft AD
- AD Connector のアプリケーションの互換性ポリシー

バージョンおよびリージョンの可用性の詳細については、「RDS for SQL Server を使用した Kerberos 認証」を参照してください。

Windows 認証のセットアップの概要

Amazon RDS は Windows 認証にミックスモードを使用します。この方法では、マスターユーザー (SQL Server DB インスタンスの作成に使用された名前とパスワード) が SQL 認証を使用します。マスターユーザーアカウントは特権を持つ認証情報のため、このアカウントへのアクセスを制限する必要があります。

オンプレミスまたはセルフホスト型の Microsoft Active Directory を使用して Windows 認証を取得するには、フォレストの信頼関係を確立します。信頼は、一方向または双方向にすることができます。Directory Service を使用してフォレストの信頼関係を設定する方法の詳細については、AWS Directory Service 管理ガイドの「信頼関係を作成する場合」を参照してください。

SQL Server DB インスタンスの Windows 認証を設定するには、次のステップに従います。詳細については、「SQL Server DB インスタンスの Windows 認証のセットアップ」を参照してください。

1. AWS Managed Microsoft AD または AWS マネジメントコンソール API のいずれかから Directory Service を使用して、AWS Managed Microsoft AD ディレクトリを作成します。

2. AWS CLI または Amazon RDS API を使用してSQL Server DB インスタンスを作成する場合は、AWS Identity and Access Management (IAM) ロールを作成します。このロールはマネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用し、Amazon RDS によるディレクトリへの呼び出しを許可します。SQL Server DB インスタンスの作成にコンソールを使用している場合、AWS は IAM ロールを作成します。

   ロールによるアクセスを許可するには、AWS Security Token Service (AWS STS) エンドポイントを AWS アカウントの AWS リージョンでアクティベートする必要があります。AWS STS エンドポイントはすべての AWS リージョンでデフォルトでアクティブになっているため、他のアクションを実行することなくエンドポイントを使用することができます。詳細については、IAM ユーザーガイドの 「AWS リージョン での AWS STS の管理」を参照してください。
3. Microsoft Active Directory のツールを使用して、AWS Managed Microsoft AD ディレクトリでユーザーとグループを作成して設定します。Active Directory にユーザーおよびグループを作成する方法の詳細については、AWS Directory Service 管理ガイドの「AWS Managed Microsoft AD でユーザーとグループを管理する」を参照してください。
4. ディレクトリと DB インスタンスを異なる VPC に配置する場合は、クロス VPC トラフィックを有効にします。
5. Amazon RDS を使用して、コンソール、AWS CLI、または Amazon RDS API のいずれかから、新しい SQL Server DB インスタンスを作成します。作成リクエストで、ディレクトリの作成時に生成されたドメイン識別子 (「`d-*`」識別子) と、作成したロールの名称を指定します。DB インスタンスのドメインおよび IAM ロールパラメータを設定して、既存の SQL Server DB インスタンスを Windows 認証を使用するように変更することもできます。
6. Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同じように SQL Server DB インスタンスに接続します。DB インスタンスは AWS Managed Microsoft AD ドメインに参加しているため、ドメイン内の Active Directory ユーザーとグループから SQL Server のログインとユーザーをプロビジョニングできます (これらは、SQL Server の「Windows」ログインとして知られています)。データベースへのアクセス許可は、これらの Windows ログインに付与され無効化されている標準の SQL サーバーのアクセス許可によって管理されています。

Amazon RDS コンソールを使用してドメイン接続された RDS for SQL Server DB インスタンスを作成すると、AWS は自動的に `rds-directoryservice-access-role` IAM ロールを作成します。このロールは、ドメイン接続されたインスタンスを管理するために不可欠であり、以下の操作に必要です。

- ドメイン接続された SQL Server インスタンスの設定変更
- Active Directory 統合設定の管理

- ドメイン結合インスタンスでのメンテナンス操作の実行

> **Important**
>
> `rds-directoryservice-access-role` IAM ロールを削除すると、Amazon RDS コンソールまたは API を使用してドメイン接続された SQL Server インスタンスを変更することはできません。インスタンスを変更しようとすると、次のエラーメッセージが表示されます。You don't have permission to iam:CreateRole. To request access, copy the following text and send it to your AWS administrator.
> このエラーは、Amazon RDS がドメイン接続を管理するためにロールを再作成する必要があるが、必要なアクセス許可がないために発生します。さらに、このエラーは CloudTrail に記録されないため、トラブルシューティングがより困難になる可能性があります。

誤って `rds-directoryservice-access-role` を削除した場合、ドメインに接続された SQL Server インスタンスに変更を加える前に、再作成するための `iam:CreateRole` アクセス許可が必要です。ロールを手動で再作成するには、ロールに `AmazonRDSDirectoryServiceAccess` 管理ポリシーがアタッチされ、RDS サービスがロールを引き受けることを許可する適切な信頼関係があることを確認します。

Kerberos 認証のエンドポイントの作成

Kerberos に基づく認証では、エンドポイントは「お客様が指定したホスト名、ピリオド、省略なしのドメイン名 (FQDN)」の形式である必要があります。次の例は、Kerberos に基づく認証で使用できるエンドポイントの例です。この例では、SQL Server DB インスタンスのホスト名は `ad-test`、ドメイン名は `corp-ad.company.com` です。

```
ad-test.corp-ad.company.com
```

接続で Kerberos が使用されていることを確認するには、次のクエリを実行します。

```
SELECT net_transport, auth_scheme
  FROM sys.dm_exec_connections
 WHERE session_id = @@SPID;
```

SQL Server DB インスタンスの Windows 認証のセットアップ

SQL Server DB インスタンスの Windows 認証をセットアップするには、AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD とも呼ばれます) を使用します。Windows 認証を設定するには、次の手順を実行します。

ステップ 1: AWS Directory Service for Microsoft Active Directory を使用してディレクトリを作成する

Directory Service は完全マネージド型の Microsoft Active Directory を AWS クラウドに作成します。AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって 2 つのドメインコントローラーと 2 つのドメインネームサービス (DNS) サーバーを作成します。ディレクトリサーバーは、 VPC 内の 2 つの異なるアベイラビリティーゾーンの 2 つのサブネットで作成されています。この冗長性により、障害が発生してもディレクトリがアクセスできるようになります。

AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって自動的に以下のタスクを実行します。

- VPC 内に Microsoft Active Directory を設定します。
- 「Admin」のユーザー名と指定されたパスワードを使用して、ディレクトリ管理者アカウントを作成します。このアカウントはディレクトリの管理に使用します。
- ディレクトリコントローラー用のセキュリティグループを作成する。

AWS Directory Service for Microsoft Active Directory を立ち上げると、AWS は組織単位 (OU) を作成します。OU にはディレクトリのオブジェクトがすべて含まれています。この OU はドメインルートにあります。OU にはディレクトリを作成する際に入力した NetBIOS 名があります。ドメインルートは AWS が所有し、管理します。

AWS Managed Microsoft AD ディレクトリに作成された管理者アカウントには、次のような、OU で最も一般的な管理業務用のアクセス権限があります。

- ユーザー、グループ、およびコンピュータを作成、更新、または削除する
- ファイルやプリントサーバーなどのドメインにリソースを追加して、追加したリソースへのアクセス許可を OU のユーザーとグループに割り当てる。
- 追加の OU やコンテナを作成する。
- 権限を委譲する。
- グループポリシーを作成し、リンクする。

- 削除されたオブジェクトを Active Directory のごみ箱から元に戻す。
- Active Directory Web Service で AD と DNS Windows PowerShell モジュールを実行する。

管理者アカウントには、ドメイン全体に関係するアクティビティを実行する権限もあります。

- DNS 設定 (レコード、ゾーン、フォワーダーの追加、削除、または更新) を管理する。
- DNS イベントログを参照する。
- セキュリティイベントログを参照する。

AWS Managed Microsoft AD でディレクトリを作成するには

1. Directory Service コンソールのナビゲーションペインで、[ディレクトリ]、[ディレクトリの設定] の順に選択します。
2. を選択します。。AWS Managed Microsoft ADこれは Amazon RDS 用に現在サポートされている唯一のオプションです。
3. [Next (次へ)] を選択します。
4. [ディレクトリ情報の入力] ページに、以下の情報を指定します。

   エディション

   目的の要件を満たすエディションを選択します。

   ディレクトリの DNS 名

   ディレクトリの完全修飾名。例: `corp.example.com`。47 文字を超える名前は SQL Server でサポートされていません。

   ディレクトリの NetBIOS 名

   ディレクトリの短縮名 (例: `CORP`)。

   ディレクトリの説明

   必要に応じて、ディレクトリの説明。

   Admin パスワード

   ディレクトリ管理者のパスワードです。ディレクトリの作成プロセスでは、ユーザー名「Admin」とこのパスワードを使用して管理者アカウントが作成されます。

ディレクトリ管理者のパスワードに「admin」の単語を含めることはできません。パスワードは大文字と小文字を区別し、8–64 文字にします。また、以下の 4 つのカテゴリうち 3 つから少なくとも 1 文字を含める必要があります。

- 小文字 (a〜z)
- 大文字 A〜Z
- 数字 (0〜9)
- アルファベットと数字以外の文字 (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

パスワードを確認

管理者のパスワードをもう一度入力します。

5. [Next (次へ)] を選択します。
6. [VPC とサブネットの選択] ページで、以下の情報を指定します。

VPC

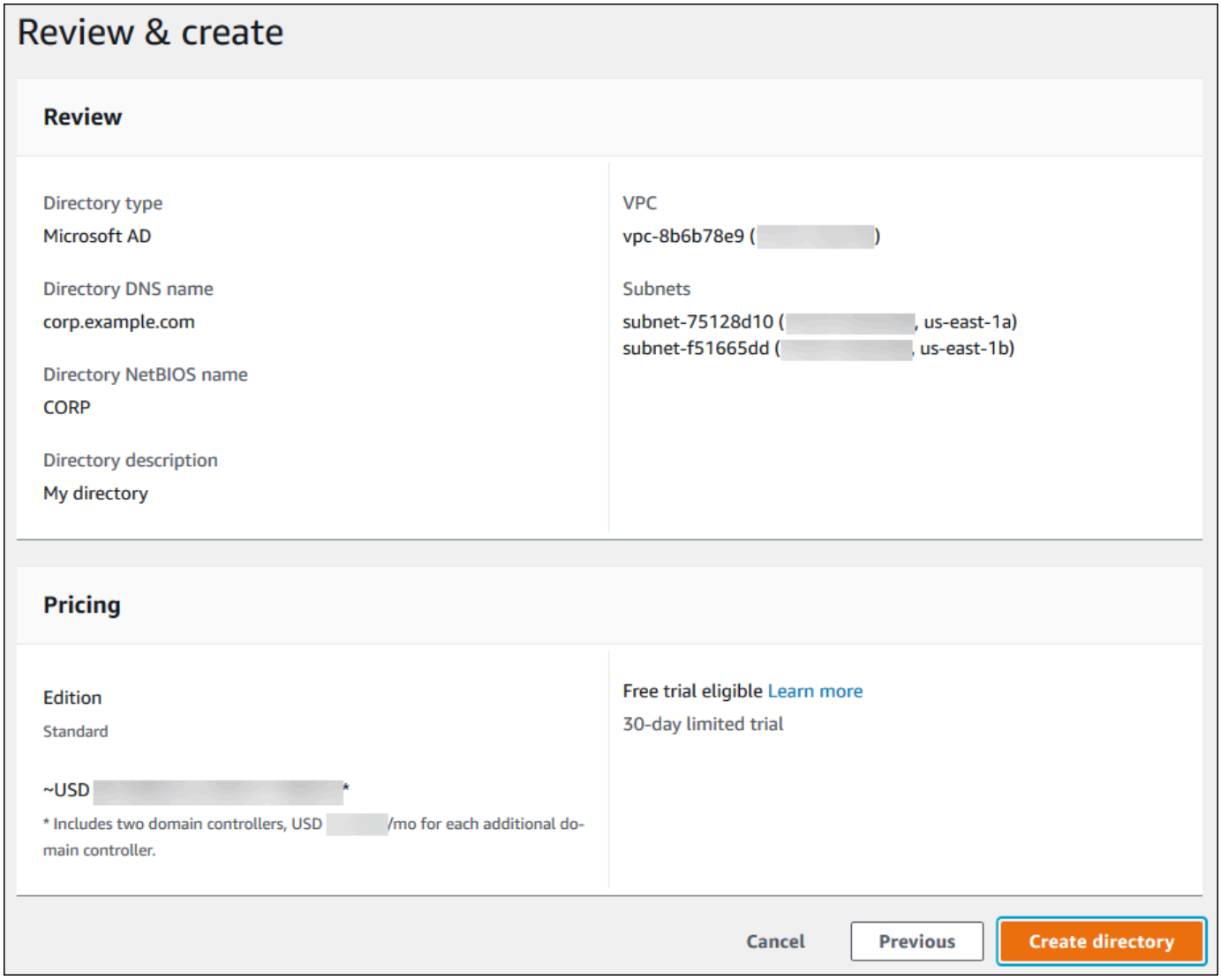
ディレクトリ用の VPC を選択します。

> **Note**
>
> ディレクトリと DB インスタンスは異なる VPC に配置できますが、その場合、必ずクロス VPC トラフィックを有効にしてください。詳細については、「ステップ 4: ディレクトリと DB インスタンス間のクロス VPC トラフィックを有効にする」を参照してください。

Subnets

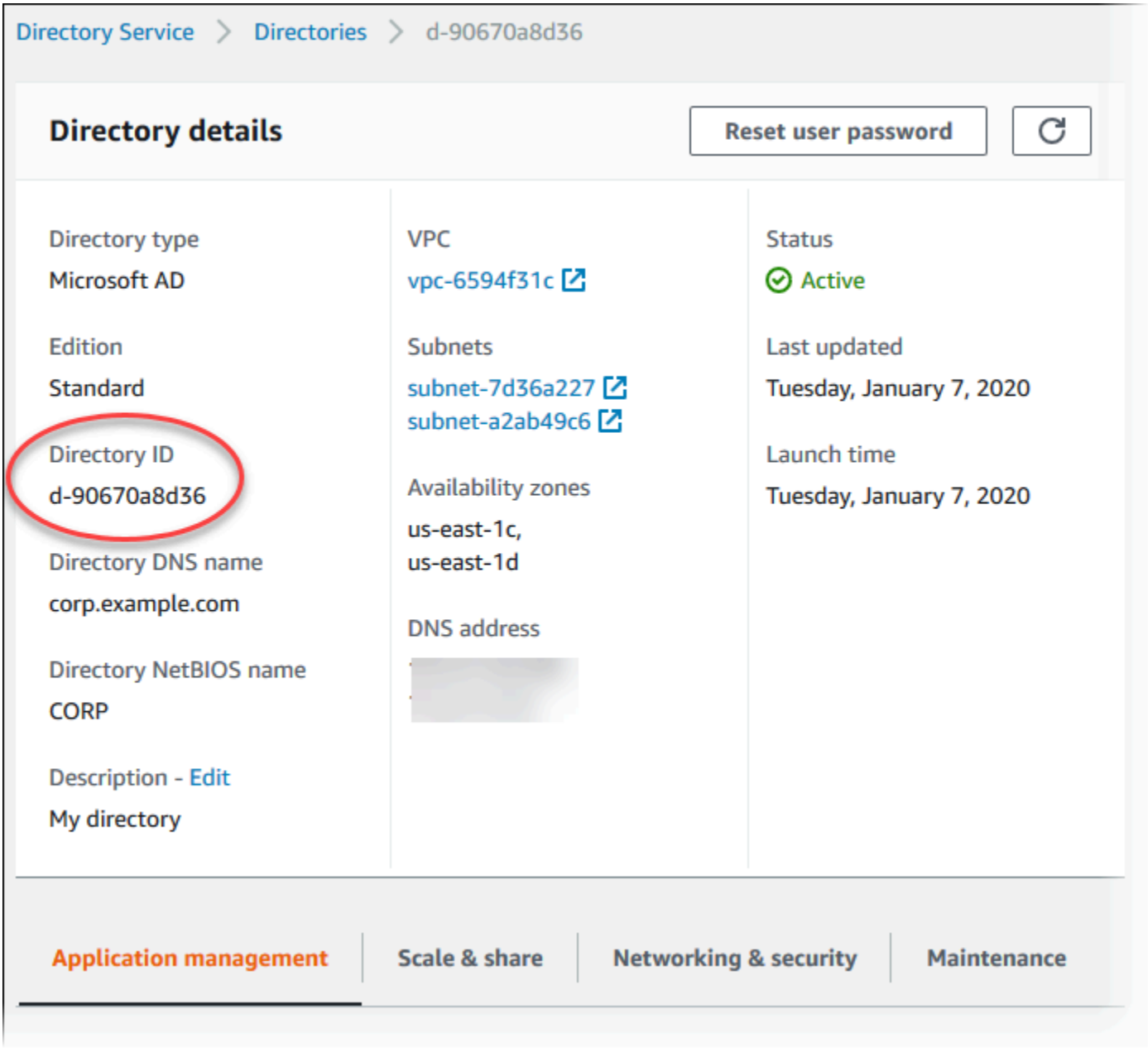
ディレクトリサーバーのサブネットを選択します。2 つのサブネットは、異なるアベイラビリティーゾーンに存在している必要があります。

7. [Next (次へ)] を選択します。
8. ディレクトリの情報を確認します。変更が必要な場合は、[Previous] を選択します。情報が正しい場合は、[Create directory (ディレクトリの作成)] を選択します。



ディレクトリが作成されるまで、数分かかります。正常に作成されると、[Status] 値が [Active] に変わります。

ディレクトリに関する情報を表示するには、ディレクトリの一覧で、そのディレクトリを選択します。ディレクトリ ID を書き留めます。この値は、SQL Server DB インスタンスを作成または変更するときに必要になります。



ステップ 2: Amazon RDS 用の IAM ロールを作成する

コンソールを使用して SQL Server DB インスタンスを作成する場合、このステップをスキップできます。SQL Server DB インスタンスの作成に CLI または RDS API を使用する場合、`AmazonRDSDirectoryServiceAccess` マネージド IAM ポリシーを使用する IAM ロールを作成する必要があります。このロールを使用すると、Amazon RDS で Directory Service への呼び出しを行うことができます。

AWS 管理 `AmazonRDSDirectoryServiceAccess` ポリシーではなく、ドメインを結合するためのカスタムポリシーを使用する場合は、`ds:GetAuthorizedApplicationDetails` アクションを許

可する必要があります。Directory Service API の変更により、この要件は、2019 年 7 月から有効となります。

次の IAM ポリシー AmazonRDSDirectoryServiceAccess は、Directory Service へのアクセスを提供します。

Example Directory Service へのアクセスを提供する IAM ポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
            "ds:DescribeDirectories",
            "ds:AuthorizeApplication",
            "ds:UnauthorizeApplication",
            "ds:GetAuthorizedApplicationDetails"
        ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

リソースベースの信頼関係では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与する特定のリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

両方のグローバル条件コンテキストキーを使用し、aws:SourceArn 値にアカウント ID を含めます。この場合は、aws:SourceAccount 値と aws:SourceArn 値のアカウントは、同じステートメントで使用する場合、同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

信頼関係では、aws:SourceArn グローバル条件コンテキストキーに、必ず、ロールにアクセスするリソースの完全な Amazon リソースネーム (ARN) を使用します。Windows 認証の場合、次の例に示すように DB インスタンスを含めてください。

Example Windows 認証のグローバル条件コンテキストキーとの信頼関係

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [

 "arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                    ]
                }
            }
        }
    ]
}
```

この IAM ポリシーと信頼関係を使用して IAM ロールを作成します。IAM ロールの作成の詳細については、IAM ユーザーガイドの「カスタマー管理ポリシーの作成」を参照してください。

ステップ 3: ユーザーとグループを作成して設定する

Active Directory ユーザーとコンピューターツールを使用して、ユーザーとグループを作成できます。このツールは、Active Directory Domain Services ツールおよび Active Directory Lightweight Directory Services ツールの 1 つです。ユーザーは、ディレクトリにアクセスできる個別の人またはエンティティを表します。グループは、個別のユーザーごとに権限を付与するのではなく、ユーザーのグループに権限を付与または拒否するために非常に便利です。

Directory Service ディレクトリにユーザーとグループを作成するには、Directory Service ディレクトリのメンバーである Windows EC2 インスタンスに接続する必要があります。また、ユーザーとグループを作成する権限を持つユーザーとしてログインしている必要があります。詳細については、AWS Directory Service 管理ガイドの「ユーザーとグループを追加する (Simple AD および AWS Managed Microsoft AD)」を参照してください。

ステップ 4: ディレクトリと DB インスタンス間のクロス VPC トラフィックを有効にする

同じ VPC 内にディレクトリおよび DB インスタンスを配置する場合は、このステップをスキップして ステップ 5: SQL Server DB インスタンスを作成または変更する に進みます。

ディレクトリと DB インスタンスを別の VPC に配置する場合は、VPC ピア接続または AWS Transit Gateway を使用してクロス VPC トラフィックを設定します。

次の手順では、VPC ピア接続を使用して VPC 間のトラフィックを有効にします。Amazon Virtual Private Cloud ピアリング接続ガイドの「VPC ピア機能とは」の手順に従います。

VPC ピア接続を使用してクロス VPC トラフィックを有効にするには

1. 適切な VPC ルーティングを設定し、ネットワークトラフィックが双方向にフローするようにします。
2. DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからのインバウンドトラフィックを受信できることを確認します。
3. トラフィックをブロックするネットワークのアクセスコントロールリスト (ACL) ルールがないことを確認します。

別の AWS アカウントがディレクトリを所有している場合は、ディレクトリを共有する必要があります。

AWS アカウント間でディレクトリを共有するには

1. DB インスタンスを作成する AWS アカウントとの間でディレクトリの共有を開始するには、AWS Managed Microsoft AD 管理ガイドの「チュートリアル: Directory Service ディレクトリを共有して、シームレスに EC2 ドメインを結合する」の手順を実行します。
2. DB インスタンスのアカウントを使用して、Directory Service コンソールにサインインし、続行する前にドメインが必ず SHARED ステータスであることを確認します。
3. DB インスタンスのアカウントを使用して Directory Service コンソールにサインインしている間に、[ディレクトリ ID] の値を書き留めておきます。このディレクトリ ID は、DB インスタンスをドメインに結合するために使用します。

ステップ 5: SQL Server DB インスタンスを作成または変更する

ディレクトリで使用する SQL Server DB インスタンスを作成または変更します。コンソール、CLI、RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これには以下の 2 つの方法があります。

- コンソール、create-db-instance CLI コマンド、または CreateDBInstance RDS API オペレーションを使用して、新しい SQL Server DB インスタンスを作成します。

  手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance CLI コマンド、または ModifyDBInstance RDS API オペレーションを使用して、既存の SQL Server DB インスタンスを変更します。

  手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、restore-db-instance-from-db-snapshot CLI コマンド、または RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから SQL Server DB インスタンスを復元します。

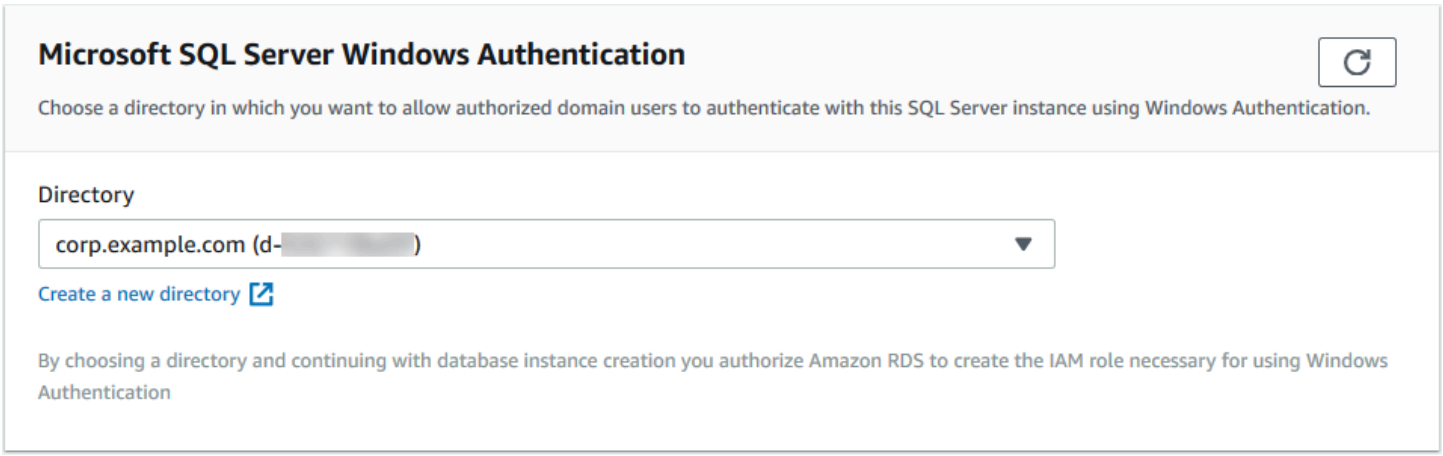
  手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、または RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、SQL Server DB インスタンスをポイントインタイムに復元します。

  手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

Windows 認証は、VPC 内の SQL Server DB インスタンスにのみサポートされています。

DB インスタンスが、作成したドメインディレクトリを使用できるようにするには、次が必要です。

- [ディレクトリ] では、ディレクトリの作成時に生成されたドメイン識別子 (d-*ID*) を選択する必要があります。
- VPC セキュリティグループに、ディレクトリとの通信を DB インスタンスに許可するアウトバウンドルールがあることを確認します。



AWS CLI を使用する場合は、DB インスタンスが、作成したディレクトリを使用できるように、以下のパラメータが必要です。

- `--domain` パラメータには、ディレクトリの作成時に生成されたドメイン識別子 (d-*ID*) を使用します。
- `--domain-iam-role-name` パラメータには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する作成済みのロールを使用します。

例えば、以下の CLI コマンドはディレクトリを使用するように DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Kerberos 認証を有効化するために DB インスタンスを変更する場合、変更後 DB インスタンスを再起動します。

ステップ 6: Windows 認証の SQL Server ログインを作成する

Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同じように SQL Server DB インスタンスに接続します。DB インスタンスは AWS Managed Microsoft AD ドメインに参加しているため、SQL Server のログインとユーザーをプロビジョニングできます。これは、ドメイン内の Active Directory のユーザーとグループから行われます。データベースへのアクセス許可は、これらの Windows ログインに付与され無効化されている標準の SQL サーバーのアクセス許可によって管理されています。

Active Directory のユーザーが SQL Server で認証するには、ユーザー、またはそのユーザーがメンバーとして含まれているグループに、SQL Server の Windows ログインが存在する必要があります。これらの SQL Server ログインでアクセスを許可したり取り消したりして、細分化されたアクセスコントロールを処理します。SQL Server ログインを持たないユーザー、またはそのようなログインを持つグループに属さないユーザーは、SQL Server DB インスタンスにアクセスできません。

Active Directory の SQL Server ログインを作成するには、ALTER ANY LOGIN 許可が必要です。このアクセス許可を持つログインをまだ作成していない場合は、SQL Server 認証を使用して DB インスタンスのマスターユーザーとして接続します。

次の例のようなデータ定義言語 (DDL) コマンドを実行して、Active Directory ユーザーまたはグループへの SQL Server ログインを作成します。

> **Note**
>
> *domainName**login_name* の形式で Windows 2000 以前のログイン名を使用して、ユーザーまたはグループを指定します。ユーザープリンシパル名 (UPN) を *login_name* @ *DomainName* の形式で使用することはできません。
> T-SQL ステートメントを使用することによってのみ、RDS for SQL Server インスタンスで Windows 認証ログインを作成することができます。SQL Server Management Studio を使用して Windows 認証ログインを作成することはできません。

```
USE [master]
```

```
GO
CREATE LOGIN [mydomain\myuser] FROM WINDOWS WITH DEFAULT_DATABASE = [master],
 DEFAULT_LANGUAGE = [us_english];
GO
```

> **Note**
>
> Active Directory ユーザーを削除し、同じ sAMAccountName で新しいユーザーを作成し、RDS for SQL Server インスタンスで CREATE LOGIN [DOMAIN\username] FROM WINDOWS を実行すると、新しいユーザーの SID ではなく、削除されたユーザーの SID を使用してログインが作成され、Windows 認証が失敗する可能性があります。これは、RDS ホストが OS レベルで名前から SID へのマッピングをキャッシュし、このキャッシュにお客様が直接アクセスできないために発生します。この問題を回避するには、プロビジョニングサイクルごとに一意の sAMAccountName を使用します (タイムスタンプやバージョンサフィックスを追加するなど)。同じ名前を再利用する必要がある場合は、AD ユーザーを再作成した後に DBCC FREESYSTEMCACHE('TokenAndPermUserStore') を実行し、最大 10 分待ってから (この間隔は RDS マネージドインスタンスでは設定できないことから、OS レベルのキャッシュの更新を許可するため)、ログインを作成する前に SUSER_SID('DOMAIN\username') が予期される SID を返すことを確認します。

詳細については、Microsoft Developer Network ドキュメントの「ログインの作成 (Transact-SQL)」を参照してください。

ドメインのユーザー (人およびアプリケーション) が Windows 認証を使用して、クライアントマシンを結合したドメインから RDS for SQL Server インスタンスに接続できるようになりました。

ドメインの DB インスタンスの管理

コンソール、AWS CLI、または Amazon RDS API を使用して、DB インスタンスおよびドメインとの関係を管理できます。例えば、DB インスタンスをドメイン内、ドメイン外、またはドメイン間で移動させることができます。

例えば、Amazon RDS API を使用して次を実行できます。

- 失敗したメンバーシップのドメイン参加を再試行するには、ModifyDBInstance API オペレーションを使用して、現在のメンバーシップのディレクトリ ID を指定します。
- メンバーシップの IAM ロール名を更新するには、ModifyDBInstance API オペレーションを使用し、現在のメンバーシップのディレクトリ ID と新しい IAM ロールを指定します。

- ドメインから DB インスタンスを削除するには、ModifyDBInstance API オペレーションを使用し、ドメインパラメータとして none を指定します。
- ドメイン間で DB インスタンスを移動するには、ModifyDBInstance API オペレーションを使用し、新しいドメインのドメイン識別子をドメインパラメータとして指定します。
- 各 DB インスタンスのメンバーシップを一覧表示するには、DescribeDBInstances API オペレーションを使用します。

ドメインのメンバーシップを理解する

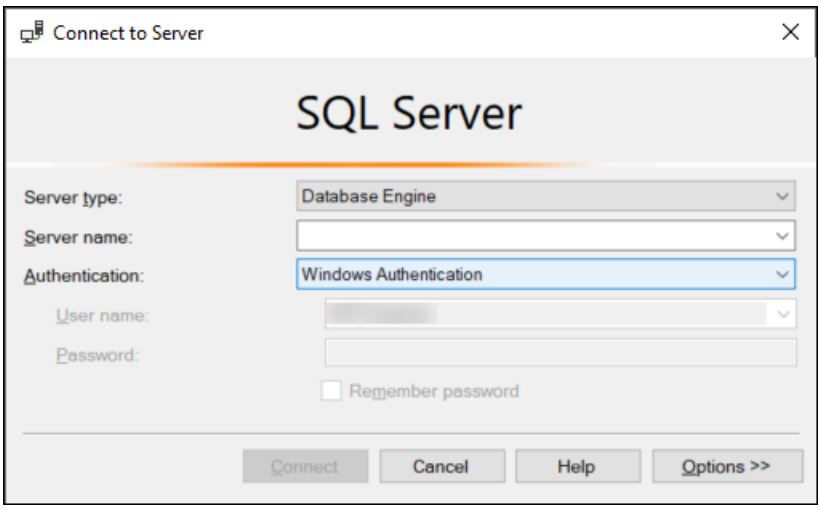
DB インスタンスを作成または変更した後、そのインスタンスは、ドメインのメンバーになります。AWS コンソールは、DB インスタンスのドメインメンバーシップのステータスを示します。DB インスタンスのステータスは、以下のいずれかです。

- joined (参加済み) – インスタンスはドメインのメンバーになっています。
- joining (参加中) – インスタンスは、ドメインのメンバーになる途中です。
- pending-join (参加保留中) – インスタンスのメンバーシップは保留中です。
- メンテナンスまで参加保留中 – 次に予定されているメンテナンスウィンドウ中に、AWS がインスタンスをドメインのメンバーにできるよう試みます。
- pending-removal (削除保留中) – ドメインからのインスタンス削除は保留中です。
- メンテナンスまで削除保留中 – 次に予定されているメンテナンスウィンドウ中に、AWS がドメインからのインスタンスの削除を試みます。
- failed (失敗) – 設定の問題により、インスタンスはドメインに参加できませんでした。インスタンスの変更コマンドを再発行する前に、設定を確認して修正してください。
- removing (削除中) – インスタンスをドメインから削除しています。

ドメインのメンバーになるリクエストは、ネットワーク接続の問題や正しくない IAM ロールが原因で失敗する場合があります。例えば、DB インスタンスを作成した、または既存のインスタンスを変更したが、DB インスタンスをドメインに参加させられない場合があります。この場合、コマンドを再発行して DB インスタンスを作成または変更するか、新しく作成されたインスタンスを変更してドメインに参加させます。

Windows 認証を使用して SQL Server に接続する

Windows 認証を使用して SQL Server に接続するには、ドメインのユーザーとしてドメイン結合されたコンピュータにログインしている必要があります。SQL Server Management Studio の起動後、認証タイプとして Windows 認証を選択します (以下参照)。



SQL Server DB インスタンスを復元してドメインに追加する

SQL Server DB インスタンスの DB スナップショットまたはポイントインタイムリカバリ (PITR) を復元し、ドメインに追加できます。DB インスタンスが復元されたら、「ステップ 5: SQL Server DB インスタンスを作成または変更する」で説明している手順に従ってインスタンスを変更し、DB インスタンスをドメインに追加します。

Microsoft SQL Server DB エンジンのアップグレード

新しいバージョンのデータベースエンジンが Amazon RDS でサポートされている場合は、DB インスタンスをその新しいバージョンにアップグレードできます。SQL Server DB インスタンスのアップグレードには、メジャーバージョンのアップグレードとマイナーバージョンのアップグレードの 2 種類あります。

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、DB インスタンスのメジャーバージョンアップグレードは手動で実行する必要があります。メジャーバージョンアップグレードを開始するには、DB インスタンスを変更します。ただし、メジャーバージョンアップグレードを実行する前に、「RDS for SQL Server のアップグレードのテスト」に記載されているステップに従ってアップグレードをテストすることをお勧めします。

マイナーバージョンアップグレードには、既存のアプリケーションとの下位互換性がある変更のみが含まれます。DB インスタンスのマイナーバージョンは、次の 2 つの方法でアップグレードできます。

- 手動 – DB インスタンスを変更してアップグレードを開始する
- 自動 – DB インスタンスのマイナーバージョン自動アップグレードを有効にする

マイナーバージョン自動アップグレードを有効にすると、RDS for SQL Server は、新しいマイナーバージョンで重要なセキュリティ更新が利用可能になったときに、スケジュールされたメンテナンス期間中にデータベースインスタンスを自動的にアップグレードします。

16.00.4120.1、15.00.4365.2、14.00.3465.1、13.00.6435.1 以降のマイナーエンジンバージョンでは、以下のセキュリティプロトコルがデフォルトで無効になっています。

- `rds.tls10` (TLS 1.0 プロトコル)
- `rds.tls11` (TLS 1.1 プロトコル)
- `rds.rc4` (RC4 暗号)
- `rds.curve25519` (Curve25519 暗号)
- `rds.3des168` (Triple DES 暗号)

以前のエンジンバージョンでは、Amazon RDS はこれらのセキュリティプロトコルをデフォルトで有効にします。

```
...

"ValidUpgradeTarget": [
    {
        "Engine": "sqlserver-se",
        "EngineVersion": "14.00.3281.6.v1",
        "Description": "SQL Server 2017 14.00.3281.6.v1",
        "AutoUpgrade": false,
        "IsMajorVersionUpgrade": false
    }
...
```

アップグレード実行の詳細については、「SQL Server DB インスタンスをアップグレードする」を参照してください。Amazon RDS で使用できる SQL Server のバージョンについての詳細は、「Amazon RDS for Microsoft SQL Server」を参照してください。

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

トピック

- RDS for SQL Server のメジャーバージョンアップグレード
- SQL Server のアップグレードに関する考慮事項
- RDS for SQL Server のアップグレードのテスト
- SQL Server DB インスタンスをアップグレードする
- サポート終了前に非推奨の DB インスタンスをアップグレードする

RDS for SQL Server のメジャーバージョンアップグレード

Amazon RDS は、現在次のメジャーバージョンの Microsoft SQL Server DB インスタンスへのアップグレードをサポートしています。

SQL Server 2008 を除く任意のバージョンから既存の DB インスタンスを SQL Server 2017、2019、2022、または 2025 にアップグレードできます。SQL Server 2008 からアップグレードするには、他のいずれかのバージョンにアップグレードしてください。

現在のバージョン	サポートされているアップグレードバージョン
SQL Server 2022	SQL Server 2025
SQL Server 2019	SQL Server 2025 SQL Server 2022
SQL Server 2017	SQL Server 2025 SQL Server 2022 SQL Server 2019
SQL Server 2016	SQL Server 2025 SQL Server 2022 SQL Server 2019 SQL Server 2017

次の例に示すような AWS CLI クエリを使用して、データベースエンジンのバージョン別に利用できるアップグレードを見つけることができます。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
    --engine sqlserver-se \
    --engine-version 14.00.3281.6.v1 \
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" \
    --output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
    --engine sqlserver-se ^
    --engine-version 14.00.3281.6.v1 ^
```

```
    --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
    --output table
```

出力は、バージョン 14.00.3281.6 を利用可能な最新の SQL Server 2017 または 2019 バージョンにアップグレードできることを示しています。

```
-------------------------
|DescribeDBEngineVersions|
+------------------------+
|      EngineVersion     |
+------------------------+
|  14.00.3294.2.v1       |
|  14.00.3356.20.v1      |
|  14.00.3381.3.v1       |
|  14.00.3401.7.v1       |
|  14.00.3421.10.v1      |
|  14.00.3451.2.v1       |
|  15.00.4043.16.v1      |
|  15.00.4073.23.v1      |
|  15.00.4153.1.v1       |
|  15.00.4198.2.v1       |
|  15.00.4236.7.v1       |
+------------------------+
```

データベース互換性レベル

Microsoft SQL Server データベース互換性レベルを使用して、いくつかのデータベースの動作を調整し、以前のバージョンの SQL Server を模倣することができます。詳細については、Microsoft ドキュメントの「互換性レベル」を参照してください。DB インスタンスをアップグレードしても、既存のすべてのデータベースは元の互換性レベルのままとなります。

ALTER DATABASE コマンドを使用して、データベースの互換性レベルを変更できます。例えば、`customeracct` という名前のデータベースが、SQL Server 2016 との互換性を持つように変更するには、次のコマンドを発行します。

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 130
```

SQL Server のアップグレードに関する考慮事項

Amazon RDS によってアップグレードプロセス中に 2 つの DB スナップショットが作成されます。初期の DB スナップショットは、アップグレードの変更が行われる前の DB インスタンスから作成されます。アップグレード完了後に、2 番目の DB スナップショットが取得されます。

> **Note**
>
> DB インスタンスのバックアップ保持期間を 0 より大きく設定した場合にのみ、Amazon RDS は DB スナップショットを作成します。バックアップ保持期間を変更するには、「Amazon RDS DB インスタンスを変更する」を参照してください。

アップグレード完了後は、データベースエンジンを前のバージョンに戻すことはできません。前のバージョンに戻す必要がある場合は、アップグレード前に取得された DB スナップショットを復元して、新しい DB インスタンスを作成します。

SQL Server のマイナーバージョンアップグレードまたはメジャーバージョンアップグレード中、[Free Storage Space] と [Disk Queue Depth] のメトリクスに [-1] が表示されます。アップグレード完了後は、両方のメトリクスが [normal (ノーマル)] に戻ります。

SQL Server インスタンスをアップグレードする前に、次の情報を確認してください。

トピック

- アップグレードを開始する前のベストプラクティス
- マルチ AZ の考慮事項
- リードレプリカに関する考慮事項
- オプショングループに関する考慮事項
- パラメータグループに関する考慮事項

アップグレードを開始する前のベストプラクティス

アップグレードプロセスを開始する前に、次の準備手順を実装して、最適なアップグレードパフォーマンスを実現し、潜在的な問題を最小限に抑えます。

タイミングとワークロードの管理

- トランザクション量が少ない期間にアップグレードをスケジュールします。

- アップグレードウィンドウ中の書き込み操作を最小限に抑えます。

  これにより、Amazon RDS はセカンダリからプライマリへのペアリング中に RDS が復元する必要があるトランザクションログファイルの数を減らすことで、アップグレードを迅速に完了できます。

トランザクション管理

- 長時間実行されるトランザクションを特定してモニタリングします。
- アップグレードを開始する前に、すべての重要なトランザクションがコミットされていることを確認します。
- アップグレードウィンドウ中に長時間実行されるトランザクションを防止します。

ログファイルの最適化

トランザクションログファイルを確認して最適化します。

- サイズが大きすぎるログファイルを縮小します。
- 高いログ消費パターンを減らします。
- 仮想ログファイル (VLF) を管理します。
- 通常のオペレーションに十分な空き領域を維持します。

マルチ AZ の考慮事項

Amazon RDS は、Microsoft SQL Server を実行する DB インスタンスで SQL Server データベースミラーリング (DBM) または Always On 可用性グループ (AG) によるマルチ AZ 配置をサポートしています。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。

マルチ AZ 配置 (ミラーリング/AlwaysOn) では、アップグレードをリクエストすると、RDS はプライマリインスタンスとセカンダリインスタンスでローリングアップグレード戦略に従います。ローリングアップグレードでは、セカンダリインスタンスのアップグレード中に少なくとも 1 つのインスタンスをトランザクションに使用できます。停止はフェイルオーバー時に限定されると予想されます。

アップグレード中、RDS はマルチ AZ 設定からセカンダリインスタンスを削除して、セカンダリインスタンスのアップグレードを実行し、切断中にプライマリから取得したトランザクションログのバックアップを復元します。すべてのログバックアップを復元すると、RDS はアップグレードしたセカンダリをプライマリに結合します。すべてのデータベースが同期状態になると、RDS はアップ

グレードしたセカンダリインスタンスへのフェイルオーバーを実行します。フェイルオーバーが完了すると、RDS は古いプライマリインスタンスのアップグレードに進み、トランザクションログのバックアップを復元して、新しいプライマリとペアリングします。

このフェイルオーバー期間を最小限に抑えるために、接続文字列で `MultiSubnetFailover` 接続オプションをサポートするクライアントライブラリを使用する場合は、AlwaysOn AG 可用性グループリスナーのエンドポイントを使用することをお勧めします。可用性グループリスナーのエンドポイントを使用する場合、フェイルオーバー時間は通常 10 秒未満ですが、この時間には追加のクラッシュ復旧時間は含まれません。

リードレプリカに関する考慮事項

データベースバージョンのアップグレード中に、Amazon RDS はプライマリ DB インスタンスと共にすべてのリードレプリカをアップグレードします。Amazon RDS では、リードレプリカのデータベースバージョンのアップグレードを個別にサポートしていません。リードレプリカの詳細については、「Amazon RDS での Microsoft SQL Server 用のリードレプリカの使用」を参照してください。

プライマリ DB インスタンスのデータベースバージョンのアップグレードを実行すると、そのすべてのリードレプリカも自動的にアップグレードされます。Amazon RDS は、プライマリ DB インスタンスをアップグレードする前に、すべてのリードレプリカを同時にアップグレードします。リードレプリカは、プライマリ DB インスタンスのデータベースバージョンのアップグレードが完了するまで使用できない場合があります。

オプショングループに関する考慮事項

DB インスタンスでカスタム DB オプショングループを使用している場合、Amazon RDS で DB インスタンスに新しいオプショングループを自動的に割り当てられないことがあります。例えば、新しいメジャーバージョンにアップグレードする場合、新しいオプショングループを指定する必要があります。新しいオプショングループを作成し、このオプショングループに既存のカスタムオプショングループと同じオプションを追加することをお勧めします。

詳細については、「オプショングループを作成する」または「オプショングループをコピーする」を参照してください。

パラメータグループに関する考慮事項

DB インスタンスがカスタム DB パラメータグループを使用している場合:

- Amazon RDS は、アップグレード後に DB インスタンスを自動的に再起動します。

- 場合によっては、RDS が新しいパラメータグループを DB インスタンスに自動的に割り当てられないことがあります。

  例えば、新しいメジャーバージョンにアップグレードする場合、新しいパラメータグループを指定する必要があります。新しいパラメータグループを作成し、そのパラメータの設定を既存のカスタムパラメータグループと同じにすることをお勧めします。

詳細については、「Amazon RDS での DB パラメータグループの作成」または「Amazon RDS での DB パラメータグループのコピー」を参照してください。

RDS for SQL Server のアップグレードのテスト

DB インスタンスのメジャーバージョンのアップグレードを実行する前に、データベースとそのデータベースにアクセスするすべてのアプリケーションについて、新しいバージョンとの互換性を綿密にテストする必要があります。以下の手順を実行することをお勧めします。

メジャーバージョンのアップグレードをテストするには

1. Microsoft ドキュメントで、新しいバージョンのデータベースエンジンに関する「SQL Server をアップグレードする」を参照し、データベースやアプリケーションに影響する可能性がある互換性の問題があるかどうかを確認します。
2. DB インスタンスでカスタムオプショングループを使用している場合は、アップグレード先の新しいバージョンと互換性がある新しいオプショングループを作成します。詳細については、「オプショングループに関する考慮事項」を参照してください。
3. DB インスタンスでカスタムパラメータグループを使用している場合は、アップグレード先の新しいバージョンと互換性がある新しいパラメータグループを作成します。詳細については、「パラメータグループに関する考慮事項」を参照してください。
4. アップグレードする DB インスタンスの DB スナップショットを作成します。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
5. DB スナップショットを復元して、新しいテスト DB インスタンスを作成します。詳細については、「DB インスタンスへの復元」を参照してください。
6. この新しいテスト DB インスタンスを変更して新しいバージョンにアップグレードするには、次に説明するいずれかの方法を使用します。

   - コンソール
   - AWS CLI

- RDS API

7. アップグレードしたインスタンスによって使用されるストレージを評価して、アップグレードに追加のストレージが必要かどうかを判断します。
8. データベースとアプリケーションが新しいバージョンで正常に動作することが確認されるまで、アップグレードした DB インスタンスに対する品質保証テストを必要な回数だけ実行します。手順 1 で特定した互換性の問題の影響を評価するための新しいテストを実行します。すべてのストアドプロシージャと関数をテストします。アプリケーションのテストバージョンを、アップグレードした DB インスタンスに割り振ります。
9. すべてのテストに合格したら、本稼働 DB インスタンスのアップグレードを実行します。すべてが正常に動作していることを確認するまでは、DB インスタンスへの書き込みオペレーションは許可しないことをお勧めします。

SQL Server DB インスタンスをアップグレードする

SQL Server DB インスタンスの手動または自動アップグレードについては、以下を参照してください。

- DB インスタンスのエンジンバージョンのアップグレード
- Best practices for upgrading SQL Server 2008 R2 to SQL Server 2016 on Amazon RDS for SQL Server

> **Important**
>
> AWS KMS を使用して暗号化されたスナップショットがある場合は、サポートが終了する前にアップグレードを開始することをお勧めします。

サポート終了前に非推奨の DB インスタンスをアップグレードする

メジャーバージョンの廃止後は、新しい DB インスタンスにインストールすることはできません。RDS では、既存のすべての DB インスタンスが自動的にアップグレードされます。

廃止予定の DB インスタンスを復元する必要がある場合、ポイントインタイムリカバリ (PITR) を実行するか、スナップショットを復元することができます。こうすることで、廃止予定のバージョンを使用する DB インスタンスに一時的にアクセスできます。ただし、メジャーバージョンが完全に廃止

されると、これらの DB インスタンスも自動的にサポートされているバージョンにアップグレードされます。

RDS for SQL Server のストレージの使用

RDS for SQL Server では、RDS for SQL Server インスタンスに最大 3 つの追加ボリュームをアタッチでき、それぞれが一意の Windows ドライブ文字にマッピングされます。これにより、データベースファイルをデフォルトの `D:` ドライブ以外の複数のボリュームに分散できます。ストレージボリュームを追加すると、データベースファイル管理とストレージ最適化の柔軟性が向上します。

その他の利点には以下が含まれます。

- 柔軟なファイル配布 – データベースデータファイルとログファイルを複数のボリュームに分散して、I/O パフォーマンスを向上させます。
- ストレージの最適化 – ワークロード要件に応じて異なるストレージタイプと設定を使用します。
- スケーラビリティ – 既存のボリュームを変更せずにストレージ容量を追加します。

このページの内容

- RDS for SQL Server で追加のストレージボリュームを使用する際の考慮事項
- RDS for SQL Server を使用してストレージボリュームを追加、削除、または変更する
- RDS for SQL Server を使用した追加のストレージボリュームの復元オペレーション
- RDS for SQL Server を使用した追加のストレージボリュームのユースケース

RDS for SQL Server で追加のストレージボリュームを使用する際の考慮事項

RDS for SQL Server で追加のストレージボリュームを使用する場合は、次の機能と制限に注意してください。

- SQL Server Standard Edition (SE)、Enterprise Edition (EE)、および Developer Edition (DEV-EE) でのみストレージボリュームを追加できます。
- インスタンスごとに最大 3 つのストレージボリュームを追加できます。
- ボリューム名は、次のように Windows ドライブ文字に自動的にマッピングされます。
  - `rdsdbdata2` – `H:` ドライブ
  - `rdsdbdata3` – `I:` ドライブ
  - `rdsdbdata4` – `J:` ドライブ

- TempDB ファイルは、NVMe インスタンスストレージを使用するときに引き続き T: ドライブを使用します。SQL Server 監査ファイルと Microsoft Business Intelligence (MSBI) ファイルは D: ドライブに残ります。
- 汎用 SSD (gp3) およびプロビジョンド IOPS SSD (io2) ストレージタイプのみを追加できます。
- 追加のストレージボリュームの最小ストレージサイズは、デフォルトの D: ドライブに設定されている制限と同じです。DB インスタンスの最大ストレージサイズは、すべてのボリュームで合計 256 TiB です。
- リードレプリカを持つインスタンスまたはリードレプリカインスタンスへのストレージボリュームの追加はサポートされていません。
- クロスリージョン自動バックアップが有効になっているインスタンスへのストレージボリュームの追加はサポートされていません。
- ストレージの自動スケーリング用の追加ストレージボリュームの設定はサポートされていません。
- 作成後のボリューム間のファイルの移動はサポートされていません。
- D: ボリュームを削除することはできませんが、空である限り、他のストレージボリュームを削除することはできます。
- スナップショット復元またはポイントインタイムリカバリ (PITR) 中の既存のボリュームのサイズの変更はサポートされていません。ただし、復元オペレーション中に新しいストレージボリュームを追加できます。

RDS for SQL Server を使用してストレージボリュームを追加、削除、または変更する

AWS CLI または AWS マネジメントコンソールを使用して、追加のストレージボリュームを追加、変更、削除できます。すべてのオペレーションは、`additional-storage-volumes` パラメータで `modify-db-instance` API オペレーションを使用します。

> **Important**
>
> ストレージボリュームを追加または削除すると、バックアップ保留中のアクションとポイントインタイムリストアブラックアウトウィンドウが作成されます。このウィンドウは、バックアップワークフローが完了すると閉じます。

トピック

- ストレージボリュームの追加

- 追加ストレージボリュームのスケーリング
- 追加ストレージボリュームの削除

ストレージボリュームの追加

デフォルトの D: ドライブ以外に最大 3 つのストレージボリュームを追加できます。RDS for SQL Server インスタンスに新しいストレージボリュームを追加するには、additional-storage-volumes パラメータを指定して modify-db-instance コマンドを使用します。

次の例では、rdsdbdata4 という名前の新しい 4,000 GiB 汎用 SSD (gp3) ボリュームを追加します。

```
aws rds modify-db-instance \
  --db-instance-identifier my-sql-server-instance \
  --region us-east-1 \
  --additional-storage-volumes
 '[{"VolumeName":"rdsdbdata4","StorageType":"gp3","AllocatedStorage":4000}]' \
  --apply-immediately
```

追加ストレージボリュームのスケーリング

ストレージサイズを除き、追加ボリュームの任意のストレージ設定を変更できます。次の例では、rdsdbdata2 ボリュームの IOPS 設定を変更します。

```
aws rds modify-db-instance \
  --db-instance-identifier my-sql-server-instance \
  --region us-east-1 \
  --additional-storage-volumes '[{"VolumeName":"rdsdbdata2","IOPS":4000}]' \
  --apply-immediately
```

追加のストレージボリュームを変更すると、そのボリュームはストレージ最適化フェーズに入ります。変更されたボリュームの最適化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの AdditionalStorageVolumes 配列内の対応するエントリを調べます。最適化中、そのエントリの StorageOperationStatus は Optimizing に設定され、StorageOperationPercentProgress は 0 ~ 100 を報告します。詳細については、「ボリュームあたりのストレージオペレーションのモニタリング」を参照してください。

追加ストレージボリュームの削除

D: ボリュームは削除できませんが、空になった他のストレージボリュームは削除できます。

> **Warning**
>
> 追加のストレージボリュームを削除する前に、データベースファイルがボリュームに保存されていないことを確認してください。

次の例では、rdsdbdata4 ボリュームを削除しています。

```
aws rds modify-db-instance \
  --db-instance-identifier my-sql-server-instance \
  --region us-east-1 \
  --additional-storage-volumes '[{"VolumeName":"rdsdbdata4","SetForDelete":true}]' \
  --apply-immediately
```

RDS for SQL Server を使用した追加のストレージボリュームの復元オペレーション

データベースを復元するときに、ストレージボリュームを追加できます。既存のボリュームのストレージ設定を変更することもできます。

トピック

- スナップショット復元
- ポイントインタイムリカバリ
- ネイティブデータベースの復元

スナップショット復元

スナップショットから復元する場合、新しいストレージボリュームを追加したり、既存のボリュームの IOPS、スループット、ストレージタイプの設定を変更したりできます。

次の例では、スナップショットから DB インスタンスを復元し、rdsdbdata2 ボリュームの IOPS 設定を変更します。

```
aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier my-restored-instance \
  --db-snapshot-identifier my-snapshot \
  --region us-east-1 \
```

```
  --additional-storage-volumes '[{"VolumeName":"rdsdbdata2","IOPS":5000}]'
```

ポイントインタイムリカバリ

ポイントインタイムリカバリ (PITR) 中に、カスタム設定で新しいストレージボリュームを追加できます。

次の例では、PITR を実行し、新しい 5,000 GiB の汎用 SSD (gp3) ボリュームを追加します。

```
aws rds restore-db-instance-to-point-in-time \
  --source-db-instance-identifier my-source-instance \
  --target-db-instance my-pitr-instance \
  --use-latest-restorable-time \
  --region us-east-1 \
  --additional-storage-volumes
 '[{"VolumeName":"rdsdbdata4","StorageType":"gp3","AllocatedStorage":5000,"IOPS":5000,"StorageT
```

ネイティブデータベースの復元

rds_restore_database ストアドプロシージャを使用して、特定の追加のストレージボリュームにデータベースを復元できます。2 つの新しいパラメータがボリューム選択をサポートします。

data_file_volume

データベースデータファイルのドライブ文字を指定します。

log_file_volume

データベースログファイルのドライブ文字を指定します。

次の例では、H: ドライブ上のデータファイルと I: ドライブ上のログファイルを含むデータベースを復元します。

```
EXEC msdb.dbo.rds_restore_database
    @restore_db_name='my_database',
    @s3_arn_to_restore_from='arn:aws:s3:::my-bucket/backup-file.bak',
    @data_file_volume='H:',
    @log_file_volume='I:';
```

ボリュームパラメータを指定しない場合、または両方のパラメータに D: ドライブを指定した場合、データベースファイルはデフォルトの D: ドライブに復元されます。

```
EXEC msdb.dbo.rds_restore_database
    @restore_db_name='my_database',
    @s3_arn_to_restore_from='arn:aws:s3:::my-bucket/backup-file.bak';
```

RDS for SQL Server を使用した追加のストレージボリュームのユースケース

追加のストレージボリュームは、さまざまなデータベース管理シナリオをサポートします。以下のセクションでは、一般的なユースケースと実装アプローチについて説明します。

トピック

- 追加のストレージボリュームでのデータベースの作成
- ストレージ容量の拡張
- ボリューム間のデータベースの移動
- コスト効率の高いストレージへのデータのアーカイブ

追加のストレージボリュームでのデータベースの作成

標準の SQL Server CREATE DATABASE ステートメントを使用して、追加のストレージボリュームで新しいデータベースを直接作成できます。

次の例では、H: ドライブにデータファイル、I: ドライブにログファイルを含むデータベースを作成します。

```
CREATE DATABASE MyDatabase
ON (
    NAME = 'MyDatabase_Data',
    FILENAME = 'H:\rdsdbdata\data\MyDatabase_Data.mdf',
    SIZE = 100MB,
    FILEGROWTH = 10MB
)
LOG ON (
    NAME = 'MyDatabase_Log',
    FILENAME = 'I:\rdsdbdata\data\MyDatabase_Log.ldf',
    SIZE = 10MB,
    FILEGROWTH = 10%
);
```

ストレージ容量の拡張

デフォルトの D: ドライブが最大容量に達したら、ストレージボリュームを追加し、既存のボリュームをスケールして、新しいボリュームに新しいデータファイルまたはログファイルを作成できます。

ストレージ容量を拡張するには

1. modify-db-instance コマンドを使用して、インスタンスにストレージボリュームを追加します。
2. 追加のストレージボリュームに新しいデータファイルを追加します。

```
ALTER DATABASE MyDatabase
ADD FILE (
    NAME = 'MyDatabase_Data2',
    FILENAME = 'H:\rdsdbdata\data\MyDatabase_Data2.ndf',
    SIZE = 500MB,
    FILEGROWTH = 50MB
);
```

ボリューム間のデータベースの移動

データベースを別のボリュームに移動するには、rds_backup_database および rds_restore_database ストアドプロシージャでバックアップと復元の方法を使用します。詳細については、「ネイティブバックアップおよび復元の使用」を参照してください。

データベースを別のボリュームに移動するには

1. rds_backup_database を使用してデータベースをバックアップします。

```
EXEC msdb.dbo.rds_backup_database
    @source_db_name='MyDatabase',
    @s3_arn_to_backup_to='arn:aws:s3:::my-bucket/database-backup.bak';
```

2. データベースをターゲットボリュームに復元します。

```
EXEC msdb.dbo.rds_restore_database
    @restore_db_name='MyDatabase_New',
    @s3_arn_to_restore_from='arn:aws:s3:::my-bucket/database-backup.bak',
    @data_file_volume='H:',
    @log_file_volume='I:';
```

3. 古いドライブからデータベースを削除して、スペースを解放します。詳細については、「Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースの削除」を参照してください。

コスト効率の高いストレージへのデータのアーカイブ

パーティション化されたテーブルの場合、古いデータを異なるパフォーマンス特性を持つ追加のストレージボリュームにアーカイブできます。

パーティション化されたデータをアーカイブするには

1. 適切なストレージタイプと容量を持つストレージボリュームを追加します。
2. 追加のストレージボリュームに新しいファイルグループを作成します。

```
ALTER DATABASE MyDatabase
ADD FILEGROUP ArchiveFileGroup;

ALTER DATABASE MyDatabase
ADD FILE (
    NAME = 'Archive_Data',
    FILENAME = 'H:\rdsdbdata\data\Archive_Data.ndf',
    SIZE = 1GB,
    FILEGROWTH = 100MB
) TO FILEGROUP ArchiveFileGroup;
```

3. SQL Server パーティション管理コマンドを使用して、パーティションを新しいファイルグループに移動します。

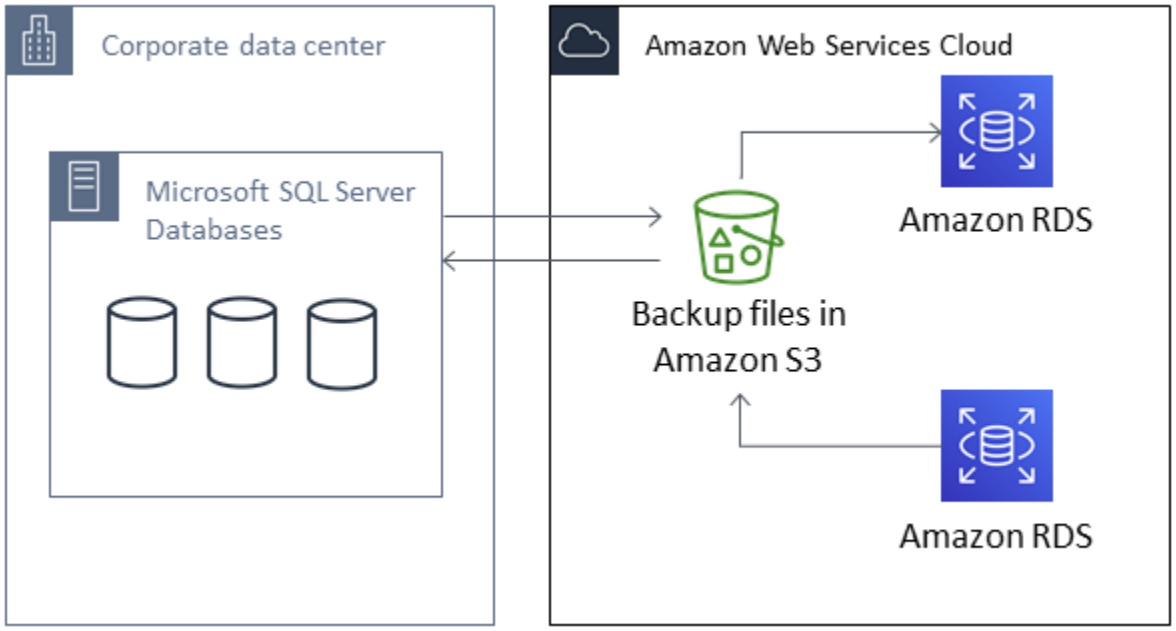
ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート

Amazon RDS では、完全バックアップファイル (.bak ファイル) を使用した Microsoft SQL Server データベースのネイティブバックアップおよび復元がサポートされています。RDS を使用すると、データベースサーバー上のローカルファイルシステムを使用せずに Amazon S3 に格納されているファイルにアクセスします。

例えば、ローカルサーバーから完全バックアップを作成し、それを S3 に保存してから、既存の Amazon RDS DB インスタンスに復元することができます。RDS からバックアップを作成して S3 に保存することで、どこへでも復元することが可能となります。

ネイティブバックアップと復元は、すべての AWS リージョンで、シングル AZ DB インスタンスおよびマルチ AZ DB インスタンス (リードレプリカを持つマルチ AZ DB インスタンスを含む) に対して使用できます。ネイティブバックアップおよび復元は、Amazon RDS でサポートされているすべてのエディションの Microsoft SQL Server で使用できます。

次の図は、サポートされるシナリオを示しています。



ネイティブの .bak ファイルを使用したデータベースのバックアップと復元は、通常は、最も素早いデータベースのバックアップと復元を実現します。ネイティブバックアップおよび復元を使用することには、他にも多くの利点があります。例えば、次の操作を実行できます。

- Amazon RDS との間でデータベースを移行できます。

- RDS for SQL Server DB インスタンス間でデータベースを移動します。
- データ、スキーマ、ストアドプロシージャ、トリガー、その他のデータベースコードを .bak ファイル内に移行します。
- DB インスタンス全体ではなく、1 つのデータベースをバックアップおよび復元できます。
- 開発、テスト、トレーニング、デモの目的でデータベースのコピーを作成できます。
- 災害対策の追加保護レイヤーとして、Amazon S3 を使用したバックアップファイルの保管および転送ができます。
- 透過的なデータ暗号化 (TDE) がオンになっているデータベースのネイティブバックアップを作成し、これらのバックアップをオンプレミスのデータベースに復元します。詳細については、「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。
- TDE がオンになっているオンプレミスデータベースのネイティブバックアップを RDS for SQL Server DB インスタンスに復元します。詳細については、「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。

目次

- 制限と推奨事項
- ネイティブバックアップおよび復元のセットアップ
  - ネイティブバックアップおよび復元用の IAM ロールの手動作成
- ネイティブバックアップおよび復元の使用
  - データベースのバックアップ
    - Usage
    - 例
  - データベースの復元
    - Usage
    - 例
  - ログの復元
    - Usage
    - 例
  - データベースの復元を終了する
    - Usage
  - 部分的に復元したデータベースの使用
    - 部分的に復元したデータベースの削除

        - 部分的に復元したデータベースのスナップショット復元とポイントインタイム復元の動作
    - タスクのキャンセル
        - Usage
    - タスクのステータスの追跡
        - Usage
        - 例
        - 応答
- バックアップファイルの圧縮
- トラブルシューティング
- 他の方法による SQL Server データのインポートとエクスポート
    - スナップショットを使用した RDS for SQL Server へのデータのインポート
        - データのインポート
            - スクリプトの生成とパブリッシュウィザード
            - インポートおよびエクスポートウィザード
            - 一括コピー
    - RDS for SQL Server からのデータのエクスポート
        - SQL Server インポートおよびエクスポートウィザード
        - SQL Server のスクリプトの生成とパブリッシュウィザードおよび bcp ユーティリティ
- Linux の BCP ユーティリティを使用してデータをインポートおよびエクスポートする
    - 前提条件
    - Linux での SQL Server コマンドラインツールのインストール
    - RDS for SQL Server からのデータのエクスポート
        - 基本的なエクスポート構文
        - エクスポートの例
    - RDS for SQL Server へのデータのインポート
        - 基本的なインポート構文
        - インポートの例
    - 一般的な BCP オプション
    - ベストプラクティスと考慮事項
    - 一般的な問題のトラブルシューティング

制限と推奨事項

ネイティブバックアップおよび復元を使用する際の制限事項を以下に示します。

- Amazon RDS DB インスタンスとは異なる AWS リージョンの Amazon S3 バケットにバックアップしたり、このバケットから復元したりすることはできません。
- 既存のデータベースと同じ名前のデータベースを復元することはできません。データベース名は一意です。
- あるタイムゾーンのバックアップファイルを、別のタイムゾーンに復元しないことを強くお勧めします。バックアップを特定のタイムゾーンから別のタイムゾーンに復元した場合は、タイムゾーンの変更がクエリとアプリケーションにもたらす影響を監査する必要があります。
- RDS for Microsoft SQL Server のサイズ制限は、ファイルあたり 5 TB です。大規模なデータベースのネイティブバックアップでは、マルチファイルバックアップを使用できます。
- S3 にバックアップできる最大データベースサイズは、DB インスタンスで使用可能なメモリ、CPU、I/O、ネットワークリソースによって異なります。データベースが大きくなるほど、バックアップエージェントが消費するメモリは多くなります。
- 同時に 10 以上のバックファイルから復元することはできません。
- 差分バックアップは、最後の完全バックアップに基づいています。差分バックアップを機能させるには、最後の完全バックアップと差分バックアップの間でスナップショットを作成することはできません。差分バックアップが必要だが、手動スナップショットや自動スナップショットが存在する場合は、差分バックアップを続行する前に別の完全バックアップを作成してください。
- ファイルの file_guid (一意の識別子) が NULL に設定されているデータベースでは、差分復元とログ復元はサポートされていません 。
- 最大 2 のバックアップまたは復元タスクを同時に実行できます。
- Amazon RDS では、SQL Server からネイティブログバックアップを実行できません。
- RDS では最大 64 TiB のデータベースのネイティブ復元をサポートしています。SQL Server Express Edition のデータベースのネイティブリストアは、10 GB に制限されています。
- メンテナンスウィンドウや、Amazon RDS がデータベースのスナップショットを作成しているときは、ネイティブバックを行うことはできません。ネイティブバックアップタスクが RDS の毎日のバックアップウィンドウと重複する場合、ネイティブバックアップタスクはキャンセルされます。
- マルチ AZ DB インスタンスでは、完全な復元モデルでバックアップされているデータベースのみを、ネイティブに復元することができます。

- トランザクション内でのネイティブバックアップと復元を目的とした RDS プロシージャの呼び出しはサポートされていません。
- 対称暗号化 AWS KMS key を使用してバックアップを暗号化します。Amazon RDS は非対称 KMS キーをサポートしていません。詳細については、AWS Key Management Service デベロッパーガイドの「非対称 KMS キーを作成する」を参照してください。
- ネイティブバックアップファイルは、「暗号化のみ」の暗号モードを使用して、指定された KMS キーで暗号化されます。暗号化されたバックアップファイルを復元するときは、「暗号化のみ」の暗号モードで暗号化されていることに注意してください。
- FILESTREAM ファイルグループを含むデータベースは復元できません。
- AWS KMS (SSE-KMS) による Amazon S3 サーバー側の暗号化は、バックアップストアドプロシージャに `@enable_bucket_default_encryption=1` を渡すと、S3 バケットのデフォルトの暗号化設定でサポートされます。デフォルトでは、復元は S3 オブジェクトのサーバー側の暗号化をサポートします。

  ストアドプロシージャに KMS キーを指定すると、ネイティブバックアップと復元は KMS キーを使用してクライアント側で暗号化および復号されます。AWS は、`@enable_bucket_default_encryption=0` の場合は SSE-S3 を使用して、`@enable_bucket_default_encryption=1` の場合は S3 バケットで設定されたデフォルトの暗号化キーを使用して、バックアップを S3 バケットに保存します。
- S3 アクセスポイントを使用する場合、アクセスポイントを RDS 内部 VPC を使用するように設定することはできません。
- 最高のパフォーマンスのために、ディレクトリバケット、またはリージョンで利用可能な場合にはディレクトリバケットのアクセスポイントを使用することをお勧めします。

バックアップファイルの作成、コピー、復元時にデータベースをオフラインにできる場合、ネイティブバックアップを使用して復元し、RDS に移行することをお勧めします。オンプレミスデータベースをオフラインにできない場合、AWS Database Migration Service を使用してデータベースを Amazon RDS に移行することをお勧めします。詳細については、「What is AWS Database Migration Service?」を参照してください。

ネイティブバックアップおよび復元は、クロスリージョンスナップショットコピー機能のデータ復元機能に代わるものではありません。Amazon RDS におけるクロスリージョンの災害対策のため、スナップショットコピーを使用してデータベーススナップショットを別の AWS リージョンにコピーしておくことをお勧めします。詳細については、「Amazon RDS の DB スナップショットのコピー」を参照してください。

ネイティブバックアップおよび復元のセットアップ

ネイティブバックアップおよび復元をセットアップするには、3 つのコンポーネントが必要です。

1. バックアップファイルを保存する Amazon S3 バケット。

   バックアップファイルに使用する S3 バケットを用意してから、RDS に移行するバックアップをアップロードする必要があります。Amazon S3 バケットが既にある場合はそれを使用できます。バケットがない場合、バケットを作成できます。または、SQLSERVER_BACKUP_RESTORE を使用して AWS マネジメントコンソール オプションを追加するときに新しいバケットの作成を選択することもできます。

   S3 を使用する方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」を参照してください。
2. バケットにアクセスするための AWS Identity and Access Management (IAM) ロール。

   IAM ロールが既にある場合はそれを使用できます。AWS マネジメントコンソール を使用して SQLSERVER_BACKUP_RESTORE オプションを追加する際に、新しい IAM ロールの作成を選択することもできます。または、手動で新しいロールを作成することもできます。

   手動で新しい IAM ロールを作成する場合は、次のセクションで示されている方法を使用します。既存の IAM ロールに信頼関係ポリシーとアクセス許可ポリシーをアタッチする場合は、同じ操作を行います。
3. DB インスタンスのオプショングループに追加された SQLSERVER_BACKUP_RESTORE オプション。

   DB インスタンスでネイティブバックアップおよび復元を有効にするには、DB インスタンスのオプショングループに SQLSERVER_BACKUP_RESTORE オプションを追加します。詳細と手順については、「SQL Server のネイティブバックアップおよび復元のサポート」を参照してください。

ネイティブバックアップおよび復元用の IAM ロールの手動作成

ネイティブバックアップおよび復元用の新しいIAM ロールを手動で作成したい場合、作成することが可能です。その場合、 Amazon RDS サービスから Amazon S3 バケットに許可を委任する役割を作成します。IAM ロールを作成するときは、信頼関係とアクセス権限ポリシーをアタッチします。信頼関係により、RDS はこのロールを引き受けることができます。許可ポリシーは、このロールが実行できるアクションを定義します。ロールの作成の詳細については、「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

ネイティブバックアップおよび復元機能では、このセクションの例と同様の信頼関係ポリシーおよびアクセス許可ポリシーを使用します。次の例では、すべてのサービスアカウントのエイリアスとしてサービスプリンシパル名 rds.amazonaws.com を使用します。他の例では、Amazon リソースネーム (ARN) を指定して、信頼ポリシーでアクセス許可を付与している別のアカウント、ユーザー、またはロールを特定します。

リソースベースの信頼関係では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与する特定のリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

両方のグローバル条件コンテキストキーを使用し、aws:SourceArn 値にアカウント ID を含めます。この場合は、aws:SourceAccount 値と aws:SourceArn 値のアカウントは、同じステートメントで使用する場合、同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

信頼関係では、ロールにアクセスするリソースの完全な ARN を持つ aws:SourceArn グローバル条件コンテキストキーを必ず使用してください。ネイティブバックアップおよび復元では、次の例に示すように、DB オプショングループと DB インスタンスの両方を含めるようにしてください。

Exampleネイティブバックアップおよび復元のためのグローバル条件コンテキストキーによる信頼関係の例

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [
```

```
 "arn:aws:rds:Region:0123456789:db:db_instance_identifier",
                        "arn:aws:rds:Region:0123456789:og:option_group_name"
                    ],
                    "aws:SourceAccount": "0123456789"
                }
            }
        }
    ]
}
```

次の例では、ARN を使用してリソースを指定しています。ARN の使用の詳細については、「Amazon リソースネーム (ARN)」を参照してください。

Example暗号化をサポートしないネイティブバックアップおよび復元のためのアクセス許可ポリシーの例

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObjectAttributes",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
```

```
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Example暗号化をサポートするネイティブバックアップおよび復元のアクセス許可ポリシー

バックアップファイルを暗号化する場合は、アクセス権限ポリシーに暗号化キーを含めます。暗号化キーの詳細については、AWS Key Management Service デベロッパーガイドの「開始方法」を参照してください。

> **Note**
>
> バックアップを暗号化するには、対称暗号化 KMS キーを使用する必要があります。Amazon RDS は非対称 KMS キーをサポートしていません。詳細については、AWS Key Management Service デベロッパーガイドの「非対称 KMS キーを作成する」を参照してください。
> IAM ロールは、KMS キーのキーユーザーおよびキー管理者であることも必要です。つまり、キーポリシーで指定する必要があります。詳細については、AWS Key Management Service デベロッパーガイドの「非対称 KMS キーを作成する」を参照してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToKey",
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Encrypt",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
    },
```

```
    {
      "Sid": "AllowAccessToS3",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
    },
    {
      "Sid": "GetS3Info",
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectAttributes",
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListMultipartUploadParts",
        "s3:AbortMultipartUpload"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

Example暗号化をサポートしないアクセスポイントを使用したネイティブバックアップおよび復元のアクセス許可ポリシー

S3 アクセスポイントを使用するために必要なアクションは、S3 バケットの場合と同じです。リソースパスは、S3 アクセスポイント ARN パターンと一致するように更新されます。

> **Note**
>
> RDS はプライベート VPC を発行しないため、アクセスポイントは [ネットワークオリジン: インターネット] を使用するように設定する必要があります。RDS インスタンスからの S3 トラフィックは、プライベート VPC を通過するため、パブリックインターネットを経由しません。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
                ],
            "Resource": [
            "arn:aws:s3:us-east-1:111122223333:accesspoint/amzn-s3-demo-ap",
            "arn:aws:s3:::underlying-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectAttributes",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
                ],
                "Resource": [
                "arn:aws:s3:us-east-1:111122223333:accesspoint/amzn-s3-demo-ap/
*",
                    "arn:aws:s3:::underlying-bucket/*"
                    ]
                }
            ]
}
```

Example暗号化をサポートしないディレクトリバケットのアクセスポイントを使用したネイティブバックアップおよび復元のアクセス許可ポリシー

ディレクトリバケットは汎用バケットとは異なるセッションベースの認可メカニズムを使用するため、ネイティブバックアップ復元に必要なアクセス許可はバケットレベルの「s3express:CreateSession」アクセス許可のみです。オブジェクトレベルのアクセスを設定するには、ディレクトリバケットのアクセスポイントを使用する必要があります。

> **Note**
>
> RDS はプライベート VPC を発行しないため、アクセスポイントは [ネットワークオリジン: インターネット] を使用するように設定する必要があります。RDS インスタンスからの S3 トラフィックは、プライベート VPC を通過するため、パブリックインターネットを経由しません。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action": "s3express:CreateSession",
        "Resource":
            [
                "arn:aws:s3express:us-east-1:111122223333:accesspoint/amzn-s3-
demo-accesspoint--use1-az6--xa-s3",
                "arn:aws:s3express:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket--use1-az6--x-s3"
            ]
        }
    ]
}
```

ネイティブバックアップおよび復元の使用

ネイティブバックアップおよび復元を有効に設定した後、使用を開始できます。最初に、Microsoft SQL Server データベースに接続し、Amazon RDS ストアドプロシージャを呼び出して作業を行います。データベースに接続する手順については、「SQL Server DB インスタンスへの接続」を参照してください。

ストアドプロシージャによっては、Amazon リソースネーム (ARN) を Amazon S3 バケットおよびファイルに指定する必要があります。ARN の形式は `arn:aws:s3:::bucket_name/file_name.extension` です。Amazon S3 には、ARN のアカウント番号または AWS リージョンは不要です。

オプションの KMS キーも指定する場合、キーの ARN の形式は `arn:aws:kms:region:account-id:key/key-id` となります。詳細については、「Amazon リソースネーム (ARN) と AWS のサービスの名前空間」を参照してください。バックアップを暗号化するには、対称暗号化 KMS キーを使用する必要があります。Amazon RDS は非対称 KMS キーをサポートしていません。詳細については、AWS Key Management Service デベロッパーガイドの「非対称 KMS キーを作成する」を参照してください。

> **Note**
>
> KMS キーを使用するかどうかにかかわらず、ネイティブバックアップおよび復元タスクでは、S3 にアップロードされたファイルに対して、SSE-S3 を介した Advanced Encryption Standard (AES) 256 ビット暗号化がデフォルトで有効になります。バックアップストアドプロシージャに `@enable_bucket_default_encryption=1` を渡すと、S3 バケットで設定されたデフォルトの暗号化キーが使用されます。

各ストアドプロシージャを呼び出す方法については、以下のトピックを参照してください。

- データベースのバックアップ
- データベースの復元
- ログの復元
- データベースの復元を終了する
- 部分的に復元したデータベースの使用
- タスクのキャンセル
- タスクのステータスの追跡

データベースのバックアップ

データベースをバックアップするには、rds_backup_database ストアドプロシージャを呼び出します。

> **Note**
>
> メンテナンスウィンドウが開いている間または Amazon RDS がスナップショットを作成している間は、データベースをバックアップできません。

Usage

```
exec msdb.dbo.rds_backup_database
 @source_db_name='database_name',
 @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name.extension',
 [@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
 [@overwrite_s3_backup_file=0|1],
 [@block_size=512|1024|2048|4096|8192|16384|32768|65536],
        [@max_transfer_size=n],
        [@buffer_count=n],
 [@type='DIFFERENTIAL|FULL'],
 [@number_of_files=n],
 [@enable_bucket_default_encryption=0|1];
```

以下のパラメータは必須です。

- @source_db_name – バックアップするデータベースの名前。
- @s3_arn_to_backup_to - バックアップに使用する Amazon S3 バケット、アクセスポイント、ディレクトリバケット、またはディレクトリバケットのアクセスポイントとバックアップファイル名を表示する ARN。

  ファイルは任意の拡張子を持つことができますが、通常は .bak が使用されます。アクセスポイント ARN は arn:aws:s3:*us-east-1*:*111122223333*:*access-point-name/object/key* の形式である必要があります。

以下のパラメータはオプションです。

- @kms_master_key_arn - 項目の暗号化に使用する対称暗号化 KMS キーの ARN。

  - デフォルトの暗号化キーは使用できません。デフォルトキーを使用すると、データベースはバックアップされません。
  - KMS キー識別子を指定しない場合、バックアップファイルは暗号化されません。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
  - KMS キーを指定すると、クライアント側の暗号化が使用されます。
  - Amazon RDS は非対称 KMS キーをサポートしていません。詳細については、AWS Key Management Service デベロッパーガイドの「非対称 KMS キーを作成する」を参照してください。
- `@overwrite_s3_backup_file` – 既存のバックアップファイルを上書きするかどうかを表す値。
  - `0` – 既存のファイルを上書きしません。この値はデフォルト値です。

    設定 `@overwrite_s3_backup_file` を 0 にすると、ファイルが既に存在している場合はエラーが返されます。
  - `1` – バックアップファイルではない場合でも、指定された名前を持つ既存のファイルを上書きします。
- `@type` – バックアップのタイプ。
  - `DIFFERENTIAL` – 差分バックアップを取ります。
  - `FULL` – 完全バックアップを取ります。この値はデフォルト値です。

  差分バックアップは、最後の完全バックアップに基づいています。差分バックアップを機能させるには、最後の完全バックアップと差分バックアップの間でスナップショットを作成することはできません。差分バックアップを取りたいがスナップショットが存在する場合、差分バックアップを続行する前に別の完全バックアップを作成してください。

  最後のフルバックアップまたはスナップショットは、以下の SQL クエリの例を使用して検索できます。

```
select top 1
database_name
,  backup_start_date
,  backup_finish_date
from    msdb.dbo.backupset
where   database_name='mydatabase'
and     type = 'D'
order by backup_start_date desc;
```

- @number_of_files – バックアップが分割される (チャンク) ファイルの数。最大数は 10 です。
  - マルチファイルバックアップは、完全バックアップと差分バックアップの両方でサポートされています。
  - 値 1 を入力するか、パラメータを省略すると、1 つのバックアップファイルが作成されます。

  ファイルに共通のプレフィックスを付けてから、そのプレフィックスにアスタリスクを付けます (*)。アスタリスクは、S3 ARN の *file_name* 部分のどこにでも使用できます。生成されたファイルのアスタリスクは、1-of-*number_of_files* で始まる一連の英数字文字列に置き換えられます。

  例えば、S3 ARN のファイル名が backup*.bak で @number_of_files=4 を設定した場合、生成されるバックアップファイルは backup1-of-4.bak、backup2-of-4.bak、backup3-of-4.bak、backup4-of-4.bak です。
  - いずれかのファイル名が既に存在し、@overwrite_s3_backup_file が 0 の場合は、エラーが返されます。
  - マルチファイルバックアップでは、S3 ARN の *file_name* 部分にアスタリスクを 1 つだけ含めることができます。
  - シングルファイルバックアップでは、S3 ARN の *file_name* 部分にアスタリスクをいくつでも含めることができます。アスタリスクは、生成されたファイル名から削除されません。
- @block_size – バックアップ処理の物理ブロックサイズをバイト単位で指定します。有効な値は 512、1024、2048、4096、8192、16384、32768、および 65536 です
- @max_transfer_size – バックアッププロセス中、I/O オペレーションごとに転送されるデータボリュームの上限 (バイト単位) を示す最大転送サイズ。有効な値は、65536 バイト (64 KB) から 4194304 バイト (4 MB) までの倍数です。
- @buffer_count – バックアッププロセスに使用する I/O バッファの合計数。
- @enable_bucket_default_encryption - S3 のサーバー側の暗号化に S3 バケットのデフォルトの暗号化設定を使用するかどうかを示す値。ディレクトリバケットは、この設定に関係なく、常にバケットのデフォルトの暗号化設定を使用します。
  - 0 – のサーバー側の暗号化では、SSE-S3 を介した Advanced Encryption Standard (AES) 256 ビット暗号化が使用されます。
  - 1 – サーバー側の暗号化は、S3 バケットで設定されたデフォルトの暗号化を使用します。

例

Example差分バックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup1.bak',
@overwrite_s3_backup_file=1,
@type='DIFFERENTIAL';
```

Exampleクライアント側の暗号化による完全バックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup1.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE',
@overwrite_s3_backup_file=1,
@type='FULL';
```

Exampleマルチファイルバックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@number_of_files=4;
```

Exampleマルチファイル差分バックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@type='DIFFERENTIAL',
@number_of_files=4;
```

Example暗号化によるマルチファイルバックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE',
```

```
@number_of_files=4;
```

Example S3 の上書きによるマルチファイルバックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@overwrite_s3_backup_file=1,
@number_of_files=4;
```

Exampleブロックサイズによるバックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@block_size=512;
```

Example@max_transfer_size と @buffer_count によるマルチファイルバックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@number_of_files=4,
@max_transfer_size=4194304,
@buffer_count=10;
```

Example@number_of_files パラメータを使用したシングルファイルバックアップ

この例では、backup*.bak という名前のバックアップファイルを生成します。

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@number_of_files=1;
```

Exampleサーバー側の暗号化による完全バックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
```

```
@s3_arn_to_backup_to='arn:aws:s3:::mybucket/backup*.bak',
@overwrite_s3_backup_file=1,
@type='FULL',
@enable_bucket_default_encryption=1;
```

Exampleアクセスポイントを使用した完全バックアップ

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/
object/backup1.bak',
@overwrite_s3_backup_file=1,
@type='FULL';
```

Exampleディレクトリバケットのアクセスポイントを使用したフルバックアップの

```
exec msdb.dbo.rds_backup_database
@source_db_name='mydatabase',
@s3_arn_to_backup_to='arn:aws:s3express:us-east-1:123456789012:accesspoint/my-access-
point--use1-az6--xa-s3/object/backup1.bak',
@overwrite_s3_backup_file=1,
@type='FULL';
```

データベースの復元

データベースを復元するには、rds_restore_database ストアドプロシージャを呼び出します。復元タスクが完了しデータベースが開くと、Amazon RDSによりデータベースの最初のスナップショットが作成されます。

Usage

```
exec msdb.dbo.rds_restore_database
 @restore_db_name='database_name',
 @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name.extension',
 @with_norecovery=0|1,
 [@keep_cdc=0|1],
 [@data_file_volume='D:|H:|I:|J:'],
 [@log_file_volume='D:|H:|I:|J:'],
 [@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
        [@block_size=512|1024|2048|4096|8192|16384|32768|65536],
        [@max_transfer_size=n],
```

```
        [@buffer_count=n],
[@type='DIFFERENTIAL|FULL'];
```

以下のパラメータは必須です。

- @restore_db_name – 復元するデータベースの名前。データベース名は一意です。既存のデータベースと同じ名前のデータベースを復元することはできません。
- @s3_arn_to_restore_from – Amazon S3 プレフィックスと、データベースの復元に使用するバックアップファイルの名前を示す ARN。
  - 単一ファイルのバックアップの場合は、ファイル名全体を入力します。
  - マルチファイルのバックアップの場合は、ファイルに共通のプレフィックスを付けてから、そのプレフィックスにアスタリスクを付けます (*)。
    - ディレクトリバケットを使用する場合、ディレクトリバケットの違いにより、ARN は /* で終わる必要があります。
  - @s3_arn_to_restore_from が空の場合は、次のエラーメッセージが返ります: S3 ARN prefix cannot be empty。

以下のパラメータは、差分復元には必須ですが、完全復元ではオプションです。

- @with_norecovery – 復元操作に使用する復元句。
  - 0 に設定して、RECOVERY で復元します。この場合、復元後にデータベースがオンラインになります。
  - 1 に設定して、NORECOVERY で復元します。この場合、復元タスクの完了後もデータベースが RESTORING 状態を保持します。このアプローチで、後から差分復元も実行することができます。
  - DIFFERENTIAL 復元は、0 または 1 を指定してください。
  - FULL 復元の場合、この値はデフォルトで 0 です。

以下のパラメータはオプションです。

- @keep_cdc – 復元されたデータベースに変更データキャプチャ (CDC) 設定を保持するかどうかを示します。KEEP_CDC を有効にするには 1 に、無効にするには 0 に設定します。デフォルト値は 0 です。
- @data_file_volume – データベースデータファイルのドライブ文字を指定します。デフォルト値は D: です。

- @log_file_volume – データベースログファイルのドライブ文字を指定します。デフォルト値は D: です。
- @kms_master_key_arn - バックアップファイルを暗号化した場合の、ファイルの復号に使用する KMS キー。

  KMS キーを指定すると、クライアント側の暗号化が使用されます。
- @type – 復元のタイプ。有効なタイプは、DIFFERENTIALとFULL です。デフォルト値は FULL です。
- @block_size – バックアップ処理の物理ブロックサイズをバイト単位で指定します。有効な値は 512、1024、2048、4096、8192、16384、32768、および 65536 です
- @max_transfer_size – バックアッププロセス中、I/O オペレーションごとに転送されるデータボリュームの上限 (バイト単位) を示す最大転送サイズ。有効な値は、65536 バイト (64 KB) から 4194304 バイト (4 MB) までの倍数です。
- @buffer_count – バックアッププロセスに使用する I/O バッファの合計数。

> **Note**
>
> 差分復元は、データベースが RESTORING 状態にあるか、NORECOVERY で復元するタスクが既に存在している必要があります。
> データベースがオンラインの場合、後から差分バックアップを復元することはできません。
> データベースに、RECOVERY を使用した復元中のタスクがある場合、復元タスクを提出することはできません。
> NORECOVERY と KEEP_CDC の両方を使用した完全な復元はサポートされていません。
> クロスリージョンリードレプリカを持つインスタンスでは、いずれのネイティブ復元もサポートされていません。
> サポートされている設定の場合、リードレプリカを持つマルチ AZ インスタンスでのデータベースの復元は、マルチ AZ インスタンスでのデータベースの復元に似ています。レプリカ上のデータベースを復元するために、追加のアクションを実行する必要はありません。

例

Example単一ファイルの復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
```

```
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak';
```

Exampleマルチファイルの復元

複数ファイル復元中のエラーを回避するために、すべてのバックアップファイルに同じプレフィックスがあり、他のファイルでそのプレフィックスが使用されていないことを確認します。

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup*';
```

Example RECOVERY を使用したデータベースの復元

以下の 3 つの例は、RECOVERY を使用した完全復元という同じタスクを実行します。

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak';
```

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
[@type='DIFFERENTIAL|FULL'];
```

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='FULL',
@with_norecovery=0;
```

Example暗号化を使用したデータベースの完全復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

Exampleブロックサイズによる復元

```
exec msdb.dbo.rds_restore_database
```

```
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@block_size=512;
```

Example@max_transfer_size と @buffer_count によるマルチファイル復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup*',
@max_transfer_size=4194304,
@buffer_count=10;
```

Example NORECOVERY を使用したデータベースの完全復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='FULL',
@with_norecovery=1;
```

Example NORECOVERY を使用した差分復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='DIFFERENTIAL',
@with_norecovery=1;
```

Example RECOVERY を使用した差分復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@type='DIFFERENTIAL',
@with_norecovery=0;
```

Exampleアクセスポイントを使用した RECOVERY によるデータベース全体の復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
```

```
@s3_arn_to_backup_to='arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/
object/backup1.bak',
@with_norecovery=0;
```

Example KEEP_CDC を使用したデータベースの復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@keep_cdc=1;
```

ログの復元

ログを復元するには、rds_restore_log ストアドプロシージャを呼び出します。

Usage

```
exec msdb.dbo.rds_restore_log
 @restore_db_name='database_name',
 @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/log_file_name.extension',
 [@kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id'],
 [@with_norecovery=0|1],
 [@keep_cdc=0|1],
 [@stopat='datetime'],
 [@block_size=512|1024|2048|4096|8192|16384|32768|65536],
        [@max_transfer_size=n],
        [@buffer_count=n];
```

以下のパラメータは必須です。

- @restore_db_name – 復元するログのデータベース名。
- @s3_arn_to_restore_from – ARN が、Amazon S3プレフィックスと、ログを復元する際に使用するログファイル名を表示します。ファイルは任意の拡張子を持つことができますが、通常は .trn が使用されます。

  @s3_arn_to_restore_from が空の場合は、次のエラーメッセージが返ります: S3 ARN prefix cannot be empty。

以下のパラメータはオプションです。

- @keep_cdc – 復元されたデータベースに変更データキャプチャ (CDC) 設定を保持するかどうかを示します。KEEP_CDC を有効にするには 1 に、無効にするには 0 に設定します。デフォルト値は 0 です。
- @kms_master_key_arn - ログを暗号化した場合の、ログの復号に使用する KMS キー。
- @with_norecovery – 復元操作に使用する復元句。この値のデフォルト値は1です。
  - 0 に設定して、RECOVERY で復元します。この場合、復元後にデータベースがオンラインになります。データベースがオンラインの場合、さらにログバックアップを復元することはできません。
  - 1 に設定して、NORECOVERY で復元します。この場合、復元タスクの完了後もデータベースが RESTORING 状態を保持します。このアプローチで、後からログ復元も実行することができます。
- @stopat – データベ - スが、指定の日付と時間の状態に復元されたこと (日付時間形式) を指定するための値。指定の日時以前に書き込まれた取引きログ記録のみが、データベースに適用されます。

  このパラメータを指定していない場合 (NULL)、完全なログが復元されます。
- @block_size – バックアップ処理の物理ブロックサイズをバイト単位で指定します。有効な値は 512、1024、2048、4096、8192、16384、32768、および 65536 です
- @max_transfer_size – バックアッププロセス中、I/O オペレーションごとに転送されるデータボリュームの上限 (バイト単位) を示す最大転送サイズ。有効な値は、65536 バイト (64 KB) から 4194304 バイト (4 MB) までの倍数です。
- @buffer_count – バックアッププロセスに使用する I/O バッファの合計数。

> **Note**
>
> ログ復元は、データベースが復元状態にあるか、NORECOVERY で復元するタスクが既に存在している必要があります。
> データベースがオンラインの場合、ログバックアップを復元することはできません。
> データベースに、RECOVERY を使用した復元中のタスクがある場合、ログ復元タスクを提出することはできません。

例

Exampleログの復元

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn';
```

Example暗号化を使用したログの復元

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@kms_master_key_arn='arn:aws:kms:us-east-1:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

Example NORECOVERY を使用したログの復元

以下の 2 つの例は、NORECOVERY を使用したログ復元という同じタスクを実行します。

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=1;
```

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn';
```

Exampleブロックサイズによる復元

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@block_size=512;
```

Example RECOVERY を使用したログの復元

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=0;
```

Example STOPAT 句を使用したログの復元

```
exec msdb.dbo.rds_restore_log
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/mylog.trn',
@with_norecovery=0,
@stopat='2019-12-01 03:57:09';
```

Example KEEP_CDC を使用したログの復元

```
exec msdb.dbo.rds_restore_database
@restore_db_name='mydatabase',
@s3_arn_to_restore_from='arn:aws:s3:::mybucket/backup1.bak',
@keep_cdc=1;
```

データベースの復元を終了する

データベースの最後の復元タスクを `@with_norecovery=1` を使用して実行した場合、データベースが RESTORING 状態になります。`rds_finish_restore`ストアドプロシージャを使用して、このデータベースを通常操作用に開きます。

Usage

```
exec msdb.dbo.rds_finish_restore @db_name='database_name';
```

> **Note**
>
> このアプローチを使用するには、実行中の復元タスクのない状態で、データベースが RESTORING 状態である必要があります。
> データベースの復元が終了したら、マスターログインを使用してください。または、NORECOVERY を使用して直近にデータベースを復元したユーザーログインを使用またはログします。

部分的に復元したデータベースの使用

部分的に復元したデータベースの削除

部分的に復元したデータベースを削除するには（RESTORING状態のまま）、`rds_drop_database` ストアドプロシージャを使用してください。

```
exec msdb.dbo.rds_drop_database @db_name='database_name';
```

> **Note**
>
> 復元の中断中または復元タスクが完了したデータベースに対し、DROP データベースリクエストを送信することはできません。
> データベースを削除するには、マスターログインを使用します。または、NORECOVERY を使用して直近にデータベースを復元したユーザーログインを使用またはログします。

部分的に復元したデータベースのスナップショット復元とポイントインタイム復元の動作

ソースインスタンス内の部分的に復元されたデータベース (RESTORING状態のまま) は、スナップショットの復元またはポイントインタイム復元中に対象のインスタンスから削除されます。

タスクのキャンセル

バックアップまたは復元タスクをキャンセルするには、rds_cancel_task ストアドプロシージャを呼び出します。

> **Note**
>
> FINISH_RESTORE タスクはキャンセルできません。

Usage

```
exec msdb.dbo.rds_cancel_task @task_id=ID_number;
```

以下のパラメータは必須です。

- @task_id – キャンセルするタスクの ID。rds_task_status を呼び出すことにより、タスク ID を取得できます。

タスクのステータスの追跡

バックアップおよび復元タスクのステータスを追跡するには、rds_task_status ストアドプロシージャを呼び出します。パラメータを何も指定しない場合、ストアドプロシージャによりすべての

タスクのステータスが返されます。タスクのステータスは、約 2 分ごとに更新されます。タスクの履歴は 36 日間保持されます。

Usage

```
exec msdb.dbo.rds_task_status
 [@db_name='database_name'],
 [@task_id=ID_number];
```

以下のパラメータはオプションです。

- @db_name – タスクのステータスを表示するデータベースの名前。
- @task_id – タスクのステータスを表示するタスクの ID。

例

Example特定タスクのステータスのリスト化

```
exec msdb.dbo.rds_task_status @task_id=5;
```

Example特定データベースおよびタスクのステータスのリスト化

```
exec msdb.dbo.rds_task_status
@db_name='my_database',
@task_id=5;
```

Example特定データベースのすべてのタスクおよびステータスのリスト化

```
exec msdb.dbo.rds_task_status @db_name='my_database';
```

Example現在のインスタンスのすべてのタスクおよびステータスのリスト化

```
exec msdb.dbo.rds_task_status;
```

応答

rds_task_status ストアドプロシージャは、次の列を返します。

列	説明
task_id	タスクの ID。

task_type	入力パラメータによるタスクタイプは以下の通りです。
	• バックアップタスク:
	  • BACKUP_DB – 完全データベース復元
	  • BACKUP_DB_DIFFERENTIAL – 差分データベースバックアップ
	• タスクの復元:
	  • RESTORE_DB – RECOVERY を使用した完全データベースの復元
	  • RESTORE_DB_NORECOVERY –NORECOVERY を使用した完全データベースの復元
	  • RESTORE_DB_DIFFERENTIAL – RECOVERYで差分データベースの復元
	  • RESTORE_DB_DIFFERENTIAL_NORECOVERY – NORECOVERY で差分データベースの復元
	  • RESTORE_DB_LOG – RECOVERY でのログの復元
	  • RESTORE_DB_LOG_NORECOVERY – NORECOVERY でログの復元
	• 復元を終了するタスク:
	  • FINISH_RESTORE – 復元を終了しデータベースを開きます
	
	以下の復元タスクが完了してデータベースが開くと、Amazon RDSが初期のスナップショットを作成します。

列	説明
	• RESTORE_DB • RESTORE_DB_DIFFERENTIAL • RESTORE_DB_LOG • FINISH_RESTORE
database_name	タスクが関連付けられているデータベースの名前。
% complete	タスクの進行状況の割合値。
duration (mins)	タスクにかかった時間 (分単位)。

列	説明
lifecycle	タスクのステータス。有効な状態には、以下が含まれます。 • CREATED – rds_backup_database または rds_restore_database を呼び出すとすぐ、タスクが作成されてステータスが CREATED に設定されます。 • IN_PROGRESS – バックアップまたは復元タスクが開始されると、ステータスが に設定されます。IN_PROGRESS ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。 • SUCCESS – バックアップまたは復元タスクが完了すると、ステータスが に設定されます。SUCCESS • ERROR – バックアップまたは復元タスクに失敗した場合、ステータスが に設定されます。ERROR工ラーの詳細については、task_info 列を参照してください。 • CANCEL_REQUESTED – rds_cancel_task を呼び出すとすぐに、タスクのステータスが CANCEL_REQUESTED に設定されます。 • CANCELLED – タスクが正常にキャンセルされると、タスクのステータスが に設定されます。CANCELLED
task_info	タスクに関する追加情報。 データベースのバックアップまたは復元中にエラーが発生した場合は、この列にエラーに関する情報が表示されます。発生する可能性がある工ラーのリストと軽減戦略については、「トラブルシューティング」を参照してください。
last_updated	タスクのステータスが最後に更新された日時。5% 進行するたびに、ステータスが更新されます。
created_at	タスクが作成された日時。

列	説明
S3_object_arn	Amazon S3プレフィックスを表す ARN とバックアップまたは復元したファイルの名前。
overwrite_s3_backup_file	バックアップタスクを呼び出すときに指定される @overwrite_s3_backup_file パラメータの値。詳細については、「データベースのバックアップ」を参照してください。
KMS_master_key_arn	(バックアップ時の) 暗号化および (復元時の) 復号に使用する KMS キーの ARN。
filepath	ネイティブバックアップおよびタスクの復元には適用されません。
overwrite_file	ネイティブバックアップおよびタスクの復元には適用されません。

バックアップファイルの圧縮

Amazon S3 バケットの容量を節約するために、バックアップファイルを圧縮できます。バックアップファイルの圧縮の詳細については、Microsoft ドキュメントの「バックアップの圧縮」を参照してください。

バックアップファイルの圧縮は、以下のデータベースエディションでサポートされています。

- Microsoft SQL Server Enterprise Edition
- Microsoft SQL Server Standard Edition

バックアップファイルの圧縮オプションを確認するには、次のコードを実行します。

```
exec rdsadmin.dbo.rds_show_configuration 'S3 backup compression';
```

バックアップファイルの圧縮を有効にするには、以下のコードを実行します。

```
exec rdsadmin.dbo.rds_set_configuration 'S3 backup compression', 'true';
```

バックアップファイルの圧縮を無効にするには、以下のコードを実行します。

```
exec rdsadmin.dbo.rds_set_configuration 'S3 backup compression', 'false';
```

トラブルシューティング

以下は、ネイティブ バックアップおよび復元を使用する場合に遭遇する可能性のある問題です。

問題	トラブルシューティングの推奨事項
データベースのバックアップ/復元オプションが有効になっていないか、有効化中です。後で再試行してください。	DB インスタンスに関連付けられた DB オプショングループにSQLSERVER_BACKUP_RESTORE オプションが追加されていることを確認します。詳細については、「ネイティブバックアップおよび復元オプションの追加」を参照してください。
EXECUTE アクセス許可は、オブジェクト「*rds_backup_database* 」、データベース「msdb」、スキーマ「dbo」で拒否されました。	ストアドプロシージャを実行するときは、マスターユーザーを使用していることを確認してください。マスターユーザーとしてログインした後でもこのエラーが発生した場合は、管理者ユーザーのアクセス許可の不一致が原因である可能性があります。マスターユーザーをリセットするには、AWS マネジメントコンソールを使用します。「Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット」を参照してください。
EXECUTE アクセス許可は、オブジェクト「*rds_restore_database* 」、データベース「msdb」、スキーマ「dbo」で拒否されました。	ストアドプロシージャを実行するときは、マスターユーザーを使用していることを確認してください。マスターユーザーとしてログインした後でもこのエラーが発生した場合は、管理者ユーザーのアクセス許可の不一致が原因である可能性があります。マスターユーザーをリセットするには、AWS マネジメントコンソールを使用します。「Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット」を参照してください。
アクセスが拒否されました	バックアップまたは復元プロセスは、バックアップファイルにアクセスできません。通常、この原因は以下のような問題にあります。 •

問題	トラブルシューティングの推奨事項
	間違ったバケットを参照している。間違った形式を使用してバケットを参照している。ARN を使用しないでファイル名を参照している。 • バケットファイルへのアクセス許可が正しくない。例えば、アクセスを試行しているアカウントとは別のアカウントでファイルが作成されている場合は、正しいアクセス許可を追加します。 • IAM ポリシーが正しくないか、不完全である。IAM ロールには、すべての必要な要素 (正しいバージョンなど) を含める必要があります。これらについては、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」で詳しく説明しています。
BACKUP DATABASE WITH COMPRESSION は <edition_name> エディションではサポートされていません	バックアップファイルの圧縮は、Microsoft SQL Server Enterprise Edition と Standard Edition でのみサポートされています。 詳細については、「バックアップファイルの圧縮」を参照してください。
キー <ARN> が存在しません	暗号化されたバックアップを復元しようとしましたが、有効な暗号化キーを指定しませんでした。暗号化キーを確認し、再試行してください。 詳細については、「データベースの復元」を参照してください。

問題	トラブルシューティングの推奨事項
正しいタイプでタスクを再発行し、プロパティを上書きしてください	データベースをバックアップする際に、既存のファイル名を指定して上書きのプロパティを false に設定すると、保存オペレーションは失敗します。このエラーを修正するには、既存のファイル名以外の名前を指定するか、上書きのプロパティを true に設定します。 詳細については、「データベースのバックアップ」を参照してください。 また、データベースを復元しようとして、間違えて rds_backup_database ストアドプロシージャを呼び出した可能性もあります。この場合は、代わりに rds_restore_database ストアドプロシージャを呼び出します。 詳細については、「データベースの復元」を参照してください。 データベースを復元するために rds_restore_database ストアドプロシージャを呼び出した場合は、有効なバックアップファイルの名前を指定したことを確認してください。 詳細については、「ネイティブバックアップおよび復元の使用」を参照してください。
RDS インスタンスと同じリージョンにあるバケットを指定してください	Amazon RDS DB インスタンスとは異なる AWS リージョンの Amazon S3 バケットにバックアップしたり、このバケットから復元したりすることはできません。Amazon S3 レプリケーションを使用すると、正しい AWS リージョンにバックアップファイルをコピーできます。 詳細については、Amazon S3 ドキュメントの「クロスリージョンレプリケーション」を参照してください。
指定されたバケットは存在しません	バケットとファイルの正しい ARN を正しい形式で指定したこと確認してください。 詳細については、「ネイティブバックアップおよび復元の使用」を参照してください。

問題	トラブルシューティングの推奨事項
ユーザー <ARN> にはリソース <ARN> で <kms action> を実行する権限がありません	暗号化されたオペレーションをリクエストしましたが、指定した AWS KMS のアクセス権限が正しくありません。適切なアクセス権限を持っていることを確認してください。持っていない場合は、追加してください。 詳細については、「ネイティブバックアップおよび復元のセットアップ」を参照してください。
Restore タスクは、10 個を超えるバックアップファイルから復元できません。一致するファイルの数を減らして、もう一度お試しください。	復元しようとしているファイル数を減らします。必要に応じて、個々のファイルを大きくすることができます。
データベース「*database_name* 」は既に存在します。大文字と小文字またはアクセントのみが異なる 2 つのデータベースは使用できません。別のデータベース名を選択します。	既存のデータベースと同じ名前のデータベースを復元することはできません。データベース名は一意です。

他の方法による SQL Server データのインポートとエクスポート

次に、自分の Microsoft SQL Server データを Amazon RDS にインポートするためのスナップショット使用に関する情報を、確認することができます。また、SQL Server を実行する RDS DB インスタンスからデータをエクスポートするためのスナップショット使用に関する情報を確認することができます。

シナリオでサポートされている場合は、ネイティブバックアップおよび復元機能を使用して Amazon RDS との間でデータを移動する方が簡単です。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

> **Note**
>
> Amazon RDS for Microsoft SQL Server では、msdb データベースへのデータのインポートがサポートされていません。

スナップショットを使用した RDS for SQL Server へのデータのインポート

スナップショットを使用して SQL Server DB インスタンスにデータをインポートするには

1. DB インスタンスを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
2. アプリケーションの送信先 DB インスタンスへのアクセスを停止します。

   データをインポートしている間 DB インスタンスにアクセスできないようにすると、データ転送が速くなります。さらに、データのロード中に他のアプリケーションが同時に DB インスタンスに書き込むことができなければ、競合について心配する必要がなくなります。何か問題が発生し、以前のデータベーススナップショットにロールバックする必要がある場合、失われる変更内容は、インポートされたデータのみです。問題の解決後に、このデータを再インポートすることができます。

   DB インスタンスへのアクセス制御の詳細については、「セキュリティグループによるアクセス制御」を参照してください。
3. ターゲットデータベースのスナップショットを作成します。

   ターゲットデータベースに既にデータが設定されている場合は、データをインポートする前にデータベースのスナップショットを作成することをお勧めします。データのインポートで何か問

題があった場合や、変更を破棄する必要がある場合は、スナップショットを使用してデータベースを前の状態に復元できます。データベースのスナップショットの詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

> **Note**
>
> データベースのスナップショットを作成するときは、バックアップ処理中の間 (数ミリ秒)、データベースの I/O オペレーションは一時停止されます。

4. ターゲットデータベースの自動バックアップを無効にします。

   ターゲット DB インスタンスの自動バックアップを無効にすると、データのインポート中のパフォーマンスが向上します。これは、自動バックアップが無効化されると、Amazon RDS がトランザクションを記録しなくなるためです。ただし、考慮しなければならないことがあります。ポイントインタイムリカバリを実行するには、自動化バックアップが必要です。そのため、データのインポート中は、データベースを指定のところまで復元することはできません。さらに、保持することを選択する場合を除き、DB インスタンスに作成されていた自動バックアップはすべて消去されます。

   自動バックアップを保持するよう選択すると、誤ってデータを削除することを防止できます。また Amazon RDS では、それぞれの自動バックアップとともにデータベースインスタンスのプロパティも保存されます。これにより、復元が容易になります。このオプションを使用することで、データベースインスタンスを (削除後でも) バックアップ保存期間内の指定した時点に復元できます。削除したデータベースインスタンスの自動バックアップは、アクティブなデータベースインスタンスの自動バックアップと同様、指定したバックアップ期間の終了時に自動的に削除されます。

   以前のスナップショットを使用してデータベースを復元することもできます。また、お客様が作成したスナップショットは、引き続き使用できます。自動バックアップの詳細については、「バックアップの概要」を参照してください。

5. 外部キーの制約を無効にします (該当する場合)。

   外部キーの制約を無効にする必要がある場合は、次のスクリプトを使用できます。

```
--Disable foreign keys on all tables
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;
```

```
    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;

    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' NOCHECK CONSTRAINT
 ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END

    CLOSE table_cursor;
    DEALLOCATE table_cursor;

    GO
```

6. インデックスを削除します (該当する場合)。
7. トリガーを無効にします (該当する場合)。

   トリガーを無効にする必要がある場合は、以下のスクリプトを使用できます。

```
--Disable triggers on all tables
    DECLARE @enable BIT = 0;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';

    OPEN trigger_cursor;
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

    WHILE @@FETCH_STATUS = 0 BEGIN
      IF @enable = 1
         SET @cmd = 'ENABLE ';
      ELSE
         SET @cmd = 'DISABLE ';

      SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
 dbo.'+QUOTENAME(@table)+' ';
      EXEC (@cmd);
```

```
      FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
    END

    CLOSE trigger_cursor;
    DEALLOCATE trigger_cursor;

    GO
```

8. 送信先 DB インスタンスにインポートする必要のあるログインについて、送信元 SQL Server インスタンスに問い合わせます。

   SQL Server では、ログインとパスワードを master データベースに保存します。Amazon RDS では master データベースへのアクセス権を付与しないため、送信先 DB インスタンスに直接ログインとパスワードをインポートすることはできません。その代わり、元の SQL サーバーインスタンスの master データベースに問い合わせ、データ定義言語 (DDL) を作成しなければなりません。このファイルには、最終的な DB インスタンスに加えたいすべてのログイン情報とパスワードが含まれるようにします。またファイルには、転送したロールメンバーシップと許可も含まれます。

   master データベースへのクエリ実行については、Microsoft サポート技術情報の「SQL Server のインスタンス間でログインおよびパスワードを転送する」を参照してください。

   スクリプトの出力は、送信先 DB インスタンスで実行できる別のスクリプトです。サポート技術情報の記事にあるスクリプトには、次のコードがあります。

   ```
   p.type IN
   ```

   p.type が表示された場合はいつでも、次のコードを代わりに使用します。

   ```
   p.type = 'S'
   ```

9. 「データのインポート」の方法を使用してデータをインポートします。
10. アプリケーションにターゲット DB インスタンスへのアクセス権を付与します。

    データのインポートが完了したら、インポート中に止めていた DB インスタンスへのアプリケーションのアクセスを許可できます。DB インスタンスへのアクセス制御の詳細については、「セキュリティグループによるアクセス制御」を参照してください。
11. ターゲット DB インスタンスの自動バックアップを有効にします。

    自動バックアップの詳細については、「バックアップの概要」を参照してください。

12. 外部キーの制約を有効化します。

    以前に外部キーの制約を無効にした場合は、ここで以下のスクリプトを使用してそれらを有効にすることができます。

```
--Enable foreign keys on all tables
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;

    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' CHECK CONSTRAINT ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END

    CLOSE table_cursor;
    DEALLOCATE table_cursor;
```

13. インデックスを有効にします (該当する場合)。
14. トリガーを有効にします (該当する場合)。

    トリガーを無効化していた場合は、以下のスクリプトでそれらを有効にすることができます。

```
--Enable triggers on all tables
    DECLARE @enable BIT = 1;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';

    OPEN trigger_cursor;
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

    WHILE @@FETCH_STATUS = 0 BEGIN
      IF @enable = 1
```

```
         SET @cmd = 'ENABLE ';
      ELSE
         SET @cmd = 'DISABLE ';

      SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
dbo.'+QUOTENAME(@table)+' ';
      EXEC (@cmd);
      FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
   END

   CLOSE trigger_cursor;
   DEALLOCATE trigger_cursor;
```

データのインポート

Microsoft SQL Server Management Studio は、Microsoft SQL Server の Express エディションを除くすべてのエディションに含まれるグラフィカル SQL Server クライアントです。SQL Server Management Studio Express は、Microsoft から無料でダウンロードできます。このダウンロードを確認するには、「Microsoft ウェブサイト」を参照してください。

> **Note**
>
> SQL Server Management Studio は、Windows ベースのアプリケーションとしてのみ使用できます。

SQL Server Management Studio には、SQL Server DB インスタンスにデータをインポートするときに便利な、次のツールが含まれています。

- スクリプトの生成とパブリッシュウィザード
- インポートおよびエクスポートウィザード
- 一括コピー

スクリプトの生成とパブリッシュウィザード

スクリプトの生成とパブリッシュウィザードは、データベースのスキーマ、データそのもの、または両方を含むスクリプトを作成します。ローカル SQL サーバーのデプロイ内のデータベースに、スクリプトを作成することができます。その後スクリプトを実行し、そこに含まれる情報を Amazon RDS DB インスタンスに転送することができます。

> **Note**
>
> 1 GiB 以上のデータベースについては、データベーススキーマのみをスクリプトするほうが効率的です。次に、インポートおよびエクスポートウィザードを使用するか、SQL Server の一括コピー機能を使用して、データを転送します。

スクリプトの生成とパブリッシュウィザードの詳細については、Microsoft SQL Server のドキュメントを参照してください。

ウィザードでは、特に [Set Scripting Options] ページの高度なオプションに注意を払い、スクリプトに含める必要のあるものがすべて選択されていることを確認します。例えば、デフォルトでは、データベースのトリガーはスクリプトに含まれていません。

スクリプトが生成されて保存されると、SQL Server Management Studio を使用して DB インスタンスに接続し、スクリプトを実行することができます。

インポートおよびエクスポートウィザード

インポートおよびエクスポートウィザードは、特別な Integration Services パッケージを作成します。これを使用して、ローカルの SQL Server データベースから転送先 DB インスタンスにデータをコピーすることができます。ウィザードでは、転送先 DB インスタンスにコピーするテーブルおよびタプルをフィルタできます。

> **Note**
>
> インポートおよびエクスポートウィザードは、大規模なデータセットには有効に機能しますが、ローカルデプロイメントからリモートにデータをエクスポートするには、最速の方法とはいえません。より速い方法については、SQL Server の一括コピー機能を検討できます。

インポートおよびエクスポートウィザードの詳細については、Microsoft SQL Server のドキュメントを参照してください。

ウィザードでは、[Choose a Destination] ページで、次の操作を行います。

- [Server Name] に、DB インスタンスのエンドポイント名を入力します。
- サーバー認証のモードの場合は、[Use SQL Server Authentication] を選択します。

- [User name] と [Password] に、DB インスタンス用に作成したマスターユーザーの認証情報を入力します。

一括コピー

SQL Server の一括コピー機能は、ソースデータベースから DB インスタンスにデータをコピーするための効率的な手段です。一括コピーは、指定したデータを ASCII ファイルなどのデータファイルに書き込みます。その後、一括コピーを再度実行して、そのファイルのコンテンツを転送先の DB インスタンスに書き込みます。

このセクションでは、SQL Server のすべてのエディションに含まれる bcp ユーティリティを使用します。一括インポートおよびエクスポートオペレーションの詳細については、Microsoft SQL Server のドキュメントを参照してください。

> **Note**
>
> 一括コピーを使用する前に、データベーススキーマを転送先の DB インスタンスにインポートする必要があります。これを行うには、このトピックですでに説明したスクリプトの生成とパブリッシュウィザードが最適なツールです。

以下のコマンドは、ローカルの SQL Server インスタンスに接続されます。そして指定されたテーブルのタブ区切りファイルを、既存の SQL Server デプロイメントの C:\ ルートディレクトリに生成します。テーブルは完全修飾名で指定し、テキストファイルはコピーされるテーブルと同じ名前になります。

```
bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost -U username -
P password -b 10000
```

前述のコードには、以下のオプションがあります。

- -n 一括コピーではコピーするデータのネイティブデータ型を使用するように指定します。
- -S bcp ユーティリティが、接続する SQL Server インスタンスを指定します。
- -U SQL Server インスタンスにログインするアカウントのユーザー名を指定します。
- -P で指定したユーザーのパスワードを指定します。-U
- -b 一括インポートするデータの行数を指定します。

> **Note**
>
> インポートの状況にとって重要な他のパラメータがある場合があります。例えば、ID 値に関連する -E パラメータを必要とする場合があります。詳細については、Microsoft SQL Server のドキュメントの bcp ユーティリティのコマンドライン構文の詳細な説明を参照してください。

例えば、store という名前のデータベースには、デフォルトスキーマ dbo を使用し、customers という名前のテーブルが含まれているとします。ユーザーアカウント admin とパスワード insecure で、customers テーブルの 10,000 行を customers.txt という名前のファイルにコピーします。

```
bcp store.dbo.customers out C:\customers.txt -n -S localhost -U admin -P insecure -b
 10000
```

データファイルの作成後、類似のコマンドを使用して、自分の DB インスタンスにデータをアップロードすることができます。事前に、ターゲット DB インスタンスにデータベースとスキーマを作成します。次に、in引数で出力ファイルを指定する代わりに、out引数を使用して入力ファイルを指定します。localhost を使用してローカル SQL Server インスタンスを指定する代わりに、DB インスタンスのエンドポイントを指定します。1433 以外のポートを使用する場合は、それも指定します。ユーザー名とパスワードは、DB インスタンスのマスターユーザーとパスワードです。構文は次のとおりです。

```
bcp dbname.schema_name.table_name
     in C:\table_name.txt -n -S endpoint,port -U master_user_name -
P master_user_password -b 10000
```

前の例を続行するために、マスターユーザー名を admin、パスワードを insecure とします。DB インスタンスのエンドポイントは rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com で、ポート 4080 を使用します。コマンドは次のとおりです。

```
bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-
east-1.rds.amazonaws.com,4080 -U admin -P insecure -b 10000
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

RDS for SQL Server からのデータのエクスポート

次のいずれかのオプションを選択して、RDS for SQL Server DB インスタンスからデータをエクスポートします。

- 完全バックアップファイル (.bak) を使用したネイティブデータベースのバックアップ – .bak ファイルを使用したデータベースのバックアップはかなり最適化されているため、通常はデータを最もすばやくエクスポートできます。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。
- SQL Server のインポートとエクスポートウィザード – 詳細については、「SQL Server インポートおよびエクスポートウィザード」を参照してください。
- SQL Server のスクリプトの生成とパブリッシュウィザードおよび bcp ユーティリティ – 詳細については、「SQL Server のスクリプトの生成とパブリッシュウィザードおよび bcp ユーティリティ」を参照してください。

SQL Server インポートおよびエクスポートウィザード

SQL Server インポートおよびエクスポートウィザードを使用して、RDS for SQL Server DB インスタンスから別のデータストアに 1 つ以上のテーブル、ビュー、クエリをコピーできます。この選択は、ターゲットデータストアが SQL Server でない場合に最適です。詳細については、SQL Server のドキュメントの「SQL Server インポートおよびエクスポートウィザード」を参照してください。

SQL Server のインポートおよびエクスポートウィザードは、Microsoft SQL Server Management Studio の一部として使用できます。このグラフィカル SQL Server クライエントは、Express エディションを除くすべての Microsoft SQL Server エディションに含まれます。SQL Server Management Studio は、Windows ベースのアプリケーションとしてのみ使用できます。SQL Server Management Studio Express は、Microsoft から無料でダウンロードできます。このダウンロードを確認するには、「Microsoft ウェブサイト」を参照してください。

SQL Server インポートおよびエクスポートウィザードを使用してデータをエクスポートするには

1. SQL Server Management Studio で、RDS for SQL Server DB インスタンスに接続します。この操作の詳細については、「SQL Server DB インスタンスへの接続」を参照してください。
2. [Object Explorer] で、[Databases] を展開し、ソースデータベースのコンテクスト (右クリック) メニューを開き、[Tasks] を選択してから、[Export Data] を選択します。ウィザードが表示されます。
3. [Choose a Data Source] ページで、次の作業を行います。
   a. [Data source] で、[**SQL Server Native Client 11.0**] を選択します。

      > **Note**
      >
      > SQL Server Native Client 11.0 (SQLNCLI11) は Microsoft によって完全に廃止され、SQL Server Management Studio (SSMS) 19 以降で削除されました。SSMS 19 以降を使用している場合は、データソースと送信先の両方で代わりに **Microsoft OLE DB Driver for SQL Server** を選択します。

   b. [Server name] ボックスに、RDS for SQL Server DB インスタンスのエンドポイントが表示されていることを確認します。
   c. [Use SQL Server Authentication] を選択します。[User name] および [Password] で、マスターユーザーの名前と DB インスタンスのパスワードを入力します。
   d. [Database] ボックスに、データのエクスポート元のデータベースが表示されていることを確認します。
   e. [Next (次へ)] を選択します。
4. [Choose a Destination] ページで、次の作業を行います。
   a. [Destination] で、[**SQL Server Native Client 11.0**] を選択します。

      > **Note**
      >
      > 他のターゲットデータソースも使用できます。例えば、.NET Framework データプロバイダー、OLE DB プロバイダー、SQL Server Native Client プロバイダー、ADO.NET プロバイダー、Microsoft Office Excel、Microsoft Office Access、フラットファイルソースを使用できます。これらのデータソースのいずれかをター

ゲットとして選択する場合、リマインダーの手順 4 は省略してください。次の接続状態の詳細は、SQL Server ドキュメントの [変換先の選択] を参照してください。

b. [Server name] で、ターゲット SQL Server DB インスタンスのサーバー名を入力します。

c. 適切な認証タイプを選択します。必要に応じてユーザー名とパスワードを入力します。

d. [Database] でターゲットデータベース名を選択するか、[New] を選択して、エクスポートされたデータを格納する新しいデータベースを作成します。

[新規作成] を選択した場合は、SQL Server ドキュメントの「データベースの作成」で、指定するデータベース情報の詳細を確認してください。

e. [Next (次へ)] を選択します。

5. [Table Copy or Query] ページで、[Copy data from one or more tables or views] または [Write a query to specify the data to transfer] を選択します。[Next (次へ)] を選択します。

6. [Write a query to specify the data to transfer] を選択した場合は、[Provide a Source Query] ページが表示されます。SQL クエリを入力するか、貼り付けた後、[Parse] を選択してクエリを検証します。クエリを検証したら、[Next] を選択します。

7. [Select Source Tables and Views] ページで、次の作業を行います。

a. エクスポートするテーブルやビューを選択するか、指定したクエリが選択されていることを確認します。

b. [Edit Mappings] を選択し、データベースと列のマッピング情報を指定します。詳細については、SQL Server ドキュメントの「列マッピング」を参照してください。

c. (オプション) エクスポートされるデータのプレビューを表示するには、テーブル、ビュー、またはクエリを選択し、[Preview] を選択します。

d. [Next (次へ)] を選択します。

8. [Run Package] ページで、[Run immediately] が選択されていることを確認します。[Next (次へ)] を選択します。

9. [Complete the Wizard] ページで、データのエクスポートの詳細が想定したとおりになっていることを確認します。[Finish] を選択します。

10. [The execution was successful] ページで、[Close] を選択します。

SQL Server のスクリプトの生成とパブリッシュウィザードおよび bcp ユーティリティ

SQL Server のスクリプトの生成とパブリッシュウィザードを使用すると、データベース全体のスクリプトまたは選択したオブジェクトのみのスクリプトを作成できます。ターゲット SQL Server DB

インスタンスにこれらのスクリプトを実行して、スクリプト化されたオブジェクトを再作成できます。次に、bcp ユーティリティを使用して、選択したオブジェクトのデータをターゲット DB インスタンスに一括エクスポートすることができます。この選択肢は、データベース全体 (テーブル以外のオブジェクトを含む) または大量のデータを、2 つの SQL Server DB インスタンス間で移動する場合に最適です。bcp のコマンドライン構文の詳細については、Microsoft SQL Server ドキュメントの「bcp ユーティリティ」を参照してください。

SQL Server スクリプトの生成とパブリッシュウィザードは、Microsoft SQL Server Management Studio の一部として使用できます。このグラフィカル SQL Server クライエントは、Express エディションを除くすべての Microsoft SQL Server エディションに含まれます。SQL Server Management Studio は、Windows ベースのアプリケーションとしてのみ使用できます。SQL Server Management Studio Express は、Microsoft から無料でダウンロードできます。

SQL Server のスクリプトの生成とパブリッシュウィザードおよび bcp ユーティリティを使用してデータをエクスポートするには

1. SQL Server Management Studio で、RDS for SQL Server DB インスタンスに接続します。この操作の詳細については、「SQL Server DB インスタンスへの接続」を参照してください。
2. [Object Explorer] で、[Databases] ノードを展開し、スクリプト化するデータベースを選択します。
3. SQL Server ドキュメントの「スクリプトの生成とパブリッシュウィザード」の手順に従ってスクリプトファイルを作成します。
4. SQL Server Management Studio で、ターゲット SQL Server DB インスタンスに接続します。
5. [オブジェクトエクスプローラー] でターゲット SQL Server DB インスタンスが選択された状態で、[ファイル] メニューの [開く] を選択します。続いて、[ファイル] を選択し、スクリプトファイルを開きます。
6. データベース全体をスクリプトした場合、スクリプト内の CREATE DATABASE ステートメントを見直してください。データベースが希望の場所に希望のパラメータで作成されていることを確認します。詳細については、SQL Server のドキュメントの「CREATE DATABASE」を参照してください。
7. スクリプトでデータベースユーザーを作成している場合は、それらのユーザーのサーバーログインがターゲット DB インスタンスに存在するかどうかを確認します。作成されていない場合、それらのユーザーのログインを作成します。作成しないと、データベースユーザーを作成するためにスクリプト化したコマンドが失敗します。詳細については、SQL Server ドキュメントの「ログインの作成」を参照してください。

8. [SQL Editor] メニューの [!Execute] を選択してスクリプトファイルを実行し、データベースオブジェクトを作成します。スクリプトが終了したら、すべてのデータベースオブジェクトが想定したとおりに存在していることを確認します。
9. bcp ユーティリティを使用して、RDS for SQL Server DB インスタンスからファイルにデータをエクスポートします。コマンドプロンプトを開き、次のコマンドを入力します。

```
bcp database_name.schema_name.table_name out data_file -n -S aws_rds_sql_endpoint -
U username -P password
```

前述のコードには、以下のオプションがあります。

- table_name は、ターゲットデータベースに再作成し、データを挿入するテーブルの 1 つの名前です。
- data_file は、作成されるデータファイルのフルパスと名前です。
- -n 一括コピーではコピーするデータのネイティブデータ型を使用するように指定します。
- -S は、エクスポート元の SQL Server DB インスタンスを指定します。
- -U は、SQL Server DB インスタンスに接続するときに使用するユーザー名を指定します。
- -P で指定したユーザーのパスワードを指定します。-U

コマンドの例を以下に示します。

```
bcp world.dbo.city out C:\Users\JohnDoe\city.dat -n -S sql-jdoe.1234abcd.us-
west-2.rds.amazonaws.com,1433 -U JohnDoe -P ClearTextPassword
```

エクスポートするすべてのテーブルのデータファイルが作成されるまで、この手順を繰り返します。

10. SQL Server ドキュメントの「データの一括インポートの準備」の手順に従って、ターゲット DB インスタンスでデータを一括インポートする準備を行います。
11. SQL Server ドキュメントの「一括インポート操作と一括エクスポート操作について」で説明されているパフォーマンスやその他の注意点を検討した後で、使用する一括インポートの方法を決定します。
12. bcp ユーティリティを使用して作成したデータファイルから、データを一括インポートします。そのためには、ステップ 11。での決定に応じて、SQL Server ドキュメントの「bcp ユーティリティを使用した一括データのインポートとエクスポート」または「BULK INSERT または OPENROWSET(BULK...) を使用した一括データのインポート」手順に従います。

Linux の BCP ユーティリティを使用してデータをインポートおよびエクスポートする

BCP (一括コピープログラム) ユーティリティを使用すると、RDS for SQL Server DB インスタンスとデータファイル間で大量のデータを効率的に転送できます。Linux 環境の BCP を使用して一括データオペレーションを実行することができるため、データ移行、ETL プロセス、定期的なデータ転送に役立ちます。

BCP は、ファイルから SQL Server テーブルへのデータのインポートと、SQL Server テーブルからファイルへのデータのエクスポートの両方をサポートしています。これは、区切りテキストファイルを含むさまざまな形式で構造化データを転送するのに特に効果的です。

前提条件

Linux の RDS for SQL Server DB インスタンスで BCP を使用する前に、以下が整っていることを確認してください。

- RDS for SQL Server DB インスタンスへのネットワーク接続を備えた Linux 環境
- Linux システムにインストールされている、以下の Microsoft SQL Server コマンドラインツール。
  - sqlcmd - SQL Server コマンドラインクエリツール
  - bcp - 一括コピープログラムユーティリティ
- RDS for SQL Server DB インスタンスの有効な認証情報
- SQL Server ポート (通常は 1433) での接続を許可するようにセキュリティグループを介して設定されたネットワークアクセス
- 実行する操作に適切なデータベースアクセス許可

Linux での SQL Server コマンドラインツールのインストール

Linux から BCP を使用するには、Microsoft SQL Server コマンドラインツールをインストールする必要があります。特定の Linux ディストリビューションのインストール手順の詳細については、次の Microsoft ドキュメントを参照してください。

- 「sqlcmd と bcp SQL Server コマンドラインツールを Linux にインストールする」
- bcp ユーティリティ - BCP ユーティリティの完全なリファレンス

インストール後、以下を実行して PATH でツールが使用可能であることを確認します。

```
bcp -v
sqlcmd -?
```

RDS for SQL Server からのデータのエクスポート

BCP を使用して、RDS for SQL Server DB インスタンスから Linux システムのファイルにデータをエクスポートできます。これは、バックアップの作成、データ分析、または移行のためのデータの準備に役立ちます。

基本的なエクスポート構文

BCP を使用してデータをエクスポートするための基本的な構文は次のとおりです。

```
bcp database.schema.table out output_file -S server_name -U username -P password
 [options]
```

コードの説明は以下のとおりです。

- `database.schema.table` - 完全修飾テーブル名
- `output_file` - 出力ファイルのパスと名前。
- `server_name` - RDS for SQL Server エンドポイント
- `username` - データベースのユーザー名
- `password` - データベースのパスワード

エクスポートの例

次の例では、`sales` データベース内の `customers` という名前のテーブルからデータをエクスポートします。

```
bcp sales.dbo.customers out /home/user/customers.txt \
    -S mydb.cluster-abc123.us-east-1.rds.amazonaws.com \
    -U admin \
    -P mypassword \
    -c \
    -t "|" \
    -r "\n"
```

このコマンドは、

- customers テーブルからデータをエクスポートします
- 出力を /home/user/customers.txt に保存します
- 文字形式 (-c) を使用します
- パイプ (|) をフィールド区切り文字として使用します (-t "|")
- 改行を行区切り文字として使用します (-r "\n")

RDS for SQL Server へのデータのインポート

BCP を使用して、Linux システムのファイルから RDS for SQL Server DB インスタンスにデータをインポートできます。これは、データ移行、テストデータのロード、または定期的なデータ更新に役立ちます。

基本的なインポート構文

BCP を使用してデータをインポートするための基本的な構文は次のとおりです。

```
bcp database.schema.table in input_file -S server_name -U username -P password
 [options]
```

コードの説明は以下のとおりです。

- database.schema.table - 完全修飾送信先テーブル名
- input_file - 入力ファイルのパスと名前。
- server_name - RDS for SQL Server エンドポイント
- username - データベースのユーザー名
- password - データベースのパスワード

インポートの例

次の例では、ファイルから customers という名前のテーブルにデータをインポートします。

```
bcp sales.dbo.customers in /home/user/customers.txt \
    -S mydb.cluster-abc123.us-east-1.rds.amazonaws.com \
    -U admin \
    -P mypassword \
    -c \
    -t "|" \
```

```
    -r "\n" \
    -b 1000
```

このコマンドは、

- データを customers テーブルにインポートします
- /home/user/customers.txt からデータを読み取ります
- 文字形式 (-c) を使用します
- パイプ (|) をフィールド区切り文字として使用します (-t "|")
- 改行を行区切り文字として使用します (-r "\n")
- 1,000 行のバッチでデータを処理します (-b 1000)

一般的な BCP オプション

BCP には、データのフォーマットと転送動作を制御するためのオプションが多数用意されています。次の表に、一般的に使用されるオプションを示します。

オプション	説明
-c	すべての列に文字データ型を使用する
-n	ネイティブデータベースデータ型を使用する
-t	フィールド区切り文字を指定する (デフォルトはタブ)
-r	行区切り文字を指定する (デフォルトは改行)
-b	一括オペレーションのバッチサイズを指定する
-F	エクスポートまたはインポートする最初の行を指定する
-L	エクスポートまたはインポートする最後の行を指定する
-e	拒否された行をキャプチャするエラーファイルを指定する
-f	データフォーマット用の形式ファイルを指定する
-q	オブジェクト名に引用符で囲まれた識別子を使用する

ベストプラクティスと考慮事項

Linux の RDS for SQL Server で BCP を使用する場合は、次のベストプラクティスを考慮してください。

- バッチ処理の使用 – 大規模なデータセットの場合は、-b オプションを使用してデータをバッチ処理します。これはパフォーマンスを向上させ、エラー復旧を改善します。
- エラーの適切な処理 – -e オプションを使用して、分析のために エラー情報と拒否された行を別のファイルにキャプチャします。
- 適切なデータ形式の選択 – クロスプラットフォームの互換性には文字形式 (-c) を使用し、送信元と送信先の両方が SQL Server の場合にパフォーマンスを向上させるにはネイティブ形式 (-n) を使用します。
- 認証情報の保護 – パスワードをコマンドラインに直接記述することは避けてください。適切なアクセス許可を持つ環境変数または設定ファイルの使用を検討してください。
- 小さなデータセットでのテスト – 大量のデータを処理する前に、小さなデータセットで BCP コマンドをテストして、フォーマットと接続を検証します。
- ネットワーク接続のモニタリング – 特に大規模なデータ転送の場合、安定したネットワーク接続を確保します。長時間実行されるオペレーションには、`screen` や `tmux` などのツールの使用を検討してください。
- データ整合性の検証 – データ転送後、行数とサンプルデータを検証して、オペレーションが正常に完了したことを確認します。

一般的な問題のトラブルシューティング

次の表は、Linux から BCP を使用する際に発生する可能性がある一般的な問題とその解決策を示しています。

問題	ソリューション
接続タイムアウトまたはネットワークエラー	Amazon RDS エンドポイント、セキュリティグループ設定、およびネットワーク接続を確認します。SQL Server ポート (通常は 1433) が Linux システムからアクセス可能であることを確認します。

問題	ソリューション
認証の失敗	ユーザー名とパスワードを確認します。データベースユーザーに、実行している操作に対する適切なアクセス許可があることを確認します。
データフォーマットエラー	フィールドと行の区切り文字を確認します。データ形式が BCP が期待するものと一致していることを確認します。複雑なデータ構造にはフォーマットファイルを使用します。
アクセス拒否エラー	データベースユーザーに、ターゲットテーブルのインポート用の INSERT アクセス許可、またはエクスポート用の SELECT アクセス許可があることを確認します。
大きなファイル処理の問題	-b オプションでバッチ処理を使用します。パフォーマンスとエラー復旧を向上させるために、大きなファイルを小さなチャンクに分割することを検討してください。
文字エンコードの問題	データファイルで互換性のある文字エンコードを使用していることを確認します。文字形式には -c オプションを使用するか、適切なコードページを指定します。

Amazon RDS での Microsoft SQL Server 用のリードレプリカの使用

リードレプリカは通常、Amazon RDS の DB インスタンス間でレプリケーションを設定するために使用します。リードレプリカの概要については、「DB インスタンスのリードレプリカの操作」を参照してください。

このセクションでは、Amazon RDS for SQL Server でのリードレプリカの使用に関する特定の情報を確認することができます。

- データベース ユーザーとオブジェクトを SQL Server リードレプリカと同期する
- SQL Server リードレプリカの問題のトラブルシューティング

SQL Server リードレプリカの設定

DB インスタンスがレプリケーション用のソース DB インスタンスとして機能するには、ソース DB インスタンスで自動バックアップを有効にする必要があります。そのためには、バックアップ保持期間の値を 0 以外の値に設定します。このタイプのデプロイを設定すると、自動バックアップが有効になります。

SQL Server リードレプリカを作成する際、プライマリ DB インスタンスを停止する必要はありません。Amazon RDS では、サービスを中断することなく、ソース DB インスタンスとリードレプリカに必要なパラメータとアクセス許可を設定できます。ソース DB インスタンスのスナップショットが作成され、このスナップショットがリードレプリカになります。リードレプリカの削除時にも停止は発生しません。

1 つのソース DB インスタンスから最大 15 つのリードレプリカを作成できます。レプリケーションを効率的に実行するには、各リードレプリカにソース DB インスタンスと同程度のコンピューティングリソースとストレージリソースを設定することをお勧めします。ソースの DB インスタンスをスケールした場合は、リードレプリカもスケールする必要があります。

ソース DB インスタンスとそのすべてのリードレプリカの SQL Server DB エンジンバージョンは同じである必要があります。Amazon RDS では、メンテナンスウィンドウに関係なく、リードレプリカがアップグレードされた後すぐにプライマリがアップグレードされます。DB エンジンバージョンのアップグレードの詳細については、「Microsoft SQL Server DB エンジンのアップグレード」を参照してください。

リードレプリカでソースからの変更を受信して適用できるように、十分なコンピューティングリソースとストレージリソースが必要です。リードレプリカのコンピューティング、ネットワーク、またはストレージがリソースの容量に達すると、リードレプリカはソースからの変更の受信または適用を停止します。リードレプリカのストレージや CPU リソースは、そのソースや他のリードレプリカとは独立して変更することができます。

リードレプリカを作成する方法については、「リードレプリカの作成」を参照してください。

SQL Server でのリードレプリカの制限

Amazon RDS の SQL Server リードレプリカには、次の制限が適用されます。

- リードレプリカは、SQL Server Enterprise Edition (EE) エンジンでのみ使用することができます。
- リードレプリカは、SQL Server バージョン 2016 – 2025 で使用できます。
- 1 つのソース DB インスタンスから最大 15 つのリードレプリカを作成できます。ソース DB インスタンスのリードレプリカが 5 つを超えると、レプリケーションで遅延が発生する場合があります。
- リードレプリカは、4 つ以上の vCPU を持つ DB インスタンスクラスで実行されている DB インスタンスでのみ使用できます。
- リードレプリカは、インスタンスクラスタイプと可用性モードに応じて最大 100 のデータベースをサポートします。データベースをリードレプリカに自動的にレプリケートするには、ソース DB インスタンスでデータベースを作成する必要があります。レプリケートするデータベースを個別に選択することはできません。詳細については、「Microsoft SQL Server DB インスタンスの制限」を参照してください。
- リードレプリカからデータベースを削除することはできません。データベースを削除するには、`rds_drop_database` ストアドプロシージャを使用してソース DB インスタンスから削除します。詳細については、「Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースの削除」を参照してください。
- ソース DB インスタンスが透過的なデータ暗号化 (TDE) を使用してデータを暗号化すると、リードレプリカも TDE を自動的に設定します。

  ソース DB インスタンスが KMS キーを使用してデータを暗号化すると、同じリージョンのリードレプリカは同じ KMS キーを使用します。クロスリージョンリードレプリカの場合、リードレプリカの作成時にリードレプリカのリージョンから KMS キーを指定する必要があります。リードレプリカの KMS キーを変更することはできません。
- リードレプリカは、作成元のアベイラビリティーゾーンに関係なく、ソース DB インスタンスと同じタイムゾーンと照合順序を持ちます。

- Amazon RDS for SQL Server では、次の機能はサポートされていません。
  - リードレプリカのバックアップ保持
  - リードレプリカからのポイントインタイムリカバリ
  - リードレプリカの手動スナップショット
  - マルチ AZ リードレプリカ
  - リードレプリカのリードレプリカの作成
  - リードレプリカへのユーザーログインの同期
- Amazon RDS for SQL Server は、ソース DB インスタンスとそのリードレプリカの間の高いレプリカラグを軽減するために介入することはありません。ソース DB インスタンスとそのリードレプリカのサイズが、コンピューティング能力とストレージの観点から、運用負荷に合わせて適切に設定されていることを確認してください。
- AWS GovCloud (米国東部) と AWS GovCloud (米国西部) リージョンの間ではレプリケーションできますが、AWS GovCloud (US) Regions との間ではレプリケーションできません。

RDS for SQL Server レプリカのオプションに関する考慮事項

RDS for SQL Server レプリカを作成する前に、次の要件、制限、推奨事項を確認してください。

- SQL Server レプリカがソース DB インスタンスと同じリージョンにある場合は、そのレプリカがソース DB インスタンスと同じオプショングループに属していることを確認してください。出典オプショングループまたは出典オプショングループメンバーシップへの変更はレプリカに反映されます。これらの変更は、レプリカのメンテナンスウィンドウに関係なく、出典 DB インスタンスに適用された後すぐにレプリカに適用されます。

  オプショングループの詳細については、「オプショングループを使用する」を参照してください。
- SQL Server クロスリージョンレプリカを作成する場合、Amazon RDS により、そのための専有オプショングループが作成されます。

  SQL Server クロスリージョンレプリカを、その専有オプショングループから削除することはできません。他の DB インスタンスで、SQL Server クロスリージョンレプリカの専有オプショングループを使用することはできません。

  以下のオプションは、レプリケートオプションです。SQL Server クロスリージョンレプリカにレプリケートオプションを追加するには、そのオプションをソース DB インスタンスのオプショングループに追加します。オプションは、すべての出典 DB インスタンスのレプリカにもインストールされます。

- TDE

以下のオプションは、レプリケートされないオプションです。専用オプショングループから、レプリケートされないオプションを追加または削除できます。

- MSDTC
- SQLSERVER_AUDIT
- クロスリージョンリードレプリカで SQLSERVER_AUDIT オプションを有効にするには、クロスリージョンリードレプリカの専用オプショングループとソースインスタンスのオプショングループに SQLSERVER_AUDIT オプションを追加します。SQL Server クロスリージョンリードレプリカのソースインスタンスに SQLSERVER_AUDIT オプションを追加することで、ソースインスタンスのクロスリージョンリードレプリカごとにサーバーレベル監査オブジェクトとサーバーレベルの監査仕様を作成できます。クロスリージョンリードレプリカにアクセスして、完成した監査ログを Amazon S3 バケットにアップロードできるようにするには、専用オプショングループに SQLSERVER_AUDIT オプションを追加し、オプション設定を行います。監査ファイルのターゲットとして使用している Amazon S3 バケットは、クロスリージョンリードレプリカと同じリージョンにある必要があります。各クロスリージョンリードレプリカの SQLSERVER_AUDIT オプションのオプション設定を個別に変更して、それぞれがそれぞれのリージョンの Amazon S3 バケットにアクセスできるようにします。

次のオプションは、リードレプリカではサポートされていません。

- SSRS
- SSAS
- SSIS

次のオプションは、クロスリージョンリードレプリカで一部サポートされています。

- SQLSERVER_BACKUP_RESTORE
- SQL Server クロスリージョンレプリカのソース DB インスタンスには SQLSERVER_BACKUP_RESTORE オプションがありますが、すべてのクロスリージョンレプリカを削除するまで、ソース DB インスタンスでネイティブ復元を実行できません。クロスリージョンレプリカの作成中は、既存のネイティブリストアタスクはすべてキャンセルされます。SQLSERVER_BACKUP_RESTORE オプションを専用のオプショングループに追加することはできません。

  ネイティブバックアップと復元の詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

SQL Server クロスリージョンリードレプリカを昇格するとき、昇格されたレプリカは、オプションの管理を含め、その他の SQL Server DB インスタンスと同じように動作します。オプショングループの詳細については、「オプショングループを使用する」を参照してください。

データベース ユーザーとオブジェクトを SQL Server リードレプリカと同期する

新しく作成されたリードレプリカには、リードレプリカの作成時にプライマリ DB インスタンスに存在するログイン、カスタムサーバーロール、SQL エージェントジョブ、またはその他のサーバーレベルのオブジェクトが存在することが期待されます。ただし、リードレプリカの作成後にプライマリ DB インスタンスで作成されたサーバーレベルのオブジェクトは自動的にレプリケートされないため、リードレプリカで手動で作成する必要があります。

データベースユーザーは、プライマリ DB インスタンスからリードレプリカに自動的にレプリケートされます。リードレプリカデータベースは読み取り専用モードであるため、データベースユーザーのセキュリティ識別子 (SID) をデータベースで更新することはできません。そのため、リードレプリカに SQL ログインを作成するときは、そのログインの SID がプライマリ DB インスタンスの対応する SQL ログインの SID と一致していることを確認することが不可欠です。SQL ログインの SID を同期しないと、リードレプリカのデータベースにアクセスできなくなります。SQL Server は Active Directory から SID を取得するため、Windows Active Directory (AD) 認証ログインでこの問題は発生しません。

プライマリ DB インスタンスからの SQL ログインをリードレプリカに同期するには

1. プライマリ DB インスタンスに接続します。
2. プライマリ DB インスタンスに新しい SQL ログインを作成します。

```
USE [master]
GO
CREATE LOGIN TestLogin1
WITH PASSWORD = 'REPLACE WITH PASSWORD';
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

3. データベースに SQL ログイン用の新しいデータベースユーザーを作成します。

```
USE [REPLACE WITH YOUR DB NAME]
GO
CREATE USER TestLogin1 FOR LOGIN TestLogin1;
GO
```

4. プライマリ DB インスタンスで新しく作成された SQL ログインの SID を確認します。

```
SELECT name, sid FROM sys.server_principals WHERE name =  'TestLogin1';
```

5. リードレプリカに接続します。新しい SQL ログインを作成します。

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'REPLACE WITH PASSWORD', SID=REPLACE WITH
 sid FROM STEP #4;
```

または、リードレプリカデータベースにアクセスできる場合は、孤立したユーザーを次のように修正できます。

1. リードレプリカに接続します。
2. データベース内で孤立したユーザーを特定します。

```
USE [REPLACE WITH YOUR DB NAME]
GO
EXEC sp_change_users_login 'Report';
GO
```

3. 孤立したデータベースユーザーに SQL ログインを作成します。

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'REPLACE WITH PASSWORD', SID=REPLACE WITH
 sid FROM STEP #2;
```

例:

```
CREATE LOGIN TestLogin1 WITH PASSWORD = 'TestPa$$word#1',
 SID=0x1A2B3C4D5E6F7G8H9I0J1K2L3M4N5O6P;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

SQL Server リードレプリカの問題のトラブルシューティング

Amazon CloudWatch のレプリケーションのラグをモニタリングするには、Amazon RDS ReplicaLag メトリクスを表示します。レプリケーションのラグタイムについては、「リードレプリケーションのモニタリング」を参照してください。

レプリケーションラグが長すぎる場合は、次のクエリを使用してラグに関する情報を取得できます。

```
SELECT AR.replica_server_name
     , DB_NAME (ARS.database_id) 'database_name'
     , AR.availability_mode_desc
     , ARS.synchronization_health_desc
     , ARS.last_hardened_lsn
     , ARS.last_redone_lsn
     , ARS.secondary_lag_seconds
FROM sys.dm_hadr_database_replica_states ARS
INNER JOIN sys.availability_replicas AR ON ARS.replica_id = AR.replica_id
--WHERE DB_NAME(ARS.database_id) = 'database_name'
ORDER BY AR.replica_server_name;
```

Amazon RDS for Microsoft SQL Server のマルチ AZ 配置

マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。予定されたデータベースメンテナンスまたは予期しないサービス障害時に、Amazon RDS は自動的に最新のセカンダリ DB インスタンスにフェイルオーバーします。この機能により、データベースオペレーションを手動介入なしで速やかに再開できます。プライマリインスタンスおよびスタンバイインスタンスは、同じエンドポイントを使用します。このエンドポイントの物理的なネットワークアドレスは、フェイルオーバープロセスの一環としてセカンダリレプリカに移行します。フェイルオーバーが発生した場合、アプリケーションを再構成する必要はありません。

Amazon RDS は、Microsoft SQL Server で SQL Server データベースミラーリング (DBM)、Always On 可用性グループ (AG)、またはブロックレベルレプリケーションによるマルチ AZ 配置をサポートしています。Amazon RDS は、マルチ AZ 配置のヘルス状態をモニタリングし、維持します。問題が発生した場合、RDS は異常のある DB インスタンスを自動的に修正し、同期を再確立して、フェイルオーバーをスタートします。フェイルオーバーは、スタンバイとプライマリが完全に同期している場合にのみスタートします。何も管理する必要はありません。

SQL Server マルチ AZ をセットアップすると、RDS は DBM、AG、またはブロックレベルレプリケーションを使用するよう、インスタンス上のすべてのデータベースを自動的に設定します。DBM または AG を設定すると、Amazon RDS では、プライマリ、モニタリング、およびセカンダリ DB インスタンスが処理されます。ブロックレベルレプリケーションの場合、RDS はプライマリ DB インスタンスとセカンダリ DB インスタンスを処理します。設定は自動的に行われるため、RDS はデプロイする SQL Server のバージョンに基づいて、DBM、Always On AG、またはブロックレベルレプリケーションを選択します。

Amazon RDS では、SQL Server の以下のバージョンおよびエディションの常時稼働 AG によるマルチ AZ がサポートされます。

- SQL Server 2025:
  - Standard Edition
  - Enterprise Edition
- SQL Server 2022:
  - Standard Edition
  - Enterprise Edition
- SQL Server 2019:
  - Standard Edition 15.00.4073.23 以降

  - Enterprise Edition
- SQL Server 2017:
  - Standard Edition 14.00.3401.7 以降
  - Enterprise Edition 14.00.3049.1 以降
- SQL Server 2016: Enterprise Edition 13.00.5216.0 以降

Amazon RDS では、上述のバージョンを除き、SQL Server の以下のバージョンおよびエディションの DBM によるマルチ AZ がサポートされます。

- SQL Server 2019: Standard Edition 15.00.4043.16
- SQL Server 2017: Standard および Enterprise Edition
- SQL Server 2016: Standard および Enterprise Edition

Amazon RDS は、SQL Server 2022 Web Edition 16.00.4215.2 以降のブロックレベルレプリケーションでマルチ AZ をサポートしています。

> **Note**
>
> 16.00.4215.2 以降で作成された新しい DB インスタンスのみが、ブロックレベルレプリケーションによるマルチ AZ 配置をサポートします。既存の SQL Server 2022 Web Edition インスタンスには、次の制限が適用されます。
>
> - バージョン 16.00.4215.2 の既存のインスタンスの場合、ブロックレベルレプリケーションを有効にするには、同じマイナーバージョン以上の新しいインスタンスにスナップショットを復元する必要があります。
> - 古いマイナーバージョンを持つ SQL Server 2022 Web インスタンスは、マイナーバージョン 16.00.4215.2 以降にアップグレードして、ブロックレベルレプリケーションを有効にすることができます。

次のストアドプロシージャを使用して、SQL Server DB インスタンスがシングル AZ、DBM によるマルチ AZ、Always On AG、またはブロックレベルのレプリケーションによるマルチ AZ のいずれであるかを特定できます。

```
EXEC msdb.dbo.rds_get_high_availability_option;
```

戻り値

出力値	説明
`Multi-AZ (AlwaysOn)`	DB インスタンスは高可用性のために Always On 可用性グループを使用します。
`Multi-AZ (Mirroring)`	DB インスタンスは、高可用性のために SQL Server データベースミラーリングを使用します。
`Multi-AZ (Block level replication)`	DB インスタンスは、高可用性のためにブロックレベルのレプリケーションを使用します。
`Single-AZ`	DB インスタンスには、高可用性設定がありません。

Microsoft SQL Server DB インスタンスへのマルチ AZ の追加

AWS マネジメントコンソールを使用して新しい SQL Server DB インスタンスを作成する場合、データベースミラーリング (DBM)、Always On AG、またはブロックレベルレプリケーションによるマルチ AZ を追加できます。これを行うには、[マルチ AZ 配置] から [はい (ミラーリング/常時/ブロックレベルレプリケーション)] を選択します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

コンソールを使用して既存の SQL Server DB インスタンスを変更する場合、[DB インスタンスを変更] ページの [マルチ AZ 配置] から [はい (ミラーリング/常時/ブロックレベルレプリケーション)] を選択して、DBM、AG、またはブロックレベルレプリケーションによるマルチ AZ を追加できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> DB インスタンスが Always On 可用性グループ (AG) ではなくデータベースのミラーリング (DBM) を実行している場合は、マルチ AZ を追加する前に、インメモリ最適化を無効にする必要が生じることがあります。DB インスタンスで SQL Server 2016、または 2017 Enterprise Edition が実行され、インメモリ最適化が有効になっている場合は、マルチ AZ を追加する前に DBM でのインメモリ最適化を無効にします。
> DB インスタンスで SQL Server Web Edition の AG またはブロックレベルレプリケーションを実行している場合、このステップは必要ありません。

Microsoft SQL Server DB インスタンスからのマルチ AZ の削除

AWS マネジメントコンソールを使用して既存の SQL Server DB インスタンスを変更する場合、DBM、AG、またはブロックレベルレプリケーションを使用してマルチ AZ を削除できます。これを行うには、[DB インスタンスを変更] ページで [マルチ AZ 配置] から [いいえ (ミラーリング/常時/ブロックレベルレプリケーション)] を選択します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Microsoft SQL Server マルチ AZ 配置の制限、注意事項、および推奨事項

以下は、RDS for SQL Server DB インスタンスでマルチ AZ 配置を使用する際の、いくつかの制限事項です。

- クロスリージョンマルチ AZ はサポートされていません。
- マルチ AZ 配置の RDS for SQL Server DB インスタンスの停止はサポートされていません。
- データベースの読み取りアクティビティを受け入れるように、セカンダリ DB インスタンスを設定することはできません。
- Always On 可用性グループ (AG) を備えたマルチ AZ は、メモリ内最適化をサポートします。
- Always On 可用性グループ (AG) を使用するマルチ AZ は、可用性グループリスナーでの Kerberos 認証をサポートしていません。これは、リスナーにサービスプリンシパル名 (SPN) がないためです。
- ブロックレベルレプリケーションを使用するマルチ AZ は現在、SQL Server Web Edition インスタンスでのみサポートされています。
- SQL Server マルチ AZ 配置内の SQL Server DB インスタンス上のデータベースの名前を変更することはできません。そのようなインスタンスのデータベースの名前を変更する必要がある場合、まず DB インスタンスのマルチ AZ を無効にし、それから名前を変更します。最後に、DB インスタンスのマルチ AZ を再び有効にします。
- 完全な復旧モデルを使用してバックアップされているマルチ AZ DB インスタンスのみ復元できます。
- マルチ AZ 配置は、SQL Server エージェントジョブの数が 10,000 に制限されています。

  制限の引き上げが必要な場合は、サポート に連絡して緩和をリクエストしてください。AWS サポート センターのページを開き、必要に応じてサインインし、[ケースの作成] を選択します。[Service Limit increase] (サービス制限の緩和) を選択します。フォームに入力して送信します。

- SQL Server マルチ AZ 配置内の SQL Server DB インスタンス上にオフラインのデータベースを配置することはできません。
- ボリュームメトリクスは、ブロックレベルレプリケーションを使用するインスタンスのセカンダリホストでは使用できません。

以下は、RDS for SQL Server DB インスタンスでマルチ AZ 配置を使用する際の、いくつかの注意事項です。

- Amazon RDS は常時稼働 AG 可用性グループのリスナーエンドポイントを公開します。エンドポイントはコンソールに表示され、`DescribeDBInstances` API オペレーション によってエンドポイントフィールドのエントリとして返されます。
- Amazon RDS は 可用性グループのマルチサブネットフェイルオーバーをサポートします。
- 仮想プライベートクラウド (VPC) 内の SQL Server DB インスタンスで SQL Server のマルチ AZ を使用するには、まず、少なくとも 2 つの異なるアベイラビリティーゾーンにサブネットを持つ DB サブネットグループを作成します。次に、その DB サブネットグループを SQL Server DB インスタンスのプライマリレプリカに割り当てます。
- マルチ AZ 配置にするために DB インスタンスが変更されている場合、変更中は、[変更中] のステータスになります。Amazon RDS によりスタンバイが作成され、プライマリ DB インスタンスのバックアップが作成されます。このプロセスが完了した後で、プライマリ DB インスタンスのステータスが [利用可能] になります。
- マルチ AZ 配置では、すべてのデータベースが同じノードにあります。プライマリホストのデータベースがフェイルオーバーする場合は、すべての SQL Server データベースが 1 つのアトミックユニットとしてスタンバイホストにフェイルオーバーします。Amazon RDS により新しい正常なホストがプロビジョニングされ、異常なホストに置き換わります。
- DBM、AG、またはブロックレベルレプリケーションを使用したマルチ AZ は、単一のスタンバイレプリカをサポートします。
- マルチ AZ 配置では、RDS for SQL Server は SQL Server ログインを作成して Always On AG またはデータベースミラーリングを許可します。RDS が作成するログインのパターンは、`db_<dbiResourceId>_node1_login`、`db_<dbiResourceId>_node2_login`、`db_<dbiResou` です。
- RDS for SQL Server は、リードレプリカへのアクセスを許可する SQL Server ログインを作成します。RDS が作成するログインのパターンは、`db_<readreplica_dbiResourceId>_node_login` です。
- 同期的データレプリケーションのため、1 つのアベイラビリティーゾーン内のスタンダード DB インスタンスのデプロイと比較した場合、レイテンシーが長くなる可能性があります。

- フェイルオーバー時間は、復旧プロセスの完了までにかかる時間の影響を受けます。大量のトランザクションがあると、フェイルオーバー時間はより長くなります。
- SQL Server マルチ AZ 配置では、フェイルオーバー再起動でプライマリ DB インスタンスのみを再起動します。フェイルオーバー後、プライマリ DB インスタンスは新しいセカンダリ DB インスタンスになります。マルチ AZ インスタンスのパラメータは更新されない可能性があります。フェイルオーバーなしの再起動の場合、プライマリ DB インスタンスとセカンダリ DB インスタンスの両方が再起動し、再起動後にパラメータが更新されます。DB インスタンスが応答しない場合は、フェイルオーバーなしで再起動することをお勧めします。

次の表は、RDS for SQL Server マルチ AZ 配置において、データベースミラーリング (DBM) または Always On 可用性グループ (AG) を使用する場合と、ブロックレベルのレプリケーションを使用する場合とで、サーバーレベルのオブジェクトがどのようにレプリケートされるかを比較したものです。

オブジェクトタイプ別のレプリケーション動作

オブジェクト	データベースミラーリング/Always On AG	ブロックレベルのレプリケーション
ログイン	はい – Amazon RDS によってレプリケートされます。`DEFAULT_DATABASE` プロパティはレプリケートされません。DBM または Always On AG インスタンスでログインを作成または変更する際は、`DEFAULT_DATABASE` を使用しないでください。	はい – `DEFAULT_DATABASE` を含みます (制限なし)。
データベースのユーザーとアクセス許可	はい – SQL Server ネイティブレプリケーション (ユーザーデータベースに保存) によってレプリケートされます。	はい。
ユーザー定義のサーバーロール	Always On AG: はい – レプリケートされます。DBM: いいえ – レプリケートされません。	はい。

オブジェクト	データベースミラーリング/Always On AG	ブロックレベルのレプリケーション
リンクサーバー	いいえ – AG またはミラーの外部にある master データベースに保存されます。マルチ AZ を有効にする前にリンクサーバーを作成するか、フェイルオーバー後に手動で再作成します。	はい – システムデータベースを含むボリューム全体がレプリケートされます。
SQL Server Audit	部分的 – データベース監査仕様 (ユーザーデータベース内) がレプリケートされます。サーバー監査およびサーバー監査仕様は、レプリケートされません。プライマリと一致するように、AUDIT_GUID パラメータを使用してセカンダリでこれらを手動で作成する必要があります。	はい – すべての監査オブジェクトはボリュームレベルでレプリケートされます。
tempdb の設定	オプション – 次のストアドプロシージャを使用した同期を有効にします。 `EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'TempDbFile';` 一時的なオブジェクトとデータはレプリケートされません。	はい – 設定はボリュームレベルでレプリケートされます。

オブジェクト	データベースミラーリング/Always On AG	ブロックレベルのレプリケーション
SQL Server エージェントジョブ	オプション – 次のストアドプロシージャを使用した同期を有効にします。 `EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'SQLAgent Job';` サポートされているカテゴリの T-SQL ジョブステップのみがレプリケートされます。SSIS、SSRS、レプリケーション、PowerShell、およびデータベースメールのステップタイプはレプリケートされません。	はい – すべてのジョブはボリュームレベルでレプリケートされます。
CDC (変更データキャプチャ)	部分的 – CDC メタデータと変更テーブル (ユーザーデータベース内) がレプリケートされます。キャプチャジョブとクリーンアップジョブ (msdb 内) は、直接レプリケートされません。Amazon RDS は、フェイルオーバー後に、以前に記録されたパラメータを使用して、新しいプライマリでそれらを削除して再作成します。`rds_set_configuration` を使用して、セカンダリに CDC パラメータをプリセットします。	はい – すべての CDC オブジェクトはボリュームレベルでレプリケートされます。

オブジェクト	データベースミラーリング/Always On AG	ブロックレベルのレプリケーション
リソースガバナー	はい – Amazon RDS によってレプリケートされます。同期を確認するには、次のクエリを実行します。 `SELECT * FROM msdb.dbo.rds_fn_server_object_last_sync_time();`	はい。
データベースの所有者	いいえ – セカンダリインスタンスでは、データベース所有者は `NT AUTHORITY\SYSTEM` に設定されます。フェイルオーバー後、`msdb.dbo.rds_changedbowner_to_rdsa` を使用して所有権をリセットします。	はい – 所有者は保持されます。
`msdb` のアクセス許可	いいえ – RDS for SQL Server は、`msdb` データベースのアクセス許可をセカンダリインスタンスにレプリケートしません。手動で再作成する必要があります。	はい – ボリュームレベルでレプリケートされます。

以下は、RDS for Microsoft SQL Server DB インスタンスでマルチ AZ 配置を使用するときのいくつかのレコメンデーションです。

- 本稼働または本稼働前に使用するデータベースでは、以下のオプションを使用することをお勧めします。
  - 高可用性を重視したマルチ AZ 配置
  - 高速で安定したパフォーマンスを実現する「プロビジョンド IOPS」
  - 「汎用」ではなく「メモリ最適化」
- セカンダリ用のインスタンスにはアベイラビリティーゾーン (AZ) を選択することができません。アプリケーションホストをデプロイするときには、この点を考慮してください。データベースが別の AZ にフェイルオーバーする可能性があるため、アプリケーションホストがデータベースと同じ

AZ に含まれない場合があります。このため、特定の AWS リージョン内のすべての AZ 間で、アプリケーションホストのバランスをとることをお勧めします。

- 最高のパフォーマンスのために、大量のデータをロードするオペレーション中はデータベースミラーリング、Always On AG、またはブロックレベルレプリケーションを有効にしないでください。できる限り高速でデータを更新する必要がある場合は、DB インスタンスをマルチ AZ 配置に変換する前にデータの更新を終了します。
- SQL Server データベースにアクセスするアプリケーションには、接続エラーを見つける例外処理が必要です。以下のコード例では、通信エラーを見つける try/catch ブロックを示しています。この例では、接続が成功した場合に `break` ステートメントは `while` ループを終了しますが、例外がスローされた場合は最大 10 回再試行します。

```
int RetryMaxAttempts = 10;
int RetryIntervalPeriodInSeconds = 1;
int iRetryCount = 0;
while (iRetryCount < RetryMaxAttempts)
{
   using (SqlConnection connection = new SqlConnection(DatabaseConnString))
   {
      using (SqlCommand command = connection.CreateCommand())
      {
         command.CommandText = "INSERT INTO SOME_TABLE VALUES ('SomeValue');";
         try
         {
            connection.Open();
            command.ExecuteNonQuery();
            break;
         }
         catch (Exception ex)
         {
            Logger(ex.Message);
            iRetryCount++;
         }
         finally {
            connection.Close();
         }
      }
   }
   Thread.Sleep(RetryIntervalPeriodInSeconds * 1000);
}
```

- DBM または AG を使用してマルチ AZ インスタンスを操作する場合は、Set Partner Off コマンドを使用しないでください。このコマンドは、ブロックレベルレプリケーションを使用するインスタンスではサポートされていません。例えば、以下は実行しないでください。

```
--Don't do this
ALTER DATABASE db1 SET PARTNER off
```

- 復旧モードを simple に設定しないでください。例えば、以下は実行しないでください。

```
--Don't do this
ALTER DATABASE db1 SET RECOVERY simple
```

- 高可用性のためにブロックレベルレプリケーションを使用しない限り、マルチ AZ DB インスタンスに新しいログインを作成するときは、DEFAULT_DATABASE パラメータは使用しないでください。これらの設定は、スタンドバイ用ミラーには適用できないためです。例えば、以下は実行しないでください。

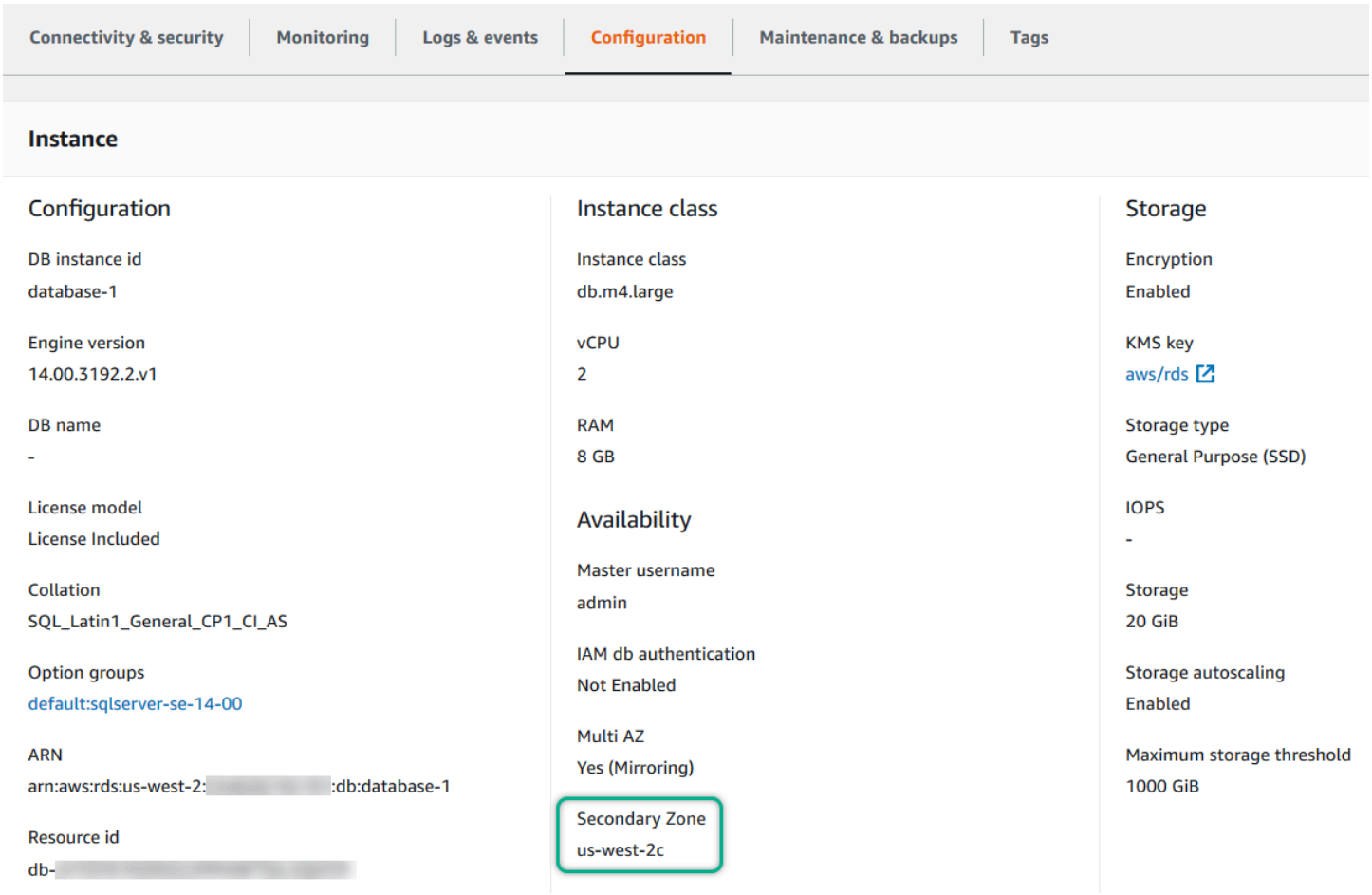
```
--Don't do this
CREATE LOGIN [test_dba] WITH PASSWORD=foo, DEFAULT_DATABASE=[db2]
```

  また、以下の操作をしないでください。

```
--Don't do this
ALTER LOGIN [test_dba] WITH DEFAULT_DATABASE=[db3]
```

セカンダリの場所を確認する

AWS マネジメントコンソール を使用して、セカンダリレプリカの場所を調べることができます。VPC 内のプライマリ DB インスタンスを設定する場合は、セカンダリの場所がわかっている必要があります。



AWS CLI の `describe-db-instances` コマンド、または RDS API の `DescribeDBInstances` オペレーションを使用して、セカンダリのアベイラビリティーゾーンを表示することもできます。その出力には、スタンバイミラーが配置されているセカンダリ AZ が表示されます。

データベースミラーリングから Always On 可用性グループへの移行

Microsoft SQL Server Enterprise Edition のバージョン 14.00.3049.1 では、Always On 可用性グループ (AG) はデフォルトで有効になっています。

データベースミラーリング (DBM) から AG に移行するには、まずバージョンを確認します。Enterprise Edition 13.00.5216.0 より前のバージョンの DB インスタンスを使用している場合は、インスタンスにパッチを適用して 13.00.5216.0 以降に変更します。Enterprise Edition 14.00.3049.1 より前のバージョンの DB インスタンスを使用している場合は、インスタンスにパッチを適用して 14.00.3049.1 以降に変更します。

AG を使用するようにミラーリングされた DB インスタンスをアップグレードする場合は、初期にアップグレードを実行し、インスタンスを変更して マルチ AZ を削除してから、もう一度変更して

マルチ AZ を追加します。これにより、インスタンスは Always On AG を使用するように変換されます。

Amazon RDS での Microsoft SQL Server の追加機能

次のセクションで、Microsoft SQL Server DB エンジンを実行する Amazon RDS インスタンスの拡張について説明します。

トピック

- RDS for SQL Server での SQL Server ログインのパスワードポリシーの使用
- Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合
- Amazon RDS for SQL Server でのデータベースメールの使用
- Amazon RDS for SQL Server の tempdb データベースに対するインスタンスストアのサポート
- Amazon RDS for Microsoft SQL Server で拡張イベントを使用する
- RDS for SQL Server によるトランザクションログのバックアップへのアクセス

RDS for SQL Server での SQL Server ログインのパスワードポリシーの使用

Amazon RDS では、Microsoft SQL Server を実行している Amazon RDS DB インスタンスのパスワードポリシーを設定できます。これを使用して、SQL Server 認証を使用して DB インスタンスに認証するログインの複雑さ、長さ、およびロックアウト要件を設定します。

重要な用語

「ログイン」

SQL Server では、データベースインスタンスに認証できるサーバーレベルのプリンシパルをログインと呼びます。他のデータベースエンジンは、このプリンシパルをユーザーと呼ぶ場合があります。RDS for SQL Server では、ログインは SQL Server 認証または Windows 認証を使用して認証できます。

SQL Server ログイン

SQL Server 認証を使用して認証するためにユーザー名とパスワードを使用するログインは、SQL Server ログインです。DB パラメータを使用して設定するパスワードポリシーは、SQL Server ログインにのみ適用されます。

Windows ログイン

Windows プリンシパルに基づき、Windows 認証を使用して認証するログインは Windows ログインです。Active Directory で Windows ログインのパスワードポリシーを設定できます。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。

各ログインのポリシーの有効化と無効化

各 SQL Server ログインには、CHECK_POLICY および CHECK_EXPIRATION のフラグがあります。デフォルトでは、新しいログインは、CHECK_POLICY を ON に設定し、CHECK_EXPIRATION を OFF に設定して作成されます。

CHECK_POLICY がログインについて有効な場合、RDS for SQL Server は複雑さと最小長の要件に照らしてパスワードを検証します。ロックアウトポリシーも適用されます。CHECK_POLICY と CHECK_EXPIRATION を有効にする T-SQL ステートメントの例:

```
ALTER LOGIN [master_user] WITH CHECK_POLICY = ON, CHECK_EXPIRATION = ON;
```

CHECK_EXPIRATION が有効になっている場合、パスワードはパスワード有効期間ポリシーの対象です。CHECK_POLICY と CHECK_EXPIRATIONが設定されているかどうかを確認する T-SQL ステートメント:

```
SELECT name, is_policy_checked, is_expiration_checked FROM sys.sql_logins;
```

パスワードポリシーパラメータ

すべてのパスワードポリシーパラメータは動的であり、有効にするために DB を再起動する必要はありません。次の表に、SQL Server ログインのパスワードポリシーを変更するために設定できる DB パラメータを示します。

DB パラメータ	説明	許可された値	デフォルト値	

rds.password_complexity_enabled	SQL Server ログインのパスワードを作成または変更するときは、パスワードの複雑さに関する要件を満たす必要があります。次の制約を満たす必要があります。
	• パスワードには、次の 3 つのカテゴリの文字を含める必要があります。
	  • 英小文字 (a ~ z)
	  • 英大文字 (A ~ Z)
		0、1	0	

DB パラメータ	説明	許可された値	デフォルト値	
	• 感嘆符 (!)、ドル記号 ($)、番号記号 (#)、パーセント (%) など、英数字以外の文字。 • パスワードにユーザーのアカウント名が含まれていません。			
rds.password_min_length	SQL Server ログインのパスワードに必要な最小文字数。	0-14	0	
rds.password_min_age	ユーザーが SQL Server ログインパスワードを変更するまでに使用する必要がある最小日数。パスワードは、0 に設定した直後に変更できます。	0 ~ 998	0	

DB パラメータ	説明	許可された値	デフォルト値	
rds.password_max_age	SQL Server ログインパスワードを使用できる最大日数。その後、ユーザーがパスワードを変更する必要があります。0 に設定されると、パスワードは期限切れになりません。	0 ~ 999	42	
rds.password_lockout_threshold	SQL Server ログインがロックアウトされる原因となるログイン試行の連続失敗回数。	0 ~ 999	0	
rds.password_lockout_duration	ロックアウトされた SQL Server ログインがロック解除されるまでに待機する必要がある時間 (分)。	1 ~ 60	10	

DB パラメータ	説明	許可された値	デフォルト値	
rds.password_lockout_reset_counter_after	ログイン試行に失敗してから、失敗したログイン試行カウンターが 0 にリセットされるまでに経過する必要がある時間 (分)。	1 ~ 60	10	

> **Note**
>
> SQL Server パスワードポリシーの詳細については、「パスワードポリシー」を参照してください。
> パスワードの複雑さと最小長ポリシーは、含まれているデータベースの DB ユーザーにも適用されます。詳細については、「含まれているデータベース」を参照してください。

パスワードポリシーパラメータには、次の制約が適用されます。

- `rds.password_max_age` が 0 に設定されていない限り、`rds.password_min_age` パラメータは `rds.password_max_age parameter` より小さい必要があります。
- `rds.password_lockout_reset_counter_after` パラメータは、`rds.password_lockout_duration` パラメータ以下である必要があります。
- `rds.password_lockout_threshold` が 0 に設定されている場合、`rds.password_lockout_duration` と `rds.password_lockout_reset_counter_after` は適用されません。

既存のログインに関する考慮事項

インスタンスのパスワードポリシーを変更した後、ログイン用の既存のパスワードは、新しいパスワードの複雑さと長さの要件に対して遡及的に評価されません。新しいパスワードのみが新しいポリシーに対して検証されます。

SQL Server は、既存のパスワードの有効期間要件を評価します。

パスワードポリシーが変更されると、パスワードがすぐに期限切れになる可能性があります。例えば、ログインの CHECK_EXPIRATION が有効であり、パスワードが 100 日前に変更され、rds.password_max_age パラメータを 5 日に設定した場合、パスワードは直ちに期限切れになり、次回のログイン時にパスワードを変更する必要があります。

> **Note**
>
> RDS for SQL Server は、パスワード履歴ポリシーをサポートしていません。履歴ポリシーは、以前に使用されたパスワードを再利用したログインを禁止します。

マルチ AZ 配置に関する考慮事項

マルチ AZ インスタンスの失敗したログイン試行カウンターとロックアウト状態は、ノード間でレプリケートされません。マルチ AZ インスタンスがフェイルオーバーしたときにログインがロックアウトされていた場合、新しいノードでログインがすでにロック解除されている可能性があります。

マスターログインのパスワードに関する考慮事項

RDS for SQL Server DB インスタンスを作成する場合、マスターユーザーパスワードはパスワードポリシーに対して評価されません。新しいマスターパスワードは、マスターユーザーに操作を実行するとき、特に ModifyDBInstance コマンドで MasterUserPassword を設定するときにも、パスワードに対して評価されません。いずれの場合も、パスワードポリシーを満たさないマスターユーザーのパスワードを設定でき、その場合でも操作は成功します。ポリシーが満たされない場合、RDS は強力なパスワードの設定を推奨して、RDS イベントのレイズを試みます。マスターユーザーには強力なパスワードのみを使用してください。

マスターユーザーのパスワードがパスワードポリシーの要件を満たしていない場合、RDS は次のイベントメッセージの生成を試みます。

- マスターユーザーは作成されましたが、パスワードがパスワードポリシーの最小長要件を満たしていません。より強力なパスワードの使用を検討してください。
- マスターユーザーは作成されましたが、パスワードがパスワードポリシーの複雑さ要件を満たしていません。より強力なパスワードの使用を検討してください。
- マスターユーザーのパスワードはリセットされましたが、パスワードがパスワードポリシーの最小長要件を満たしていません。より強力なパスワードの使用を検討してください。

- マスターユーザーのパスワードはリセットされましたが、パスワードがパスワードポリシーの複雑さ要件を満たしていません。より強力なパスワードの使用を検討してください。

デフォルトでは、マスターユーザーは CHECK_POLICY と CHECK_EXPIRATION が OFF に設定されて作成されます。パスワードポリシーをマスターユーザーに適用するには、DB インスタンスの作成後、マスターユーザーに対してこれらのフラグを手動で有効にする必要があります。これらのフラグを有効にしたら、SQL Server でマスターユーザーのパスワードを直接 (T-SQL ステートメントや SSMS などを介して) 変更し、新しいパスワードをパスワードポリシーに照らして検証します。

> **Note**
>
> マスターユーザーがロックアウトされた場合は、ModifyDBInstance コマンドを使用してマスターユーザーのパスワードをリセットすることで、ユーザーのロックを解除できます。

マスターユーザーパスワードの変更

マスターユーザーのパスワードは、ModifyDBInstance コマンドを使用して変更できます。

> **Note**
>
> マスターユーザーのパスワードをリセットすると、RDS はマスターユーザーのさまざまなアクセス許可をリセットし、マスターユーザーは特定のアクセス許可を失う可能性があります。マスターユーザーのパスワードをリセットすると、マスターユーザーがロックアウトされた場合もロックが解除されます。

RDS はマスターユーザーの新しいパスワードを検証し、パスワードがポリシーを満たさない場合は RDS イベントの発行を試みます。RDS は、パスワードポリシーを満たさない場合でもパスワードを設定します。

Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合

Amazon RDS for SQL Server を実行する DB インスタンスと Amazon S3 バケットの間でファイルを転送できます。これにより、BULK INSERT などの SQL Server 特性で Amazon S3 を使用することができます。例えば、Amazon S3 から .csv、.xml、.txt、その他ファイルを DB インスタンスホストにダウンロードして、D:\S3\ からデータベースにデータをインポートできます。全てのファイルは DB インスタンスの D:\S3\ に保存されます。

以下の制限が適用されます。

> **Note**
>
> RDS ホストと S3 間のトラフィックは、S3 を使用するすべての SQL Server 機能の RDS 内部 VPC エンドポイントを経由します。このトラフィックは RDS インスタンスエンドポイント ENI を使用しません。S3 バケットポリシーは、ネットワーク条件によって RDS トラフィックを制限することはできません。

- D:\S3 フォルダ内のファイルは、マルチ AZ インスタンスでのフェイルオーバー後にスタンバイレプリカで削除されます。詳細については、「S3 統合のマルチ AZ の制限」を参照してください。
- DB インスタンスと S3 バケットが同じ AWS リージョンに存在する必要があります。
- 一度に複数の S3 統合タスクを実行する場合、タスクは並列ではなく順次に実行されます。

  > **Note**
  >
  > S3 統合タスクは、ネイティブバックアップおよび復元タスクと同じキューを共有します。このキューでは、一度に最大2つのタスクまで同時進行させることができます。したがって、実行中の 2 つのネイティブバックアップおよび復元タスクによって S3 統合タスクがブロックされます。

- S3 統合機能を復元インスタンスで再度有効にする必要があります。S3 統合は元のインスタンスから復元インスタンスに伝搬されることはありません。D:\S3 のファイルは、復元インスタンスで削除されます。
- DB インスタンスへのダウンロードは、最大 100 ファイルまでです。言い換えれば、D:\S3\ 内には 100 ファイル以上は入れておけません。

- ファイル拡張子がない、または次のファイル拡張子があるファイルのみがダウンロードできます。.abf、.asdatabase、.bcp、.configsettings、.csv、.dat、.deploymentoptions、.deploymenttargets、.fmおよび.xmla
- S3 バケットは、関連の AWS Identity and Access Management (IAM) 役割に関連した同じオーナーである必要があります。したがって、クロスアカウント S3 統合はサポートされていません。
- S3 バケットは一般に公開できません。
- RDS から S3 へのアップロードのファイルサイズは、1 ファイルあたり 50 GB に制限されます。
- S3 から RDS へのダウンロードのファイルサイズは、S3 でサポートされている最大サイズに制限されます。

トピック

- RDS for SQL Server を S3 と統合するための前提条件
- RDS for SQL Server と S3 の統合を有効にする
- RDS for SQL Server と Amazon S3 間のファイル転送
- RDS DB インスタンスでのファイル一覧表示
- RDS DB インスタンスでのファイル削除
- ファイル転送タスクのステータスをモニタリングする
- タスクのキャンセル
- S3 統合のマルチ AZ の制限
- RDS for SQL Server と S3 の統合を無効にする

Amazon S3 でのファイルの操作の詳細については、「Amazon Simple Storage Service の開始方法」を参照してください。

RDS for SQL Server を S3 と統合するための前提条件

開始する前に、使用するS3バケットを選択してください。また、許可を追加してRDS DBインスタンスがS3バケットにアクセスできるようにしてください。このアクセスを設定するには、IAMポリシーとIAMロールの両方を作成します。

コンソール

Amazon S3にアクセスするための IAM ポリシーを作成します。

1. IAMマネジメントコンソールのナビゲーションペインで、ポリシー を選択します。

2. 新しいポリシーを作成し、ビジュアルエディタタブを使用して以下の手順を行ってください。
3. サービスのため、**S3**を入力してS3サービスを選択します。
4. 実行のため，以下を選択してDBインスタンス要件にアクセスできるようにします。

   - `ListAllMyBuckets` – 必須
   - `ListBucket` – 必須
   - `GetBucketAcl` – 必須
   - `GetBucketLocation` – 必須
   - `GetObject` – S3 から にファイルをダウンロードする際に必須`D:\S3\`
   - `PutObject` – `D:\S3\` から S3 にファイルをアップロードする際に必須
   - `ListMultipartUploadParts` – `D:\S3\` から S3 にファイルをアップロードする際に必須
   - `AbortMultipartUpload` – `D:\S3\` から S3 にファイルをアップロードする際に必須
5. [リソース] では、表示されるオプションは、前の手順で選択した内容により異なります。[バケット]、[オブジェクト]、またはその両方に対するオプションが表示されます。それぞれ、適切な Amazon リソースネーム (ARN) を加えてください。

   [バケット] は、使用したいバケットに対する ARN を追加します。例えば、バケットの名前が *`amzn-s3-demo-bucket`* の場合、ARN を `arn:aws:s3:::`*`amzn-s3-demo-bucket`* に設定します。

   [オブジェクト] は、バケットの ARN を入力してから以下のいずれかを選択します。

   - 特定のバケット内のすべてのファイルへのアクセスを許可するには、[バケット名] および [オブジェクト名] の両方に対して [すべて] を選択してください。
   - バケット内の特定のファイルやフォルダへのアクセスを許可する場合は、SQL Server からのアクセスを希望する特定のバケットやオブジェクトに、ARN を提供します。
6. ポリシーの作成が完了するまで、コンソール内の指示に従ってください。

   前述の部分は、ポリシーの簡略設定ガイドです。IAM ポリシーを作成する詳細な手順については、IAM ユーザーガイドの IAM ポリシーの作成を参照してください。

前の手順からの IAM ポリシーを使用する IAM ロールを作成します。

1. [IAM 管理コンソール] で、ナビゲーションペインから [ロール] を選択します。

2. 新しい IAM ロールを作成し、コンソール内に表示された以下のオプションを選択してください。

   - AWS のサービス
   - RDS
   - RDS – ロールをデータベースに加える

   次に、[Next:Permissions] を選択します。
3. アクセス権限ポリシーをアタッチする で、事前に作成した IAM ポリシーの名前を入力してください。次にリストからポリシーを選択します。
4. ロールの作成が完了するまで、コンソール内の指示に従ってください。

   前述の部分は、ロールの簡略設定ガイドです。ロールを作成する詳細な手順については、IAM ユーザーガイドの IAM ロールを参照してください。

AWS CLI

Amazon RDS に Amazon S3 バケットへのアクセスを付与するには、次のプロセスを使用します。

1. S3 バケットに Amazon RDS アクセスを付与する IAM ポリシーを作成します。
2. S3 バケットにアクセスするには、お客様に代わって Amazon RDS が引き受けることのできる IAM ロールを作成します。

   詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。
3. 作成した IAM ポリシーを、作成した IAM ロールにアタッチします。

IAM ポリシーを作成するには

DB インスタンスが必要とするアクセスを付与するための適切なアクションが含まれるようにしてください。

- `ListAllMyBuckets` – 必須
- `ListBucket` – 必須
- `GetBucketAcl` – 必須
- `GetBucketLocation` – 必須

- GetObject – S3 から にファイルをダウンロードする際に必須D:\S3\
- PutObject – D:\S3\ から S3 にファイルをアップロードする際に必須
- ListMultipartUploadParts – D:\S3\ から S3 にファイルをアップロードする際に必須
- AbortMultipartUpload – D:\S3\ から S3 にファイルをアップロードする際に必須

1. 以下の AWS CLI コマンドでは、これらのオプションを指定して、rds-s3-integration-policy という名前の IAM ポリシーを作成します。*amzn-s3-demo-bucket* という名前のバケットへのアクセスを許可します。

   Example

   Linux、macOS、Unix の場合:

```
aws iam create-policy \
  --policy-name rds-s3-integration-policy \
  --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "s3:ListAllMyBuckets",
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketAcl",
                    "s3:GetBucketLocation"
                ],
                "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetObject",
                    "s3:PutObject",
                    "s3:ListMultipartUploadParts",
                    "s3:AbortMultipartUpload"
                ],
                "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
```

```
            }
        ]
    }'
```

Windows の場合:

インターフェイスでサポートされた改行コードに、必ず変更してください (^ではなく\)。また Windows では、すべてのダブルクオテーションを \ にエスケープしてください。JSON のクオーツをエスケープしなくても済むようにするには、代わりにファイルに保存し、パラメータとしてパスします。

最初に、以下の許可ポリシーで policy.json ファイルを作成してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketACL",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
        }
```

```
    ]
}
```

次のコマンドを使用してポリシーを作成します。

```
aws iam create-policy ^
     --policy-name rds-s3-integration-policy ^
     --policy-document file://file_path/assume_role_policy.json
```

2. ポリシーが作成されたら、そのポリシーの Amazon リソースネーム (ARN) を書き留めます。この ARN は、後のステップで必要になります。

IAM ロールを作成するには

- 次の AWS CLI コマンドでは、この目的で `rds-s3-integration-role` IAM ロールを作成します。

  Example

  Linux、macOS、Unix の場合:

```
aws iam create-role \
    --role-name rds-s3-integration-role \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
             "Service": "rds.amazonaws.com"
           },
          "Action": "sts:AssumeRole"
        }
      ]
    }'
```

  Windows の場合:

  インターフェイスでサポートされた改行コードに、必ず変更してください (^ではなく\)。また Windows では、すべてのダブルクオーテーションを \ にエスケープしてください。JSON のク

オーツをエスケープしなくても済むようにするには、代わりにファイルに保存し、パラメータとしてパスします。

最初に、次のポリシーで、assume_role_policy.json ファイルを作成します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

次に、以下のコマンドを使用して IAM ロールを作成します。

```
aws iam create-role ^
     --role-name rds-s3-integration-role ^
     --assume-role-policy-document file://file_path/assume_role_policy.json
```

Exampleグローバル条件コンテキストキーを使用した IAM ロールの作成

リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与するリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

両方のグローバル条件コンテキストキーを使用し、aws:SourceArn 値にアカウント ID を含めます。この場合、同じポリシーステートメントで使用する際に、aws:SourceAccount 値と aws:SourceArn 値のアカウントで同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

ポリシーでは、ロールにアクセスするリソースの完全な ARN を持つ aws:SourceArn グローバル条件コンテキストキーを必ず使用してください。S3 統合の場合、次の例に示すように DB インスタンスの ARN を必ず含めてください。

Linux、macOS、Unix の場合:

```
aws iam create-role \
    --role-name rds-s3-integration-role \
    --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
             "Service": "rds.amazonaws.com"
           },
          "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {

 "aws:SourceArn":"arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                    }
                }
        }
      ]
    }'
```

Windows の場合:

assume_role_policy.json にグローバル条件コンテキストキーを追加します。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {

 "aws:SourceArn":"arn:aws:rds:Region:my_account_ID:db:db_instance_identifier"
                }
            }
        }
    ]
}
```

IAM ポリシーを IAM ロールにアタッチするには

- 以下の AWS CLI コマンドでは、rds-s3-integration-role という名前のロールにこのポリシーをアタッチします。*your-policy-arn* を、以前のステップで書き留めたポリシー ARN に置き換えます。

  Example

  Linux、macOS、Unix の場合:

  ```
  aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-integration-role
  ```

  Windows の場合:

  ```
  aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-integration-role
  ```

RDS for SQL Server と S3 の統合を有効にする

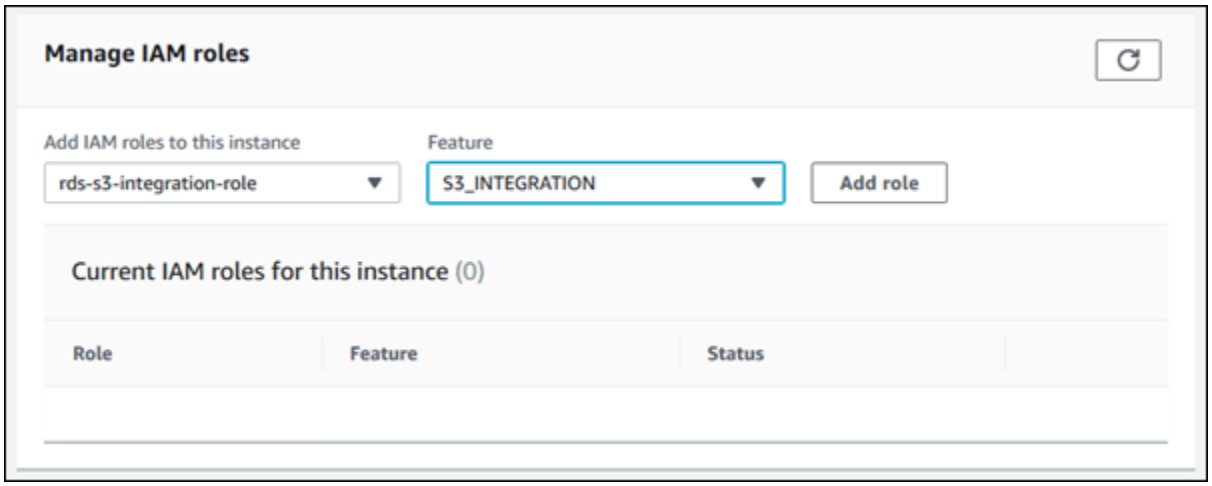
このセクションでは、Amazon RDS for SQL Server と Amazon S3 の統合を有効にする方法を確認できます。S3 統合を行うには、`S3_INTEGRATION` FeatureName パラメータを使用する前に、事前に作成した IAM ロールに DB インスタンスが関連付けられていなければなりません。

> **Note**
>
> IAM ロールを DB インスタンスに追加するには、DB インスタンスのステータスが [有効] である必要があります。

コンソール

IAM ロールを DB インスタンスに関連付けるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. RDS for SQL Server DB インスタンスの名前を選択して詳細を表示します。
3. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、[このインスタンスに IAM ロールを追加] で追加する IAM ロールを選択します。
4. [機能] で、[S3_INTEGRATION] を選択します。



5. [Add role] を選択します。

AWS CLI

IAM ロールを RDS for SQL Server DB インスタンスに追加するには

- 以下の AWS CLI コマンドは、IAM ロールを *mydbinstance* と名前の付いた RDS for SQL Server DB インスタンスに追加します。

  Example

  Linux、macOS、Unix の場合:

  ```
  aws rds add-role-to-db-instance \
      --db-instance-identifier mydbinstance \
      --feature-name S3_INTEGRATION \
      --role-arn your-role-arn
  ```

  Windows の場合:

  ```
  aws rds add-role-to-db-instance ^
      --db-instance-identifier mydbinstance ^
      --feature-name S3_INTEGRATION ^
      --role-arn your-role-arn
  ```

  your-role-arn を、以前のステップで書き留めたロール ARN に置き換えます。`S3_INTEGRATION` オプションには `--feature-name` が指定されている必要があります。

RDS for SQL Server と Amazon S3 間のファイル転送

Amazon RDS ストアドプロシージャを使用して、Amazon S3 と RDS DB インスタンス間でファイルのダウンロードおよびアップロードを行います。また、Amazon RDS ストアドプロシージャを使用して、RDS インスタンスのファイルを記入および削除することができます。

S3 からダウンロードまたは S3 へアップロードするファイルは、`D:\S3` フォルダに保存します。このフォルダは、ファイルにアクセスする際に使用できる唯一のフォルダとなります。ダウンロード時に対象フォルダを設定する際に作成したサブフォルダ内でファイルを構成することができます。

ストアドプロシージャによっては、Amazon Resource Name (ARN) を S3 バケットおよびファイルに指定する必要があります。ARN の形式は `arn:aws:s3:::`*amzn-s3-demo-bucket*`/file_name` です。Amazon S3 には、ARN のアカウント番号または AWS リージョンは不要です。

S3 統合タスクは順次実行され、同じキューをネイティブバックアップとして共有し、タスクの復元を行います。このキューでは、一度に最大2つのタスクまで同時進行させることができます。タスクが処理を開始するまでに、最大 5 分かかります。

Amazon S3 バケットから SQL Server DB インスタンスにファイルをダウンロードする

S3 バケットから RDS for SQL Server DB インスタンスにファイルをダウンロードするには、Amazon RDS ストアドプロシージャ `msdb.dbo.rds_download_from_s3` を使用してください。

パラメータ名	データ型	デフォルト	必須	説明
`@s3_arn_of_file`	NVARCHAR	–	必須	ダウンロードするファイルの S3 ARN 例: `arn:aws:s3::: amzn-s3-demo-bucket / mydata.csv`
`@rds_file_path`	NVARCHAR	–	オプション	RDS インスタンスのファイルパス。指定されなかった場合、ファイルパスは `D:\S3\<filename in s3>` です。RDS は、絶対パスと相対パスをサポートしています。サブフォルダを作成したい場合、ファイルパス内に含めます。
`@overwrite_file`	INT	0	オプション	既存のファイルを上書きしてください。 0 = 上書きしないでください 1 = 上書きしてください

ファイル拡張子のないファイルと、ファイル拡張子
が .bcp、.csv、.dat、.fmt、.info、.lst、.tbl、.txt、.xml のファイルをダウンロードできます。

> **Note**
>
> SQL Server Integration Services が有効になっている場合、ファイル拡張子が .ispac のファイルのダウンロードがサポートされます。SSIS の有効化の詳細については、「SQL Server Integration Services」を参照してください。
> SQL Server Analysis Services が有効になっている場合、ファイル拡張子
> が .abf、.asdatabase、.configsettings、.deploymentoptions、.deploymenttargets、.xmla のファイルのダウンロードがサポートされます。SSAS の有効化の詳細については、「SQL Server Analysis Services」を参照してください。

以下の例は S3 からファイルをダウンロードするためのストアドプロシージャを表します。

```
exec msdb.dbo.rds_download_from_s3
     @s3_arn_of_file='arn:aws:s3:::amzn-s3-demo-bucket/bulk_data.csv',
     @rds_file_path='D:\S3\seed_data\data.csv',
     @overwrite_file=1;
```

例 `rds_download_from_s3` の操作は、フォルダがまだない場合、`seed_data` に `D:\S3\` という名前のフォルダを作成します。次に例ではソースファイル `bulk_data.csv` を S3 から DB インスタンスの `data.csv` という名前の新しいファイルにダウンロードします。`@overwrite_file` パラメータが 1 に設定されているため、すでにファイルが存在する場合は上書きされます。

SQL Server DB インスタンスから Amazon S3 バケットにファイルをアップロードする

RDS for SQL Server DB インスタンスから S3 バケットにファイルをアップロードするには、Amazon RDS ストアドプロシージャ `msdb.dbo.rds_upload_to_s3` を以下のパラメータで使用してください。

パラメータ名	データ型	デフォルト	必須	説明
`@s3_arn_of_file`	NVARCHAR	–	必須	ファイルの S3 ARN が S3 内で作成されます (例: `arn:aws:s3::: amzn-s3-demo-bucket / mydata.csv`)。

パラメータ名	データ型	デフォルト	必須	説明
@rds_file_path	NVARCHAR	–	必須	S3 にアップロードするファイルのファイルパス。絶対パスと相対パスの両方をサポートしています。
@overwrite_file	INT	–	オプション	既存のファイルを上書きしてください。 0 = 上書きしないでください 1 = 上書きしてください

以下の例では、data.csv という名前のファイルを D:\S3\seed_data\ 内の指定の場所から、ARN が指定する S3 バケットに、ファイル new_data.csv をアップロードします。

```
exec msdb.dbo.rds_upload_to_s3
  @rds_file_path='D:\S3\seed_data\data.csv',
  @s3_arn_of_file='arn:aws:s3:::amzn-s3-demo-bucket/new_data.csv',
  @overwrite_file=1;
```

@overwrite_file パラメータが 1 に設定されているため、ファイルが S3 にすでに存在している場合は上書きされます。

RDS DB インスタンスでのファイル一覧表示

DB インスタンスで使用可能なファイルをリスト表示するには、ストアドプロシージャと機能の両方を使用します。最初に、以下のストアドプロシージャを実行して D:\S3\ 内のファイルから詳細を集めます。

```
exec msdb.dbo.rds_gather_file_details;
```

ストアドプロシージャは、タスクの ID を返します。保管のタスクと同様、ストアドプロシージャも同期せずに実行されます。タスクのステータスが SUCCESS になるとすぐ

に、rds_fn_list_file_details 機能のタスク ID を使用して、 D:\S3\ 内の既存のファイルやディレクトリのリスト表示ができます。以下を参照してください。

```
SELECT * FROM msdb.dbo.rds_fn_list_file_details(TASK_ID);
```

rds_fn_list_file_details 機能は、次の列があるテーブルを返します。

出力パラメータ	説明
filepath	ファイルの絶対パス (例: D:\S3\mydata.csv)
size_in_bytes	ファイルサイズ (バイト単位)
last_modified_utc	UTC 形式での最終変更を行った日時
is_directory	アイテムがディレクトリ (true/false) かどうかを表示するオプション

RDS DB インスタンスでのファイル削除

DB インスタンスで使用可能なファイルを削除するには、Amazon RDS ストアドプロシージャ msdb.dbo.rds_delete_from_filesystem を以下のパラメータで使用します。

パラメータ名	データ型	デフォルト	必須	説明
@rds_file_path	NVARCHAR	–	必須	削除するファイルのファイルパス。絶対パスと相対パスの両方をサポートしています。
@force_delete	INT	0	オプション	ディレクトリを削除するには、このフラグが 1 に含まれ設定されている必要があります。 1 = ディレクトリを削除する

パラメータ名	データ型	デフォルト	必須	説明
				ファイルを削除する場合、このパラメータは無視されます。

ディレクトリを削除するには、@rds_file_path がバックスラッシュ (\) で終了し、@force_delete が 1 に設定される必要があります。

次の例では、ファイル D:\S3\delete_me.txt が削除されます。

```
exec msdb.dbo.rds_delete_from_filesystem
    @rds_file_path='D:\S3\delete_me.txt';
```

次の例では、ディレクトリ D:\S3\example_folder\ が削除されます。

```
exec msdb.dbo.rds_delete_from_filesystem
    @rds_file_path='D:\S3\example_folder\',
    @force_delete=1;
```

ファイル転送タスクのステータスをモニタリングする

S3 統合タスクのステータスを追跡するには、rds_fn_task_status 機能を呼び出してください。2 つのパラメータを使用します。1 つめのパラメータは常に NULL を選択してください。これは、S3 統合に適用されないためです。2 つめのパラメータは、タスク ID を受け入れます。

全タスクのリストを見るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータを 0 に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

特定のタスクを受け取るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータをタスク ID に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 機能は次の情報を返します。

出力パラメータ	説明
task_id	タスクの ID。
task_type	S3 統合では、タスクには以下のタスクタイプがあります。 • DOWNLOAD_FROM_S3 • UPLOAD_TO_S3 • LIST_FILES_ON_DISK • DELETE_FILES_ON_DISK
database_name	S3 統合タスクには適用できません。
% complete	タスクの進行状況の割合。
duration(mins)	タスクにかかった時間 (分単位)。
lifecycle	タスクのステータス。有効な状態には以下のものがあります。 • CREATED – S3 統合ストアドプロシージャを呼び出すと、タスクが作成され、ステータスが に設定されます。CREATED • IN_PROGRESS – タスクが開始すると、ステータスが に設定されます。IN_PROGRESS ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。 • SUCCESS - タスクが完了すると、ステータスが に設定されます。SUCCESS • ERROR - タスクが失敗すると、ステータスがに設定されます。ERROR工ラーの詳細については、task_info 列を参照してください。

出力パラメータ	説明
	• CANCEL_REQUESTED - rds_cancel_task を呼び出すと、タスクのステータスが CANCEL_REQUESTED になります。 • CANCELLED - タスクが正常にキャンセルされると、タスクのステータスが に設定されます。CANCELLED
task_info	タスクに関する追加情報。処理中にエラーが発生した場合、この列にエラーに関する情報が含まれます。
last_updated	タスクのステータスが最後に更新された日時。
created_at	タスクが作成された日時。
S3_object_arn	S3 オブジェクトの ARN はダウンロードまたはアップロードされます。
overwrite_S3_backup_file	S3 統合タスクには適用できません。
KMS_master_key_arn	S3 統合タスクには適用できません。
filepath	RDS DB インスタンスのファイルパス。
overwrite_file	既存のファイルが上書きされるかどうかを表示するオプション
task_metadata	S3 統合タスクには適用できません。

タスクのキャンセル

S3 統合タスクをキャンセルするには、msdb.dbo.rds_cancel_task パラメータで task_id ストアドプロシージャを使用します。進行中の削除およびリスト作成タスクは、キャンセルできません。以下の例は、タスクをキャンセルするリクエストを示します。

```
exec msdb.dbo.rds_cancel_task @task_id = 1234;
```

すべてのタスクとそのタスク ID の概要を表示するには、 rds_fn_task_status に記載のように ファイル転送タスクのステータスをモニタリングする 機能を使用してください。

S3 統合のマルチ AZ の制限

マルチ AZ インスタンスでは、D:\S3 フォルダ内のファイルはフェイルオーバー後にスタンバイレプリカから削除されます。フェイルオーバーは、例えば、インスタンスクラスの変更やエンジンバージョンのアップグレードなど、DB インスタンスの変更中に計画できます。または、プライマリの停止中にフェイルオーバーが予定外になることがあります。

> **Note**
>
> D:\S3 フォルダをファイルストレージに使用することはお勧めしません。ベストプラクティスは、作成したファイルを Amazon S3 にアップロードして耐久性を保持し、データをインポートする必要があるときにファイルをダウンロードすることです。

最後のフェイルオーバー時間を決定するには、msdb.dbo.rds_failover_time ストアドプロシージャを使用できます。詳細については、「Amazon RDS for SQL Server の最終フェイルオーバー時間の決定」を参照してください。

Example最近のフェイルオーバーがない例

この例は、エラーログに最近のフェイルオーバーが存在しない場合の出力を示しています。2020-04-29 23:59:00.01 以降、フェイルオーバーは発生していません。

したがって、rds_delete_from_filesystem ストアドプロシージャを使用して削除されていない、その時間の後にダウンロードされたすべてのファイルは、現在のホストで引き続きアクセスできます。それ以前にダウンロードされたファイルも利用可能になる場合があります。

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	null

Example最近のフェイルオーバーの

この例は、エラーログにフェイルオーバーが存在する場合の出力を示しています。最新のフェイルオーバーは、2020-05-05 18:57:51.89 に発生しています。

`rds_delete_from_filesystem` ストアドプロシージャを使用して削除されていない、その時間以降にダウンロードされたすべてのファイルは、現在のホストで引き続きアクセスできます。

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	2020-05-05 18:57:51.8900000

RDS for SQL Server と S3 の統合を無効にする

ここでは、Amazon RDS for SQL Server と Amazon S3 の統合を無効にする方法について説明します。S3 統合を無効化するときは、`D:\S3\` のファイルは削除されません。

> **Note**
>
> IAM ロールを DB インスタンスから削除するには、DB インスタンスのステータスが `available` である必要があります。

コンソール

IAM ロールと DB インスタンスの関連を外す

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. RDS for SQL Server DB インスタンスの名前を選択して詳細を表示します。
3. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、削除する IAM ロールを選択します。
4. [削除] を選択します。

AWS CLI

IAM ロールを RDS for SQL Server DB インスタンスから削除するには

- 以下の AWS CLI コマンドが、IAM ロールを *`mydbinstance`* と名前の付いたRDS for SQL Server DB インスタンスから削除します。

Example

Linux、macOS、Unix の場合:

```
aws rds remove-role-from-db-instance \
    --db-instance-identifier mydbinstance \
    --feature-name S3_INTEGRATION \
    --role-arn your-role-arn
```

Windows の場合:

```
aws rds remove-role-from-db-instance ^
    --db-instance-identifier mydbinstance ^
    --feature-name S3_INTEGRATION ^
    --role-arn your-role-arn
```

your-role-arn を、--feature-name オプションにとって適した IAM ロール ARN に交換します。

Amazon RDS for SQL Server でのデータベースメールの使用

データベースメールを使用して、SQL Server データベースインスタンスの Amazon RDS からユーザーに E メールメッセージを送信できます。メッセージには、ファイルとクエリ結果を含めることができます。データベースメールは、次のコンポーネントを含みます。

- 設定オブジェクトおよびセキュリティオブジェクト – これらのオブジェクトは 、プロファイルとアカウントを作成し、msdb データベースに保存されます。
- メッセージングオブジェクト – これらのオブジェクトは、メッセージの送信に使用する sp_send_dbmail ストアドプロシージャと、メッセージに関する情報を保持するデータ構造を含みます。それらは msdb データベースに保存されます。
- ログオブジェクトと監査オブジェクト – データベースメールは、msdb データベースと Microsoft Windows アプリケーションイベントログにログ情報を書き込みます。
- データベースメール 実行可能ファイル – DatabaseMail.exe は、msdb データベースのキューから読み取り、E メールメッセージを送信します。

RDS は、Web Edition、Standard Edition、および Enterprise Edition の SQL Server のすべてのバージョンで、データベースメールをサポートします。

制約事項

SQL Server DB インスタンスでのデータベースメールの使用には、次の制約事項が適用されます。

- データベースメールは、SQL Server Express Edition ではサポートされていません。
- データベースメールの設定パラメータの変更はサポートされていません。プリセット (デフォルト) の値を表示するには、sysmail_help_configure_sp ストアドプロシージャを使用します。
- 添付ファイルは、完全にはサポートされていません。詳細については、「添付ファイルの使用」を参照してください。
- 添付ファイルの最大サイズは 1 MB です。
- データベースメールは、マルチ AZ DB インスタンスで追加の設定が必要です。詳細については、「マルチ AZ 配置に関する考慮事項」を参照してください。
- 定義済み演算子に E メールメッセージを送信する SQL Server エージェントの設定はサポートされていません。

データベースメールの有効化

DB インスタンスでデータベースメールを有効にするには、次の手順に従います。

1. 新しいパラメータグループを作成します。
2. パラメータグループを変更して、`database mail xps` パラメータを 1 に設定します。
3. パラメータグループを DB インスタンスに関連付けます。

データベースメールパラメータグループの作成

DB インスタンスの SQL Server のエディションとバージョンに対応する `database mail xps` パラメータのパラメータグループを作成します。

> **Note**
>
> 既存のパラメータグループを変更することもできます。「データベースメールを有効にするパラメータの変更」 の手順に従います。

コンソール

次の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [Create parameter group] を選択します。
4. [パラメータグループの作成] ペインで、次の操作を行います。
   a. [パラメータグループファミリー] で、[sqlserver-se-13.0] を選択します。
   b. [グループ名] に、パラメータグループの識別子 (**`dbmail-sqlserver-se-13`** など) を入力します。
   c. [説明] に「**`Database Mail XPs`**」と入力します。
5. [作成] を選択します。

CLI

次の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "Database Mail XPs"
```

  Windows の場合:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "Database Mail XPs"
```

データベースメールを有効にするパラメータの変更

DB インスタンスの SQL Server のエディションとバージョンに対応するパラメータグループの `database mail xps` パラメータを変更します。

データベースメールを有効にするには、`database mail xps` パラメータを 1 に設定します。

コンソール

次の例では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [dbmail-sqlserver-se-13] などのパラメータグループを選択します。

4. [パラメータ] で、パラメータのリストを **mail** でフィルタ処理します。
5. [データベースメール xps] を選択します。
6. [Edit parameters] を選択します。
7. **1** と入力します。
8. [Save changes] (変更の保存) をクリックします。

CLI

次の例では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --parameters "ParameterName='database mail
 xps',ParameterValue=1,ApplyMethod=immediate"
```

  Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --parameters "ParameterName='database mail
 xps',ParameterValue=1,ApplyMethod=immediate"
```

パラメータグループと DB インスタンスの関連付け

AWS マネジメントコンソール または AWS CLI を使用して、データベースメールパラメータグループを DB インスタンスに関連付けることができます。

コンソール

データベースメールパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれを関連付けます。詳しくは、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

データベースメールパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

データベースメールパラメータグループを使用して DB インスタンスを作成するには

- パラメータグループの作成時に使用したものと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li
    --db-parameter-group-name dbmail-sqlserver-se-13
```

  Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
```

```
    --storage-type gp2 ^
    --license-model li ^
    --db-parameter-group-name dbmail-sqlserver-se-13
```

DB インスタンスを変更し、データベースメールパラメータグループを関連付けるには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name dbmail-sqlserver-se-13 \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name dbmail-sqlserver-se-13 ^
    --apply-immediately
```

データベースメールの設定

データベースメールを設定するには、次のタスクを実行します。

1. データベースメールプロファイルを作成します。
2. データベースメールアカウントを作成します。
3. データベースメールアカウントをデータベースメールプロファイルに追加します。
4. データベースメールプロファイルにユーザーを追加します。

> **Note**
>
> データベースメールを設定するには、execute データベースのストアドプロシージャに msdb アクセス権限があることを確認します。

データベースメールプロファイルの作成

データベースメールプロファイルを作成するには、sysmail_add_profile_sp ストアドプロシージャを使用します。次の例では、Notifications という名前のプロファイルを作成します。

プロファイルを作成するには

- 次の SQL 文を使用します。

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_profile_sp
    @profile_name         = 'Notifications',
    @description          = 'Profile used for sending outgoing notifications using
 Amazon SES.';
GO
```

データベースメールアカウントの作成

データベースメールアカウントを作成するには、sysmail_add_account_sp ストアドプロシージャを使用します。次の例では、Amazon Simple Email Service を使用して、プライベート VPC の RDS for SQL Server DB インスタンスに SES という名前のアカウントを作成します。

Amazon SES を使用するには、以下のパラメータが必要です。

- @email_address – Amazon SES 検証済みの アイデンティティ。詳細については、Verified identities in Amazon SES を参照してください。
- @mailserver_name – Amazon SES SMTP エンドポイント。詳細については、Amazon SES SMTP エンドポイントへの接続を参照してください。
- @username – Amazon SES SMTP ユーザー名。詳細については、Amazon SES SMTP 認証情報の取得を参照してください。

  AWS Identity and Access Management ユーザー名を使用しないでください。
- @password – Amazon SES SMTP パスワード。詳細については、Amazon SES SMTP 認証情報の取得を参照してください。

アカウントを作成するには

- 次の SQL 文を使用します。

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_account_sp
    @account_name        = 'SES',
    @description         = 'Mail account for sending outgoing notifications.',
    @email_address       = 'nobody@example.com',
    @display_name        = 'Automated Mailer',
    @mailserver_name     = 'vpce-0a1b2c3d4e5f-01234567.email-smtp.us-
west-2.vpce.amazonaws.com',
    @port                = 587,
    @enable_ssl          = 1,
    @username            = 'Smtp_Username',
    @password            = 'Smtp_Password';
GO
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

データベースメールアカウントのデータベースメールプロファイルへの追加

データベースメールアカウントをデータベースメールプロファイルに追加するには、sysmail_add_profileaccount_sp ストアドプロシージャを使用します。次の例では、SES アカウントを Notifications プロファイルに追加します。

プロファイルにアカウントを追加するには

- 次の SQL 文を使用します。

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
    @profile_name        = 'Notifications',
```

```
    @account_name        = 'SES',
    @sequence_number     = 1;
GO
```

データベースメールプロファイルへのユーザーの追加

msdb データベースプリンシパルにデータベースメールプロファイルを使用するアクセス権限を付与するには、sysmail_add_principalprofile_sp ストアドプロシージャを使用します。プリンシパル は、SQL Server リソースをリクエストできるエンティティです。データベースプリンシパルは、SQL Server 認証ユーザー、Windows 認証ユーザー、または Windows 認証グループにマッピングする必要があります。

次の例では、Notifications プロファイルへのパブリックアクセスを許可します。

プロファイルにユーザーを追加するには

- 次の SQL 文を使用します。

```
USE msdb
GO

EXECUTE msdb.dbo.sysmail_add_principalprofile_sp
    @profile_name       = 'Notifications',
    @principal_name     = 'public',
    @is_default         = 1;
GO
```

データベースメールの Amazon RDS ストアドプロシージャと関数

Microsoft が提供するストアドプロシージャにより、データベースメールの使用 (アカウントやプロファイルの作成、一覧表示、更新、削除) が可能です。加えて、RDS は、次の表に示すストアドプロシージャおよびデータベースメールの機能を提供します。

プロシージャ/関数	説明
rds_fn_sysmail_allitems	送信メッセージ (他のユーザーの送信メッセージを含む) を表示します。

プロシージャ/関数	説明
rds_fn_sysmail_event_log	イベント (他のユーザーの送信メッセージのイベントを含む) を表示します。
rds_fn_sysmail_mailattachments	添付ファイル (他のユーザーの送信メッセージの添付ファイルも含む) を表示します。
rds_sysmail_control	メールキュー (DatabaseMail.exe プロセス) を開始および停止します。
rds_sysmail_delete_mailitems_sp	すべてのユーザーが送信した E メールメッセージをデータベースメール内部テーブルから削除します。

データベースメールを使用した E メールメッセージの送信

データベースメールを使用して E メールメッセージを送信するには、sp_send_dbmail ストアドプロシージャを使用します。

Usage

```
EXEC msdb.dbo.sp_send_dbmail
@profile_name = 'profile_name',
@recipients = 'recipient1@example.com[; recipient2; ... recipientn]',
@subject = 'subject',
@body = 'message_body',
[@body_format = 'HTML'],
[@file_attachments = 'file_path1; file_path2; ... file_pathn'],
[@query = 'SQL_query'],
[@attach_query_result_as_file = 0|1]';
```

以下のパラメータは必須です。

- @profile_name – メッセージの送信元となるデータベースメールプロファイル名
- @recipients – メッセージの送信先となるセミコロン区切りの E メールアドレスリスト
- @subject – メッセージの件名
- @body – メッセージの本文 宣言された変数を本文として使用することもできます。

以下のパラメータはオプションです。

- @body_format – このパラメータは、HTML 形式で E メールを送信するため、宣言された変数と共に使用します。
- @file_attachments – セミコロン区切りのメッセージ添付ファイルリスト。ファイルパスは絶対パスである必要があります。
- @query – 実行する SQL クエリ クエリ結果は、ファイルで添付することも、メッセージの本文に含めることもできます。
- @attach_query_result_as_file – クエリ結果をファイルでアタッチするかどうか。[いいえ] の場合は 0、[はい] の場合は 1 に設定します。デフォルトは 0 です。

例

次の例は、E メールメッセージを送信する方法をデモンストレーションします。

Example単一の受信者へのメッセージの送信の

```
USE msdb
GO

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Automated DBMail message - 1',
     @body               = 'Database Mail configuration was successful.';
GO
```

Example複数の受信者へのメッセージの送信の

```
USE msdb
GO

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'recipient1@example.com;recipient2@example.com',
     @subject            = 'Automated DBMail message - 2',
     @body               = 'This is a message.';
GO
```

Example添付ファイルでの SQL クエリ結果の送信の

```
USE msdb
GO

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Test SQL query',
     @body               = 'This is a SQL query test.',
     @query              = 'SELECT * FROM abc.dbo.test',
     @attach_query_result_as_file = 1;
GO
```

Example HTML 形式でのメッセージの送信の

```
USE msdb
GO

DECLARE @HTML_Body as NVARCHAR(500) = 'Hi, <h4> Heading </h4> </br> See the report. <b>
 Regards </b>';

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'Test HTML message',
     @body               = @HTML_Body,
     @body_format        = 'HTML';
GO
```

Exampleデータベースで特定のイベントが発生した時の、トリガーを使用したメッセージの送信の

```
USE AdventureWorks2017
GO
IF OBJECT_ID ('Production.iProductNotification', 'TR') IS NOT NULL
DROP TRIGGER Purchasing.iProductNotification
GO

CREATE TRIGGER iProductNotification ON Production.Product
   FOR INSERT
   AS
   DECLARE @ProductInformation nvarchar(255);
```

```
   SELECT
   @ProductInformation = 'A new product, ' + Name + ', is now available for $' +
 CAST(StandardCost AS nvarchar(20)) + '!'
   FROM INSERTED i;

EXEC msdb.dbo.sp_send_dbmail
     @profile_name       = 'Notifications',
     @recipients         = 'nobody@example.com',
     @subject            = 'New product information',
     @body               = @ProductInformation;
GO
```

メッセージ、ログ、添付ファイルの表示

RDS ストアドプロシージャを使用して、メッセージ、イベントログ、および添付ファイルを表示します。

すべての E メールメッセージを表示するには

- 次の SQL クエリを使用します。

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_allitems(); --WHERE sent_status='sent' or
 'failed' or 'unsent'
```

すべての E メールイベントログを表示するには

- 次の SQL クエリを使用します。

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_event_log();
```

すべての E メールの添付ファイルを表示するには

- 次の SQL クエリを使用します。

```
SELECT * FROM msdb.dbo.rds_fn_sysmail_mailattachments();
```

メッセージの削除

`rds_sysmail_delete_mailitems_sp` ストアドプロシージャを使用して、メッセージを削除します。

> **Note**
>
> RDS は、データベースメール履歴データのサイズが 1 GB に達すると、メールテーブル項目を自動的に削除します。保持期間は最短 24 時間です。
> メールアイテムを長期間保持する場合、アーカイブできます。詳細については、Microsoft ドキュメントの「データベースメールメッセージとイベントログをアーカイブする SQL Server Agent ジョブの作成」を参照してください。

E メールメッセージをすべて削除するには

- 次の SQL 文を使用します。

```
DECLARE @GETDATE datetime
SET @GETDATE = GETDATE();
EXECUTE msdb.dbo.rds_sysmail_delete_mailitems_sp @sent_before = @GETDATE;
GO
```

特定のステータスの E メールメッセージをすべて削除するには

- 失敗したメッセージをすべて削除するには、次の SQL ステートメントを使用します。

```
DECLARE @GETDATE datetime
SET @GETDATE = GETDATE();
EXECUTE msdb.dbo.rds_sysmail_delete_mailitems_sp @sent_status = 'failed';
GO
```

メールキューの開始と停止

DB メールキューを開始および停止するには、次の手順に従います。

トピック

- メールキューの開始

- メールキューの停止

メールキューの開始

rds_sysmail_control ストアドプロシージャを使用して、データベースメールプロセスを開始します。

> **Note**
>
> データベースメールを有効にすると、メールキューが自動的に開始します。

メールキューを開始するには

- 次の SQL 文を使用します。

```
EXECUTE msdb.dbo.rds_sysmail_control start;
GO
```

メールキューの停止

rds_sysmail_control ストアドプロシージャを使用して、データベースメールプロセスを停止します。

メールキューを停止するには

- 次の SQL 文を使用します。

```
EXECUTE msdb.dbo.rds_sysmail_control stop;
GO
```

添付ファイルの使用

SQL Server の RDS からのデータベースメールメッセージでは、次の添付ファイル拡張子をサポートしていません。.ade、.adp、.apk、.appx、.appxbundle、.bat、.bak、.cab、.chm、.cmd、.com、.cpl、.dll、.dmg、.exおよび.wsh

データベースメールは、現在のユーザーの Microsoft Windows セキュリティコンテキストを使用して、ファイルへのアクセスを制御します。SQL Server 認証でログインするユーザーは、`@file_attachments` ストアドプロシージャで`sp_send_dbmail` パラメータを使用してファイルをアタッチすることはできません。Windows では、リモートコンピュータから別のリモートコンピュータに、SQL Server が認証情報を提供することはできません。したがって、データベースメールは、SQL Server を実行しているコンピュータ以外のコンピュータからコマンドを実行すると、ネットワーク共有からファイルをアタッチすることはできません。

ただし、SQL Server Agent ジョブを使用して、ファイルをアタッチすることができます。SQL Server Agent の詳細については、Microsoft ドキュメントの「Amazon RDS 用 SQL Server エージェントの使用」および「SQL Server Agent」を参照してください。

マルチ AZ 配置に関する考慮事項

マルチ AZ DB インスタンスでデータベースメールを設定しても、設定はセカンダリに自動的には反映されません。マルチ AZ インスタンスをシングル AZ インスタンスに変換し、データベースメールを設定した後に、DB インスタンスをマルチ AZ に戻すことをお勧めします。次に、プライマリノードとセカンダリノードの両方に、データベースメールの設定があります。

データベースメールを設定したマルチ AZ インスタンスからリードレプリカを作成すると、レプリカはその設定を継承しますが、SMTP サーバーのパスワードは継承しません。パスワードを使用して、データベースメールアカウントを更新します。

SMTP (ポート 25) 制限の削除

デフォルトでは、AWS は RDS for SQL Server DB インスタンスの SMTP (ポート 25) でのアウトバウンドトラフィックをブロックします。これは、Elastic Network Interface 所有者のポリシーに基づいてスパムを防ぐために行われます。必要に応じて、この制限を削除できます。詳細については、「Amazon EC2 インスタンスまたは Lambda 関数のポート 25 の制限を解除するにはどうすればよいですか?」を参照してください。

Amazon RDS for SQL Server の tempdb データベースに対するインスタンスストアのサポート

インスタンスストアは、DB インスタンスに一時ブロックレベルのストレージを提供します。このストレージは、ホストコンピュータに物理的にアタッチされたディスク上にあります。これらのディスクには、ソリッドステートドライブ (SSD) に基づく不揮発性メモリエクスプレス (NVMe) インスタンスストレージがあります。このストレージは、低レイテンシー、非常に高いランダム I/O パフォーマンス、高いシーケンシャル読み取りスループットを実現するために最適化されています。

`tempdb` データファイルと `tempdb` ログファイルをインスタンスストアに配置することで、Amazon EBS に基づく標準ストレージと比べて、読み取りおよび書き込みのレイテンシーを低く抑えることができます。

> **Note**
>
> SQL Server データベースファイルとデータベースログファイルは、インスタンスストアに配置されません。

インスタンスストアの有効化

RDS が次のいずれかのインスタンスクラスを使用して DB インスタンスをプロビジョニングすると、`tempdb` データベースは自動的にインスタンスストアに配置されます。

- db.m5d
- db.r5d
- db.x2iedn
- db.x2m

インスタンスストアを有効にするには、次のいずれかの操作を行います。

- これらのインスタンスタイプの 1 つを使用して SQL Server DB インスタンスを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の SQL Server DB インスタンスを変更して、そのうちの 1 つを使用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

インスタンスストアは、これらのインスタンスタイプの 1 つ以上がサポートされているすべての AWS リージョンで使用できます。db.m5d、db.r5d、db.x2iedn と db.x2m インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。Amazon RDS for SQL Server でサポートされるインスタンスクラスの詳細については、Microsoft SQL Server の DB インスタンスクラスのサポート を参照してください。

ファイルの場所とサイズに関する考慮事項

インスタンスストアがないインスタンスでは、RDS は tempdb データとログファイルを D:\rdsdbdata\DATA ディレクトリに保存します。デフォルトでは、どちらのファイルも 8 MB から始まります。

インスタンスストアがあるインスタンスでは、RDS は tempdb データとログファイルを T:\rdsdbdata\DATA ディレクトリに保存します。

tempdb にデータファイル (tempdb.mdf) とログファイル (templog.ldf) が 1 つしかない場合、templog.ldf はデフォルトでは 8 MB から始まり、tempdb.mdf はインスタンスのストレージ容量の 80%以上から始まります。ストレージ容量の 20% または 200 GB のどちらか小さい方を、無料で起動できます。複数の tempdb データファイルは 80% のディスク領域を均等に分割しますが、ログファイルの初期サイズは常に 8 MB です。

例えば、DB インスタンスクラスを db.m5.2xlarge から db.m5d.2xlarge に変更すると、tempdb データファイルのサイズはそれぞれ 8 MB から合計 234 GB に増加します。

> **Note**
>
> インスタンスストア (tempdb) の T:\rdsdbdata\DATA データとログファイルに加えて、データボリューム (tempdb) に追加の D:\rdsdbdata\DATA データとログファイルを作成することもできます。これらのファイルの初期サイズは常に 8 MB です。

バックアップに関する考慮事項

バックアップを長期間保持する必要がありますが、時間の経過とともにコストがかかる場合があります。tempdb データブロックとログブロックは、ワークロードに応じて非常に頻繁に変更されることがあります。これにより、DB スナップショットのサイズが大幅に増加する可能性があります。

tempdb がインスタンスストアにある場合、スナップショットに一時ファイルは含まれません。つまり、EBS のみのストレージと比べて、スナップショットのサイズが小さくなり、空いたバックアップ割り当ての使用量が少なくなります。

ディスクがいっぱいになるエラー

インスタンスストア内の使用可能な領域をすべて使用すると、次のようなエラーが表示されることがあります。

- データベース「tempdb」のトランザクションログは「ACTIVE_TRANSACTION」のため、いっぱいになりました。
- PRIMARY」ファイルグループがいっぱいであるため、データベース「tempdb」のオブジェクト「dbo.SORT の一時実行ストレージ: 140738941419520」に領域を割り当てられませんでした。不要なファイルの削除、ファイルグループ内のオブジェクトの削除、ファイルグループへのファイルの追加、またはファイルグループ内の既存のファイルの自動拡張を有効に設定して、ディスク領域を作ります。

インスタンスストアがいっぱいになると、次の項目のうちから 1 つまたは複数を実行できます。

- ワークロードや tempdb の使用方法を調整します。
- より多くの NVMe ストレージを持つ DB インスタンスクラスを使用するように拡張します。
- インスタンスストアの使用を停止し、EBS ストレージのみを持つインスタンスクラスを使用します。
- EBS ボリューム上の tempdb のセカンダリデータまたはログファイルを追加して、混合モードを使用します。

インスタンスストアの削除

インスタンスストアを削除するには、インスタンスストアをサポートしないインスタンスタイプ (db.m5、db.r5、db.x1e など) を使用するように SQL Server DB インスタンスを変更します。

> **Note**
>
> インスタンスストアを削除すると、一時ファイルは D:\rdsdbdata\DATA ディレクトリに移動され、8 MB に縮小されます。

Amazon RDS for Microsoft SQL Server で拡張イベントを使用する

Microsoft SQL Server の拡張イベントを使用して、Amazon RDS for SQL Server のデバッグおよびトラブルシューティング情報をキャプチャできます。拡張イベントは、Microsoft によって非推奨にされている SQL Trace と Server Profiler を置き換えます。拡張イベントは、プロファイラートレースに似ていますが、トレースされるイベントをより細かく制御できます。拡張イベントは、Amazon RDS で SQL Server バージョン 2016 以降でサポートされています。詳細については、Microsoft のドキュメントの「Extended events overview」を参照してください。

Amazon RDS for SQL Server のマスターユーザー特権を持つユーザーの場合、拡張イベントは自動的にオンになります。

トピック

- 制限と推奨事項
- RDS for SQL Server での拡張イベントの設定
- マルチ AZ 配置に関する考慮事項
- 拡張イベントファイルのクエリ

制限と推奨事項

RDS for SQL Server で拡張イベントを使用する場合、次の制限が適用されます。

- 拡張イベントは、Enterprise エディションと Standard エディションでのみサポートされます。
- デフォルトの拡張イベントセッションは変更できません。
- セッションメモリのパーティションモードを NONE に設定してください。
- セッションイベント保持モードは、ALLOW_SINGLE_EVENT_LOSS または ALLOW_MULTIPLE_EVENT_LOSS のいずれかになります。
- Event Tracing for Windows (ETW) ターゲットはサポートされていません。
- ファイルターゲットが D:\rdsdbdata\log ディレクトリにあることを確認します。
- ペアが一致するターゲットの場合は、respond_to_memory_pressure プロパティを 1 に設定します。
- リングバッファーターゲットメモリは 4 MB を超えることはできません。
- 次のアクションはサポートされていません。
  - debug_break

  - create_dump_all_threads
  - create_dump_single_threads
- rpc_completed イベント
  は、15.0.4083.2、14.0.3370.1、13.0.5865.1、12.0.6433.1、11.0.7507.2 以降のバージョンでサポートされています。

RDS for SQL Server での拡張イベントの設定

RDS for SQL Server では、拡張イベントセッションの特定のパラメータ値を設定できます。次の表では、設定可能なパラメータについて説明しています。

パラメータ名	説明
xe_session_max_memory	イベントバッファリングのためにセッションに割り当てる セッションの max_memory 設定に対応しています。
xe_session_max_event_size	大きなイベントで許可される最大メモリサイズを指定しま _size 設定に対応しています。
xe_session_max_dispatch_latency	拡張イベントセッションターゲットにディスパッチされる 定します。この値は、イベントセッションの max_dispa
xe_file_target_size	ファイルターゲットの最大サイズを指定します。この値に 対応します。
xe_file_retention	イベントセッションのファイルターゲットによって生成さ

> **Note**
>
> xe_file_retention をゼロに設定すると、これらのファイルのロックが SQL Server によって解放された後、.xel ファイルが自動的に削除されます。ロックは、.xel ファイルが xe_file_target_size で設定されたサイズ制限に達すると解放されます。

rdsadmin.dbo.rds_show_configuration ストアドプロシージャを使用して、これらのパラメータの現在の値を表示できます。例えば、xe_session_max_memory の現在の設定を表示するには、次の SQL 文を使用します。

```
exec rdsadmin.dbo.rds_show_configuration 'xe_session_max_memory'
```

rdsadmin.dbo.rds_set_configuration ストアドプロシージャを使用して変更することができます。例えば、xe_session_max_memory を 4 MB に設定するには、次の SQL ステートメントを使用します。

```
exec rdsadmin.dbo.rds_set_configuration 'xe_session_max_memory', 4
```

マルチ AZ 配置に関する考慮事項

プライマリ DB インスタンスで拡張イベントセッションを作成すると、そのセッションはスタンバイレプリカに伝達されません。新しいプライマリ DB インスタンスで、拡張イベントセッションをフェイルオーバーして作成できます。または、マルチ AZ 設定を削除してから再び追加して、拡張イベントセッションをスタンバイレプリカに伝達することもできます。RDS は、スタンバイレプリカ上のデフォルト以外の拡張イベントセッションをすべて停止し、これらのセッションがスタンバイ上のリソースを消費しません。このため、スタンバイレプリカがプライマリ DB インスタンスになった後、新しいプライマリで拡張イベントセッションを手動で開始するようにしてください。

> **Note**
>
> このアプローチは、Always On 可用性グループとデータベースミラーリングの両方に適用されます。

SQL Server エージェントジョブを使用して、スタンバイレプリカを追跡し、スタンバイがプライマリになる場合はセッションを開始することもできます。例えば、SQL Server エージェントのジョブステップで次のクエリを使用して、プライマリ DB インスタンスでイベントセッションを再開します。

```
BEGIN
    IF (DATABASEPROPERTYEX('rdsadmin','Updateability')='READ_WRITE'
    AND DATABASEPROPERTYEX('rdsadmin','status')='ONLINE'
    AND (DATABASEPROPERTYEX('rdsadmin','Collation') IS NOT NULL OR
 DATABASEPROPERTYEX('rdsadmin','IsAutoClose')=1)
    )
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM sys.dm_xe_sessions WHERE name='xe1')
            ALTER EVENT SESSION xe1 ON SERVER STATE=START
        IF NOT EXISTS (SELECT 1 FROM sys.dm_xe_sessions WHERE name='xe2')
```

```
            ALTER EVENT SESSION xe2 ON SERVER STATE=START
    END
END
```

このクエリにより、イベントセッション xe1 と xe2 が停止状態の場合は、これらのイベントセッションがプライマリ DB インスタンスで再開されます。また、このクエリに便利な間隔のスケジュールを追加することもできます。

拡張イベントファイルのクエリ

SQL Server Management Studio または sys.fn_xe_file_target_read_file 関数を使用して、ファイルターゲットを使用する拡張イベントのデータを表示できます。この関数の詳細については、Microsoft ドキュメントの sys.fn_xe_file_target_read_file (Transact-SQL) を参照してください。

拡張イベントファイルターゲットは、RDS for SQL Server の D:\rdsdbdata\log ディレクトリにのみファイルを書き込むことができます。

例えば、次の SQL クエリを使用して、名前が xe で始まる拡張イベントセッションのすべてのファイルの内容を一覧表示します。

```
SELECT * FROM sys.fn_xe_file_target_read_file('d:\rdsdbdata\log\xe*', null,null,null);
```

RDS for SQL Server によるトランザクションログのバックアップへのアクセス

RDS for SQL Server のトランザクションログのバックアップにアクセスすることで、データベースのトランザクションログのバックアップファイルを一覧表示し、ターゲット Amazon S3 バケットにコピーできます。Amazon S3 バケットにトランザクションログのバックアップをコピーすることで、それらをデータベースの完全バックアップや差分バックアップと組み合わせて使用し、特定の時点でのデータベース復元を実行できます。RDS ストアドプロシージャを使用して、トランザクションログのバックアップへのアクセスを設定し、利用可能なトランザクションログバックアップを一覧表示して、Amazon S3 バケットにコピーします。

トランザクションログのバックアップにアクセスすると、次のような機能と利点を利用できます。

- RDS for SQL Server DB インスタンスにあるデータベースの利用可能なトランザクションログのバックアップのメタデータを一覧表示して表示します。
- 使用可能なトランザクションログのバックアップを RDS for SQL Server からターゲット Amazon S3 バケットにコピーします。
- DB インスタンス全体を復元しなくても、データベースのポイントインタイム復元を実行できます。DB クラスターのポイントインタイム復元の方法については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

可用性およびサポート

トランザクションログのバックアップへのアクセスは、すべての AWS リージョンでサポートされています。トランザクションログバックアップへのアクセスは、Amazon RDS でサポートされている Microsoft SQL Server のすべてのエディションとバージョンで使用できます。

要件

トランザクションログのバックアップへのアクセスを有効にする前に、以下の要件を満たす必要があります。

- DB インスタンスで自動バックアップを有効に設定し、バックアップの保存期間を 1 日以上の値に設定する必要があります。自動バックアップの有効化と保存ポリシーの設定の詳細については、「自動バックアップの有効化」を参照してください。
- Amazon S3 バケットは、ソース DB インスタンスと同じアカウントとリージョンに存在している必要があります。トランザクションログのバックアップへのアクセスを有効にする前に、トラン

ザクションログのバックアップファイルに使用するため、既存の Amazon S3 バケットを選択するか、新しいバケットを作成します。

- Amazon RDS によってトランザクションログファイルをコピーできるように、Amazon S3 バケットのアクセス権限ポリシーを次のように設定する必要があります。

  1. バケットのオブジェクトアカウントの所有権プロパティを [Bucket Owner Preferred] (バケット所有者優先) に設定します。
  2. 以下のポリシーを追加します。デフォルトではポリシーがないため、バケットアクセスコントロールリスト (ACL) を使用してバケットポリシーを編集し、追加します。

  次の例では、ARN を使用してリソースを指定しています。リソースベースの信頼関係では SourceArn および SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与する特定のリソースへのアクセス許可を制限することをお勧めします。ARN での使用の詳細については、「Amazon リソースネーム (ARN)」および「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

  Exampleトランザクションログのバックアップへのアクセスするための Amazon S3 権限ポリシーの例

  JSON

```
   {
   "Version":"2012-10-17",
   "Statement": [
       {
           "Sid": "Only allow writes to my bucket with bucket owner full
control",
           "Effect": "Allow",
           "Principal": {
               "Service": "backups.rds.amazonaws.com"
           },
           "Action": "s3:PutObject",
           "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/{customer_path}/*",
           "Condition": {
               "StringEquals": {
                   "s3:x-amz-acl": "bucket-owner-full-control",
                   "aws:sourceAccount": "{customer_account}",
                   "aws:sourceArn": "{db_instance_arn}"
               }
```

```
            }
        }
    ]
}
```

- Amazon S3 バケットにアクセスするための AWS Identity and Access Management (IAM) ロール。IAM ロールが既にある場合はそれを使用できます。AWS マネジメントコンソール を使用して SQLSERVER_BACKUP_RESTORE オプションを追加する際に、新しい IAM ロールの作成を選択することもできます。または、手動で新しいロールを作成することもできます。SQLSERVER_BACKUP_RESTORE による IAM ロールの作成と設定の詳細については、「ネイティブバックアップおよび復元用の IAM ロールの手動作成」を参照してください。
- SQLSERVER_BACKUP_RESTORE オプションは、DB インスタンスのオプショングループに追加されている必要があります。SQLSERVER_BACKUP_RESTORE オプションの追加についての詳細は、「SQL Server のネイティブバックアップおよび復元のサポート」を参照してください。

> **Note**
>
> DB インスタンスでストレージ暗号化が有効になっている場合は、ネイティブバックアップと復元オプショングループで提供される IAM ロールに AWS KMS (KMS) アクションとキーを指定する必要があります。

オプションで、rds_restore_log ストアドプロシージャを使用してデータベースのポイントインタイム復元を実行する場合は、ネイティブバックアップ、復元オプショングループ、トランザクションログのバックアップへのアクセスに、同じ Amazon S3 パスを使用することをお勧めします。この方法により、Amazon RDS がオプショングループから復元ログ機能を実行する役割を引き受けると、同じ Amazon S3 パスからトランザクションログのバックアップを取得できるようになります。

- DB インスタンスが暗号化されている場合、暗号化タイプ (AWSマネージドキーまたはカスタマーマネージドキー) に関係なく、IAM ロールと rds_tlog_backup_copy_to_S3 ストアドプロシージャでカスタマーマネージド KMS キーを指定する必要があります。

制限と推奨事項

トランザクションログのバックアップへのアクセスには、次の制限と推奨事項があります。

- バックアップの保存期間が 1 日から 35 日の間に設定されている任意の DB インスタンスについて、過去 7 日間のトランザクションログのバックアップを一覧表示してコピーできます。

- トランザクションログのバックアップのアクセスに使用する Amazon S3 バケットは、ソース DB インスタンスと同じアカウントとリージョンに存在している必要があります。クロスアカウントおよびクロスリージョンコピーはサポートされていません。
- トランザクションログのバックアップのコピー先として設定できる Amazon S3 バケットは 1 つだけです。`rds_tlog_copy_setup` ストアドプロシージャを使用して、新しいターゲット Amazon S3 バケットを選択できます。新しいターゲット Amazon S3 バケットの選択の詳細については、「トランザクションログのバックアップへのアクセス設定」を参照してください。
- RDS インスタンスでストレージ暗号化が有効になっていない場合、`rds_tlog_backup_copy_to_S3` ストアドプロシージャを使用するときに KMS キーを指定することはできません。
- マルチアカウントコピーはサポートされていません。コピーに使用される IAM ロールは、DB インスタンスの所有者アカウント内の Amazon S3 バケットへの書き込みアクセスのみを許可します。
- RDS for SQL Server DB インスタンスでは、どのような種類のタスクでも 2 つまでしか実行できません。
- 1 つのデータベースに対して、同時に実行できるコピータスクは 1 つだけです。DB インスタンス上の複数のデータベースのトランザクションログのバックアップをコピーする場合は、データベースごとに個別のコピータスクを使用します。
- 既に Amazon S3 バケットに同じ名前で存在するトランザクションログのバックアップをコピーすると、既存のトランザクションログのバックアップは上書きされます。
- プライマリ DB インスタンスのトランザクションログのバックアップへのアクセス権が提供されているストアドプロシージャのみを実行できます。これらのストアドプロシージャは、RDS for SQL Server のリードレプリカや、マルチ AZ DB クラスターのセカンダリインスタンスでは実行できません。
- `rds_tlog_backup_copy_to_S3` ストアドプロシージャの実行中に RDS for SQL Server DB インスタンスを再起動すると、DB インスタンスがオンラインに戻ったときに、タスクは自動的に最初から再開されます。再起動前に、タスクの実行中に Amazon S3 バケットにコピーされたトランザクションログのバックアップはすべて上書きされます。
- Microsoft SQL Server システムデータベースと `RDSAdmin` データベースは、トランザクションログのバックアップにアクセスするように設定できません。
- SSE-KMS で暗号化されたバケットへのコピーはサポートされていません。

トランザクションログのバックアップへのアクセス設定

トランザクションログのバックアップへのアクセスを設定するには、要件 セクションの要件リストを入力して、次に rds_tlog_copy_setup ストアドプロシージャを実行します。この手順により、DB インスタンスレベルでトランザクションログのバックアップ機能にアクセスできるようになります。DB インスタンス上の個々のデータベースごとに実行する必要はありません。

> **Important**
>
> トランザクションログのバックアップ機能へのアクセスを設定し、使用するには、データベースユーザーに SQL Server 内の各データベースの db_owner ロールを付与する必要があります。

Example使用例:

```
exec msdb.dbo.rds_tlog_copy_setup
@target_s3_arn='arn:aws:s3:::amzn-s3-demo-bucket/myfolder';
```

以下のパラメータは必須です。

- @target_s3_arn – トランザクションログのバックアップファイルをコピーするターゲット Amazon S3 バケットの ARN。

Example Amazon S3 バケットのターゲットバケットの設定例:

```
exec msdb.dbo.rds_tlog_copy_setup @target_s3_arn='arn:aws:s3:::amzn-s3-demo-logging-bucket/mytestdb1';
```

設定を検証するには、rds_show_configuration ストアドプロシージャを呼び出します。

Example設定の検証例:

```
exec rdsadmin.dbo.rds_show_configuration @name='target_s3_arn_for_tlog_copy';
```

トランザクションログのバックアップへのアクセス先を別の Amazon S3 バケットに変更するには、現在の Amazon S3 バケット値を表示し、@target_s3_arn の新しい値を使用して、rds_tlog_copy_setup ストアドプロシージャを再実行します。

Exampleトランザクションログのバックアップにアクセスするため設定された既存の Amazon S3 バケットの表示例

```
exec rdsadmin.dbo.rds_show_configuration @name='target_s3_arn_for_tlog_copy';
```

Example新しいターゲット Amazon S3 バケットへの更新例

```
exec msdb.dbo.rds_tlog_copy_setup @target_s3_arn='arn:aws:s3:::amzn-s3-demo-logging-bucket1/mynewfolder';
```

利用可能なトランザクションログのバックアップの一覧表示

RDS for SQL Server では、完全な復旧モデルを使用するように設定されたデータベースで、DB インスタンスのバックアップ保存期間を 1 日以上に設定すると、トランザクションログのバックアップが自動で有効になります。トランザクションログのバックアップへのアクセスを有効にすると、最大 7 日間分のトランザクションログのバックアップを Amazon S3 バケットにコピーできるようになります。

トランザクションログのバックアップへのアクセスを有効にすると、それを使用して、利用可能なトランザクションログバックアップファイルの一覧表示と、コピーを開始できます。

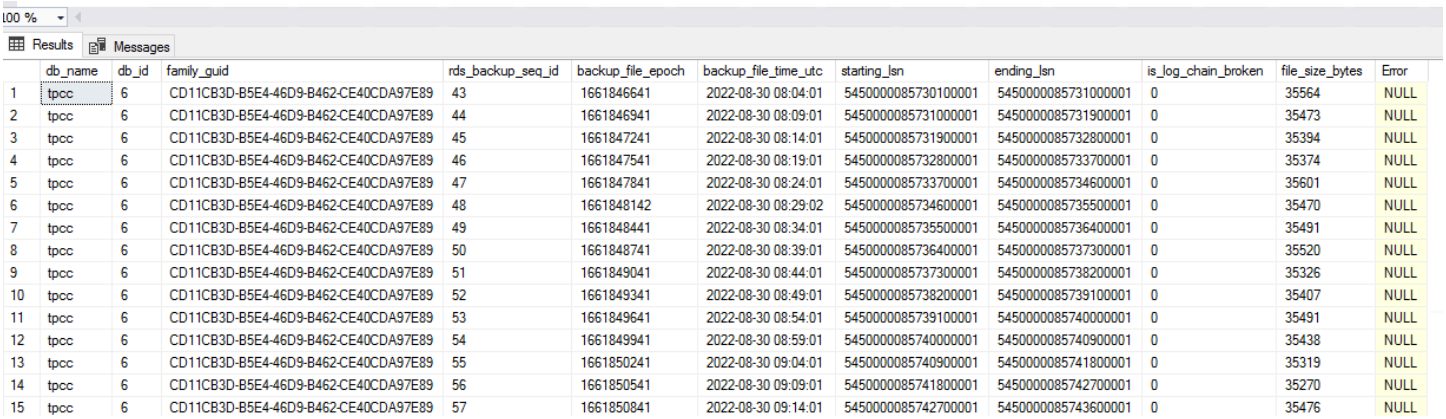
トランザクションログのバックアップの一覧表示

個々のデータベースで利用可能なすべてのトランザクションログのバックアップを一覧表示するには、rds_fn_list_tlog_backup_metadata 関数を呼び出します。関数を呼び出すときは、ORDER BY または WHERE 句を使用できます。

Example利用可能なトランザクションログのバックアップファイルの一覧表示とフィルタリング例

```
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename');
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename') WHERE
 rds_backup_seq_id = 3507;
```

```
SELECT * from msdb.dbo.rds_fn_list_tlog_backup_metadata('mydatabasename') WHERE
 backup_file_time_utc > '2022-09-15 20:44:01' ORDER BY backup_file_time_utc DESC;
```



100 %

Results Messages

	db_name	db_id	family_guid	rds_backup_seq_id	backup_file_epoch	backup_file_time_utc	starting_lsn	ending_lsn	is_log_chain_broken	file_size_bytes	Error
1	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	43	1661846641	2022-08-30 08:04:01	5450000085730100001	5450000085731000001	0	35564	NULL
2	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	44	1661846941	2022-08-30 08:09:01	5450000085731000001	5450000085731900001	0	35473	NULL
3	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	45	1661847241	2022-08-30 08:14:01	5450000085731900001	5450000085732800001	0	35394	NULL
4	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	46	1661847541	2022-08-30 08:19:01	5450000085732800001	5450000085733700001	0	35374	NULL
5	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	47	1661847841	2022-08-30 08:24:01	5450000085733700001	5450000085734600001	0	35601	NULL
6	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	48	1661848142	2022-08-30 08:29:02	5450000085734600001	5450000085735500001	0	35470	NULL
7	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	49	1661848441	2022-08-30 08:34:01	5450000085735500001	5450000085736400001	0	35491	NULL
8	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	50	1661848741	2022-08-30 08:39:01	5450000085736400001	5450000085737300001	0	35520	NULL
9	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	51	1661849041	2022-08-30 08:44:01	5450000085737300001	5450000085738200001	0	35326	NULL
10	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	52	1661849341	2022-08-30 08:49:01	5450000085738200001	5450000085739100001	0	35407	NULL
11	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	53	1661849641	2022-08-30 08:54:01	5450000085739100001	5450000085740000001	0	35491	NULL
12	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	54	1661849941	2022-08-30 08:59:01	5450000085740000001	5450000085740900001	0	35438	NULL
13	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	55	1661850241	2022-08-30 09:04:01	5450000085740900001	5450000085741800001	0	35319	NULL
14	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	56	1661850541	2022-08-30 09:09:01	5450000085741800001	5450000085742700001	0	35270	NULL
15	tpcc	6	CD11CB3D-B5E4-46D9-B462-CE40CDA97E89	57	1661850841	2022-08-30 09:14:01	5450000085742700001	5450000085743600001	0	35476	NULL

`rds_fn_list_tlog_backup_metadata` 関数は、次の出力を返します。

列名	データ型	説明
db_name	sysname	トランザクションログのバックアップを一覧表示するために指定されたデータベース名。
db_id	int	入力パラメータ db_name の内部データベース識別子。
family_guid	uniqueidentifier	作成時の元のデータベースの一意の ID。この値は、データベースが復元されても、データベース名が異なっても変わりません。
rds_backup_seq_id	int	RDS が各トランザクションログのバックアップファイルのシーケンス番号を保持するために内部で使用する ID。
backup_file_epoch	bigint	トランザクションのバックアップファイルが生成されたエポック時間。
backup_file_time_utc	datetime	backup_file_epoch 値の UTC 時刻変換後の値。

列名	データ型	説明
starting_lsn	numeric(25,0)	トランザクションログのバックアップファイルの最初または最も古いログレコードのログシーケンス番号。
ending_lsn	numeric(25,0)	トランザクションログのバックアップファイルの最後またはその次のログレコードのログシーケンス番号。
is_log_chain_broken	bit	現在のトランザクションログのバックアップファイルと、以前のトランザクションログのバックアップファイル間でログチェーンが壊れているかどうかを示す boolean 値。
file_size_bytes	bigint	トランザクションのバックアップセットのサイズ (バイト単位)。
Error	varCHAR(4000)	rds_fn_list_tlog_backup_metadata 関数が例外をスローした場合のエラーメッセージ。例外がない場合は NULL です。

トランザクションログのバックアップのコピー

個々のデータベースの利用可能なトランザクションログのバックアップセットを Amazon S3 バケットにコピーするには、rds_tlog_backup_copy_to_S3 ストアドプロシージャを呼び出します。rds_tlog_backup_copy_to_S3 ストアドプロシージャでは、トランザクションログのバックアップをコピーする新しいタスクを開始します。

> **Note**
>
> rds_tlog_backup_copy_to_S3 ストアドプロシージャでは、is_log_chain_broken 属性を検証せずにトランザクションログのバックアップをコピーします。このため、rds_tlog_backup_copy_to_S3 ストアドプロシージャを実行する前に、ログチェーンが壊れていないことを手動で確認する必要があります。追加の説明については、「トランザクションログのバックアップログチェーンの検証」を参照してください。

Example **rds_tlog_backup_copy_to_S3** ストアドプロシージャの使用例

```
exec msdb.dbo.rds_tlog_backup_copy_to_S3
 @db_name='mydatabasename',
 [@kms_key_arn='arn:aws:kms:region:account-id:key/key-id'],
 [@backup_file_start_time='2022-09-01 01:00:15'],
 [@backup_file_end_time='2022-09-01 21:30:45'],
 [@starting_lsn=149000000112100001],
 [@ending_lsn=149000000120400001],
 [@rds_backup_starting_seq_id=5],
 [@rds_backup_ending_seq_id=10];
```

次の入力パラメータが利用可能です。

パラメータ	説明
@db_name	トランザクションログのバックアップをコピーするためのデータベース名。
@kms_key_arn	カスタマーマネージド KMS キー。AWS マネージド KMS キーで DB インスタンスを暗号化する場合は、カスタマーマネージドキーを作成する必要があります。カスタマーマネージドキーで DB インスタンスを暗号化する場合、同じ KMS キー ARN を使用できます。
@backup_file_start_time	rds_fn_list_tlog_backup_metadata 関数の [backup_file_time_utc] 列から提供された UTC タイムスタンプ。
@backup_file_end_time	rds_fn_list_tlog_backup_metadata 関数の [backup_file_time_utc] 列から提供された UTC タイムスタンプ。
@starting_lsn	rds_fn_list_tlog_backup_metadata 関数の [starting_lsn] 列から提供されたログシーケンス番号 (LSN)
@ending_lsn	rds_fn_list_tlog_backup_metadata 関数の [ending_lsn] 列から提供されたログシーケンス番号 (LSN)。
@rds_backup_starting_seq_id	rds_fn_list_tlog_backup_metadata 関数の [rds_backup_seq_id] 列から提供されたシーケンス ID。

パラメータ	説明
@rds_backup_ending_seq_id	rds_fn_list_tlog_backup_metadata 関数の [rds_backup_seq_id] 列から提供されたシーケンス ID。

時間、LSN、シーケンス ID のいずれかのパラメータセットを指定できます。必要なパラメータは 1 セットだけです。

また、どのセットでもパラメータを 1 つだけ指定できます。例えば、backup_file_end_time パラメータの値のみを指定すると、7 日間の制限内であれば、それ以前に利用可能なすべてのトランザクションログのバックアップファイルが Amazon S3 バケットにコピーされます。

rds_tlog_backup_copy_to_S3 ストアドプロシージャの有効な入力パラメータの組み合わせは次のとおりです。

指定されたパラメータ	予想される結果
exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name = 'testdb1', @backup_file_start_time='2022-08-23 00:00:00', @backup_file_end_time='2022-08-30 00:00:00';	過去 7 日間のトランザクションログのバックアップをコピーします。このバックアップは、指定された backup_file_start_time から backup_file_end_time の範囲に存在します。この例では、ストアドプロシージャは「2022-08-23 00:00:00」から「2022-08-30 00:00:00」の間に生成されたトランザクションログのバック

指定されたパラメータ	予想される結果	
	アップをコピーします。	
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name = 'testdb1', @backup_file_start_time='2022-08-23 00:00:00';`	指定された backup_file_start_time を起点として、過去 7 日間のトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは「2022-08-23 00:00:00」のトランザクションログのバックアップを最新のトランザクションログのバックアップにコピーします。	

指定されたパラメータ	予想される結果
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name = 'testdb1', @backup_file_end_time='2022-08-30 00:00:00';`	指定された `backup_file_end_time` まで、過去 7 日間のトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは「2022-08-23 00:00:00」から「2022-08-30 00:00:00」までのトランザクションログのバックアップをコピーします。

指定されたパラメータ	予想される結果	
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @starting_lsn =14900000 00040007, @ending_lsn = 149000000 0050009;`	過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。このバックアップは、指定された starting_lsn から ending_lsn の範囲にあります。この例では、ストアドプロシージャは、LSN 範囲が 1490000000040007 から 149000000 0050009 までの過去 7 日間のトランザクションログのバックアップをコピーします。	

指定されたパラメータ	予想される結果
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @starting_lsn =14900000 00040007;`	指定された starting_lsn から、過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは LSN 1490000000040007 からのトランザクションログのバックアップを最新のトランザクションログのバックアップにコピーします。
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @ending_lsn =14900000 00050009;`	指定された ending_lsn まで、過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは、LSN が 1490000000050009 までの過去 7 日間のトランザクションログのバックアップをコピーします。

指定されたパラメータ	予想される結果	
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000, @rds_backup_ending_seq_id= 5000;`	過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。このバックアップは、指定された `rds_backup_starting_seq_id` から `rds_backup_ending_seq_id` の範囲に存在します。この例では、ストアドプロシージャは、seq_id 2000 から seq_id 5000 までの rds バックアップシーケンス ID の範囲内で、過去 7 日間のトランザクションログのバックアップをコピーします。	

指定されたパラメータ	予想される結果
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000;`	指定された `rds_backup_starting_seq_id` から、過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは seq_id 2000 から始まる最新のトランザクションログのバックアップを最新のトランザクションログのバックアップにコピーします。
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_ending_seq_id= 5000;`	指定された `rds_backup_ending_seq_id` まで、過去 7 日間の利用可能なトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは、seq_id 5000 までの過去 7 日間のトランザクションログのバックアップをコピーします。

指定されたパラメータ	予想される結果	
`exec msdb.dbo.rds_tlog_backup_copy_to_S3 @db_name='testdb1', @rds_backup_starting_seq_id= 2000; @rds_backup_ending_seq_id= 2000;`	過去 7 日以内に利用可能な場合、指定された rds_backup_starting_seq_id で 1 つのトランザクションログのバックアップをコピーします。この例では、ストアドプロシージャは、seq_id が 2000 の 1 つのトランザクションログのバックアップをコピーします (過去 7 日以内に存在する場合)。	

トランザクションログのバックアップログチェーンの検証

トランザクションログのバックアップにアクセスするように設定されたデータベースでは、自動バックアップ保持が有効になっている必要があります。自動バックアップ保持により、DB インスタンスのデータベースが FULL 復旧モデルに設定されます。データベースのポイントインタイム復元をサポートするには、データベース復旧モデルを変更しないでください。データベース復旧モデルを変更すると、ログチェーンが壊れる可能性があります。データベースは FULL 復旧モデルに設定しておくことをお勧めします。

トランザクションログのバックアップをコピーする前にログチェーンを手動で検証するには、rds_fn_list_tlog_backup_metadata 関数を呼び出して is_log_chain_broken 列の値を確認します。値が「1」の場合、現在のログのバックアップと前回のログのバックアップの間でログチェーンが壊れていたことを示します。

次の例は、rds_fn_list_tlog_backup_metadata ストアドプロシージャからの出力のログチェーンが壊れていることを示しています。

rds_sequence_id	first_lsn	last_lsn	is_log_chain_broken
43	90023	90457	0
44	90457	90985	0
45	90987	92034	1

通常のログチェーンでは、特定の rds_sequence_id の first_lsn のログシーケンス番号 (LSN) 値は、前の rds_sequence_id の last_lsn の値と一致する必要があります。この図では、rds_sequence_id が 45 の first_lsn の値は 90987 ですが、その前の rds_sequence_id が 44 の last_lsn 値 90985 と一致しません。

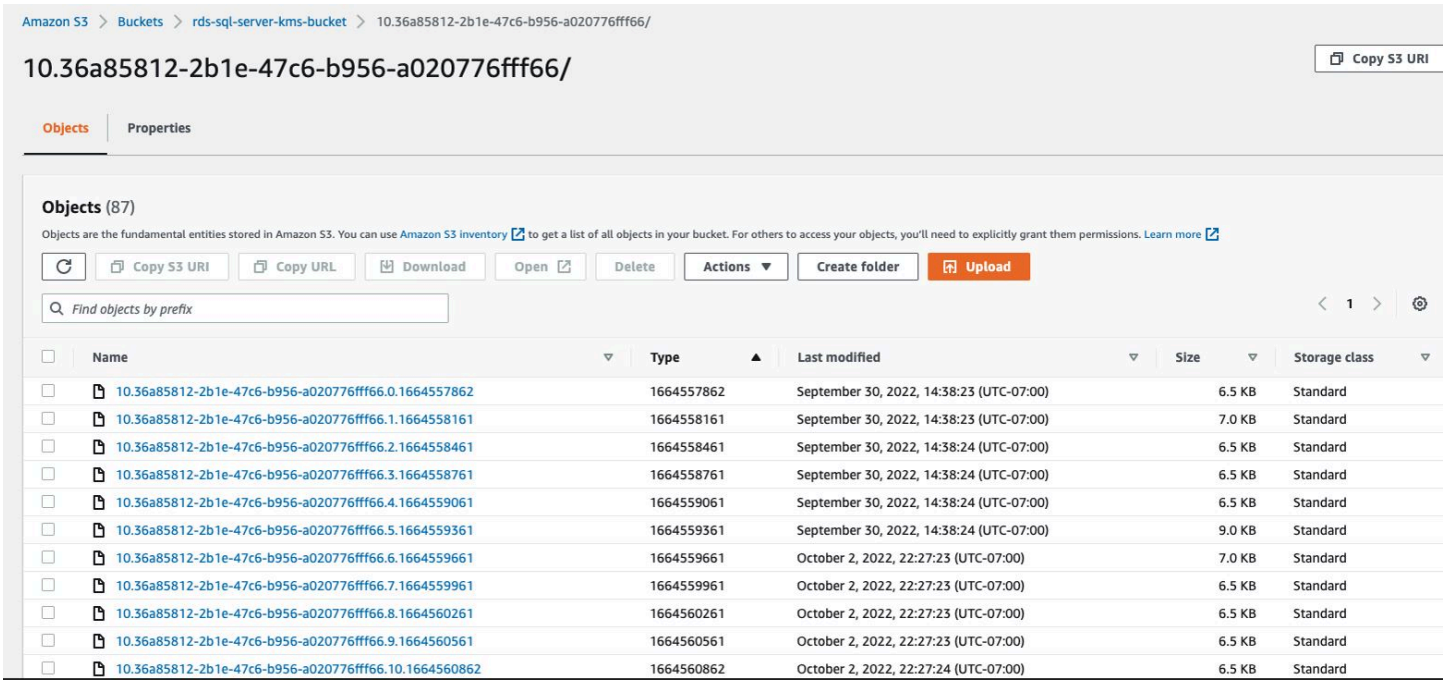
SQL Server のトランザクションログアーキテクチャとログシーケンス番号の詳細については、Microsoft SQL Server ドキュメントの「トランザクションログの論理アーキテクチャ」を参照してください。

Amazon S3 バケットフォルダおよびファイル構造

Amazon S3 バケット内のトランザクションログのバックアップには、以下の標準構造と命名規則があります。

- 各データベースの `target_s3_arn` パスの下に、`{db_id}.{family_guid}` という命名構造を持つ新しいフォルダが作成されます。
- フォルダ内のトランザクションログのバックアップは、`{db_id}.{family_guid}.{rds_backup_seq_id}.{backup_file_epoch}` のようなファイル名構造を持ちます。
- `rds_fn_list_tlog_backup_metadata` 関数を使用すると、`family_guid,db_id,rds_backup_seq_id and backup_file_epoch` の詳細を表示できます。

以下の例では、Amazon S3 バケット内の一連のトランザクションログのバックアップのフォルダおよびファイル構造を示しています。



タスクのステータスの追跡

コピータスクのステータスを追跡するには、rds_task_status ストアドプロシージャを呼び出します。パラメータを何も指定しない場合、ストアドプロシージャによりすべてのタスクのステータスが返されます。

Example使用例:

```
exec msdb.dbo.rds_task_status
  @db_name='database_name',
  @task_id=ID_number;
```

以下のパラメータはオプションです。

- @db_name – タスクのステータスを表示するデータベースの名前。
- @task_id – タスクのステータスを表示するタスクの ID。

Example特定タスク ID のステータスの一覧表示例:

```
exec msdb.dbo.rds_task_status @task_id=5;
```

Example特定データベースおよびタスクのステータスの一覧表示例:

```
exec msdb.dbo.rds_task_status@db_name='my_database',@task_id=5;
```

Example特定データベースのすべてのタスクおよびステータスの一覧表示例:

```
exec msdb.dbo.rds_task_status @db_name='my_database';
```

Example現在の DB インスタンスのすべてのタスクおよびステータスの一覧表示例:

```
exec msdb.dbo.rds_task_status;
```

タスクのキャンセル

実行中のタスクをキャンセルするには、rds_cancel_task ストアドプロシージャを呼び出します。

Example使用例:

```
exec msdb.dbo.rds_cancel_task @task_id=ID_number;
```

以下のパラメータは必須です。

- @task_id – キャンセルするタスクの ID。rds_task_status ストアドプロシージャを呼び出すことにより、タスク ID を表示できます。

実行中のタスクの表示とキャンセルの詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

トランザクションログのバックアップへのアクセスについてのトラブルシューティング

トランザクションログのバックアップへのアクセスにストアドプロシージャを使用する場合、次のような問題が発生する場合があります。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_copy_setup	この DB インスタンスでは、バックアップは無効になっています。DB インスタンスのバックアップを「1」以上の保持期間で有効にして、もう一度試してください。	DB インスタンスの自動バックアップを有効化する。	DB インスタンスのバックアップの保持期間は、最低でも 1 日以上有効にする必要があります。自動バックアップの有効化バックアップの保持期間の設定の詳細については、「バックアップの保存期間」を参照してください。
rds_tlog_copy_setup	rds_tlog_copy_setup ストアドプロシージャの実行中にエラーが発生しました。RDS エンドポイントに再接続して、もう一度試してください。	内部エラーが発生しました。	RDS エンドポイントに再接続し、`rds_tlog_copy_setup` ストアドプロシージャを再実行します。
rds_tlog_copy_setup	rds_tlog_backup_copy_setup ス	ストアドプロシージャでは、`BEGIN` および `END` を使用	`rds_tlog_copy_setup` ストアドプロシージャを実行する

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
	トアドプロシージャをトランザクション内で実行することはサポートされていません。セッションに未処理のトランザクションがないことを確認して、もう一度試してください。	してトランザクション内で試行されました。	ときは、BEGIN および END を使用しないでください。
rds_tlog_copy_setup	入力パラメータ @target_s3_arn の S3 バケット名には、スペース以外の文字が少なくとも 1 文字含まれている必要があります。	入力パラメータ @target_s3_arn に間違った値が指定されました。	入力パラメータ @target_s3_arn に完全な Amazon S3 バケット ARN が指定されていることを確認します。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_copy_setup	SQLSERVER_BACKUP_RESTORE オプションが有効になっていないか、有効化処理中です。オプションを有効にするか、後でもう一度試してください。	SQLSERVER_BACKUP_RESTORE オプションが DB インスタンスで有効になっていないか、有効化されたばかりで内部アクティベーションが保留されています。	要件セクションで指定されている SQLSERVER_BACKUP_RESTORE オプションを有効にします。数分間待って、再度 rds_tlog_copy_setup ストアドプロシージャを実行してください。
rds_tlog_copy_setup	入力パラメータ @target_s3_arn のターゲット S3 arn を空または null にすることはできません。	入力パラメータ @target_s3_arn に NULL 値が指定されたか、値が指定されませんでした。	入力パラメータ @target_s3_arn に完全な Amazon S3 バケット ARN が指定されていることを確認します。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_copy_setup	入力パラメータ @target_s3_arn のターゲット S3 arn は、arn:aws で始まる必要があります。	入力パラメータ @target_s3_arn は、前に arn:aws を付けずに指定されました。	入力パラメータ @target_s3_arn に完全な Amazon S3 バケット ARN が指定されていることを確認します。
rds_tlog_copy_setup	ターゲット S3 ARN は、既に指定された値が設定されています。	rds_tlog_copy_setup ストアドプロシージャは前に実行され、Amazon S3 バケット ARN で設定されていました。	トランザクションログのバックアップにアクセスするために Amazon S3 バケット値を変更するには、別の target S3 ARN を指定します。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_copy_setup	トランザクションログのバックアップへのアクセスを有効にする認証情報を生成できません。rds_tlog_copy_setup で指定されている S3 パス ARN を確認し、後でもう一度試してください。	トランザクションログのバックアップにアクセスするための認証情報の生成中に、不明なエラーが発生しました。	設定設定を確認して、もう一度試してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_copy_setup	保留中のタスクがある間は、rds_tlog_copy_setup ストアドプロシージャを実行できません。保留中のタスクが完了するのを待ち、もう一度試してください。	一度に実行できるタスクは 2 つだけです。完了を待っている保留中のタスクがあります。	保留中のタスクを表示して、完了するのを待ちます。モニタリングタスクのステータスの詳細については、「タスクのステータスの追跡」をご参照ください。
rds_tlog_backup_copy_to_S3	T-log バックアップファイルコピータスクがデータベース: %s、タスク ID: %d で既に発行されています。後でもう一度試してください。	特定のデータベースに対して、同時に実行できるコピータスクは 1 つだけです。完了を待っている保留中のコピータスクがあります。	保留中のタスクを表示して、完了するのを待ちます。モニタリングタスクのステータスの詳細については、「タスクのステータスの追跡」をご参照ください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	これらの 3 セットのパラメータセットのうち、少なくとも 1 セットを指定する必要があります。SET-1:(@backup_file_start_time、@backup_file_end_time) | SET-2:(@starting_lsn、@ending_lsn) | SET-3:(@rds_backup_starting_seq_id、@rds_backup_ending_seq_id)	3 セットのパラメータセットのいずれも指定されていないか、指定されたパラメータセットに必要なパラメータが不足しています。	時間、lsn、シーケンス ID のいずれかのパラメータを指定できます。これら 3 セットのパラメータのうち 1 セットが必要です。必要なパラメータの詳細については、「トランザクションログのバックアップのコピー」を参照してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	インスタンスでは、バックアップは無効になっています。バックアップを有効にして、しばらくしてからもう一度試してください。	DB インスタンスの自動バックアップを有効化する。	自動バックアップの有効化バックアップの保持期間の設定の詳細については、「バックアップの保存期間」を参照してください。
rds_tlog_backup_copy_to_S3	指定されたデータベース %s が見つかりません。	入力パラメータ @db_name に指定された値が DB インスタンスのデータベース名と一致しません。	正しいデータベース名を使用してください。すべてのデータベースを名前で一覧表示するには、SELECT * from sys.databases を実行します。
rds_tlog_backup_copy_to_S3	SQL Server システムデータベースまたは rdsadmin データベースの RDS_TLOG_Backup_copy_to_S3 ストアドプロシージャを実行できません。	入力パラメータ @db_name として指定された値は、SQL Server システムデータベース名または RDSAdmin データベースと一致します。	データベース master, model, msdb, tempdb, RDSAdmin. は、トランザクションログバックアップへのアクセスには使用できません。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	入力パラメータ @db_name のデータベース名を空または null にすることはできません。	入力パラメータ @db_name に空または NULL の値が指定されました。	正しいデータベース名を使用してください。すべてのデータベースを名前で一覧表示するには、SELECT * from sys.databases を実行します。
rds_tlog_backup_copy_to_S3	rds_tlog_backup_copy_setup ストアドプロシージャを実行するには、DB インスタンスのバックアップ保持期間を少なくとも 1 に設定する必要があります。	DB インスタンスの自動バックアップを有効化する。	自動バックアップの有効化バックアップの保持期間の設定の詳細については、「バックアップの保存期間」を参照してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	ストアドプロシージャ rds_tlog_backup_copy_to_S3 の実行中にエラーが発生しました。RDS エンドポイントに再接続して、もう一度試してください。	内部エラーが発生しました。	RDS エンドポイントに再接続し、`rds_tlog_backup_copy_to_S3` ストアドプロシージャを再実行します。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	これらの 3 セットのパラメータセットのうち、1 セットのみ指定できます。SET-1: (@backup_file_start_time、@backup_file_end_time) | SET-2: (@starting_lsn、@ending_lsn) | SET-3:(@rds_backup_starting_seq_id、@rds_backup_ending_seq_id)	複数のパラメータセットが指定されました。	時間、lsn、シーケンス ID のいずれかのパラメータを指定できます。これら 3 セットのパラメータのうち 1 セットが必要です。必要なパラメータの詳細については、「トランザクションログのバックアップのコピー」を参照してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	rds_tlog_backup_copy_to_S3 ストアドプロシージャをトランザクション内で実行することはサポートされていません。セッションに未処理のトランザクションがないことを確認して、もう一度試してください。	ストアドプロシージャでは、BEGIN および END を使用してトランザクション内で試行されました。	`rds_tlog_backup_copy_to_S3` ストアドプロシージャを実行するときは、BEGIN および END を使用しないでください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	指定されたパラメータは、トランザクションバックアップログの保持期間外です。使用可能なトランザクションログのバックアップファイルを一覧表示するには、rds_fn_list_tlog_backup_metadata 関数を実行します。	指定された入力パラメータで、コピー保持期間内で該当するトランザクションのログバックアップはありませんでした。	有効なパラメータセットを使用してもう一度試してください。必要なパラメータの詳細については、「トランザクションログのバックアップのコピー」を参照してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	リクエストの処理中にアクセス許可エラーがありました。バケットが DB インスタンスと同じアカウントとリージョンにあることを確認し、公開ドキュメントのテンプレートに対する S3 バケットポリシーのアクセス許可を確認してください。	指定された S3 バケットまたはそのポリシーのアクセス許可で問題が検出されました。	トランザクションログのバックアップにアクセスするための設定が正しいことを確認します。S3 バケットの設定要件の詳細については、「要件」を参照してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	RDS リードレプリカインスタンスで rds_tlog_backup_copy_to_S3 ストアドプロシージャを実行することは許可されていません。	RDS リードレプリカインスタンスでストアドプロシージャが試行されました。	RDS プライマリ DB インスタンスに接続して、rds_tlog_backup_copy_to_S3 ストアドプロシージャを実行します。
rds_tlog_backup_copy_to_S3	入力パラメータ @starting_lsn の LSN は @ending_lsn 未満である必要があります。	入力パラメータ @starting_lsn に指定された値が、入力パラメータ @ending_lsn に指定された値より大きい。	入力パラメータ @starting_lsn に指定された値が、入力パラメータ @ending_lsn に指定された値より小さいことを確認してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	rds_tlog_backup_copy_to_S3 ストアドプロシージャは、ソースデータベース内の db_owner ロールのメンバーのみが実行できます。	指定された db_name で rds_tlog_backup_copy_to_S3 ストアドプロシージャを実行しようとしているアカウントには、db_owner ロールが付与されていません。	ストアドプロシージャを実行しているアカウントに、指定された db_name の db_owner ロールが付与されていることを確認してください。
rds_tlog_backup_copy_to_S3	入力パラメータ @rds_backup_starting_seq_id のシーケンス ID は、@rds_backup_ending_seq_id 以下である必要があります。	入力パラメータ @rds_backup_starting_seq_id に指定された値が、入力パラメータ @rds_backup_ending_seq_id に指定された値より大きい。	入力パラメータ @rds_backup_starting_seq_id に指定された値が、入力パラメータ @rds_backup_ending_seq_id に指定された値より小さいことを確認してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	SQLSERVER_BACKUP_RESTORE オプションが有効になっていないか、有効化処理中です。オプションを有効にするか、後でもう一度試してください。	SQLSERVER_BACKUP_RESTORE オプションが DB インスタンスで有効になっていないか、有効化されたばかりで内部アクティベーションが保留されています。	要件セクションで指定されている SQLSERVER_BACKUP_RESTORE オプションを有効にします。数分間待って、再度 rds_tlog_backup_copy_to_S3 ストアドプロシージャを実行してください。
rds_tlog_backup_copy_to_S3	入力パラメータ @backup_file_start_time の開始時刻は @backup_file_end_time より早い必要があります。	入力パラメータ @backup_file_start_time に指定された値が、入力パラメータ @backup_file_end_time に指定された値より遅い。	入力パラメータ @backup_file_start_time に指定された値が、入力パラメータ @backup_file_end_time に指定された値より小さいことを確認してください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	アクセスできないため、リクエストを処理できませんでした。機能の設定とアクセス許可を確認してください。	Amazon S3 バケットのアクセス許可に問題があるか、指定された Amazon S3 バケットが別のアカウントまたはリージョンにある可能性があります。	Amazon S3 バケットポリシーのアクセス許可が、RDS のアクセスを許可するように設定されていることを確認してください。Amazon S3 バケットが DB インスタンスと同じアカウントとリージョンにあることを確認してください。
rds_tlog_backup_copy_to_S3	ストレージが暗号化されていないインスタンスのストアドプロシージャへの入力パラメータとして、KMS Key ARN は指定できません。	DB インスタンスでストレージ暗号化が有効化されていない場合は、入力パラメータ `@kms_key_arn` を指定しないでください。	`@kms_key_arn` の入力パラメータは指定しないでください。

ストアドプロシージャ	エラーメッセージ	問題	トラブルシューティングの推奨事項
rds_tlog_backup_copy_to_S3	ストレージ暗号化インスタンスのストアドプロシージャへの入力パラメータとして、KMS キー ARN を指定する必要があります。	DB インスタンスでストレージ暗号化が有効化されている場合は、入力パラメータ `@kms_key_arn` を指定する必要があります。	`@kms_key_arn` の入力パラメータに、トランザクションログのバックアップに使用する Amazon S3 バケットの ARN と一致する値を指定します。
rds_tlog_backup_copy_to_S3	`rds_tlog_backup_copy_to_S3` ストアドプロシージャを実行する前に、`rds_tlog_copy_setup` ストアドプロシージャを実行して、`@target3_arn` を設定する必要があります。	`rds_tlog_backup_copy_to_S3` ストアドプロシージャの実行を試行する前に、トランザクションログのバックアップへのアクセスのセットアップ手順が完了しませんでした。	`rds_tlog_backup_copy_to_S3` ストアドプロシージャを実行する前に、`rds_tlog_copy_setup` ストアドプロシージャを実行してください。トランザクションログのバックアップにアクセスするためのセットアップ手順の実行の詳細については、「トランザクションログのバックアップへのアクセス設定」を参照してください。

Microsoft SQL Server データベースエンジンのオプション

このセクションでは、Microsoft SQL Server DB エンジンを実行する Amazon RDS インスタンスに使用できるオプションについて説明します。これらのオプションを有効にするには、オプショングループに追加してから、そのオプショングループを DB インスタンスに関連付けます。詳細については、「オプショングループを使用する」を参照してください。

RDS オプショングループを介して追加されないオプション機能 (SSL、Microsoft Windows 認証、Amazon S3 統合など) を必要とする場合は、「Amazon RDS での Microsoft SQL Server の追加機能」を参照してください。

Amazon RDS では、Microsoft SQL Server DB インスタンスの以下のオプションがサポートされています。

オプション	オプション ID	エンジンのエディション
Oracle OLEDB とリンクされたサーバー	`OLEDB_ORACLE`	SQL Server Enterprise Edition SQL Server Standard Edition
ネイティブバックアップおよび復元	`SQLSERVER_BACKUP_RESTORE`	SQL Server Enterprise Edition SQL Server Standard Edition SQL Server Web Edition SQL Server Express Edition
透過的なデータ暗号化	`TRANSPARENT_DATA_ENCRYPTION` (RDS コンソール)	SQL Server 2016–2022 Enterprise Edition

オプション	オプション ID	エンジンのエディション
	TDE (AWS CLI と RDS API)	SQL Server 2022 Standard Edition
SQL Server Audit	SQLSERVER_AUDIT	RDS では、SQL Server 2016 以降、SQL Server のすべてのエディションでサーバーレベルの監査がサポートされており、Enterprise Edition でもデータベースレベルの監査がサポートされています。 SQL Server SQL Server 2016 (13.x) SP1 以降では、すべてのエディションでサーバーレベルとデータベースレベルの両方の監査がサポートされています。 詳細については、SQL Server ドキュメントの「SQL Server Audit (database engine)」を参照してください。

オプション	オプション ID	エンジンのエディション
SQL Server Analysis Services	SSAS	SQL Server Enterprise Edition SQL Server Standard Edition
SQL Server Integration Services	SSIS	SQL Server Enterprise Edition SQL Server Standard Edition
SQL Server Reporting Services と Power BI Report Server	SSRS	SQL Server Enterprise Edition SQL Server Standard Edition
Microsoft 分散トランザクションコーディネーター	MSDTC	RDS では、SQL Server 2016 以降、SQL Server のすべてのエディションで分散トランザクションがサポートされています。
SQL Server リソースガバナー	RESOURCE_GOVERNOR	SQL Server Enterprise Edition SQL Server 2022 Developer Edition

SQL Server のバージョンとエディションで使用できるオプションの一覧表示

describe-option-group-options AWS CLI コマンドを使用して、SQL Server のバージョンとエディションで使用可能なオプション、およびそれらのオプションの設定を一覧表示できます。

次の例は、SQL Server 2019 Enterprise Edition のオプションとオプションの設定を示しています。--engine-name オプションは必須です。

```
aws rds describe-option-group-options --engine-name sqlserver-ee --major-engine-version
 15.00
```

出力は次のようになります。

```
{
    "OptionGroupOptions": [
        {
            "Name": "MSDTC",
            "Description": "Microsoft Distributed Transaction Coordinator",
            "EngineName": "sqlserver-ee",
            "MajorEngineVersion": "15.00",
            "MinimumRequiredMinorEngineVersion": "4043.16.v1",
            "PortRequired": true,
            "DefaultPort": 5000,
            "OptionsDependedOn": [],
            "OptionsConflictsWith": [],
            "Persistent": false,
            "Permanent": false,
            "RequiresAutoMinorEngineVersionUpgrade": false,
            "VpcOnly": false,
            "OptionGroupOptionSettings": [
                {
                    "SettingName": "ENABLE_SNA_LU",
                    "SettingDescription": "Enable support for SNA LU protocol",
                    "DefaultValue": "true",
                    "ApplyType": "DYNAMIC",
                    "AllowedValues": "true,false",
                    "IsModifiable": true,
                    "IsRequired": false,
                    "MinimumEngineVersionPerAllowedValue": []
                },
        ...
```

```
        {
            "Name": "TDE",
            "Description": "SQL Server - Transparent Data Encryption",
            "EngineName": "sqlserver-ee",
            "MajorEngineVersion": "15.00",
            "MinimumRequiredMinorEngineVersion": "4043.16.v1",
            "PortRequired": false,
            "OptionsDependedOn": [],
            "OptionsConflictsWith": [],
            "Persistent": true,
            "Permanent": false,
            "RequiresAutoMinorEngineVersionUpgrade": false,
            "VpcOnly": false,
            "OptionGroupOptionSettings": []
        }
    ]
}
```

Amazon RDS for SQL Server での Oracle OLEDB によるリンクされたサーバーのサポート

RDS for SQL Server 上の Oracle Provider for OLEDB とリンクされたサーバーを使用すると、Oracle データベース上の外部データソースにアクセスできます。リモート Oracle データソースからデータを読み取り、RDS for SQL Server DB インスタンスの外部にあるリモート Oracle データベースサーバーに対してコマンドを実行できます。Oracle OLEDB とリンクされたサーバーを使用すると、次のことが可能になります。

- SQL Server 以外のデータソースに直接アクセスする
- データを移動することなく、同じクエリでさまざまな Oracle データソースに対してクエリを実行する
- エンタープライズエコシステム全体のデータソースに対して分散クエリ、更新、コマンド、トランザクションを発行する
- Microsoft ビジネスインテリジェンススイート (SSIS、SSRS、SSAS) 内から Oracle データベースへの接続を統合する
- Oracle データベースから RDS for SQL サーバーに移行

既存または新しい RDS for SQL Server DB インスタンスで、Oracle 用の 1 つ以上のリンクされたサーバーをアクティブ化できます。その後、外部の Oracle データソースを DB インスタンスと統合できます。

目次

- サポート対象のバージョンとリージョン
- 制限と推奨事項
- Oracle とリンクされたサーバーのアクティベーション
  - OLEDB_ORACLE のオプショングループの作成
  - OLEDB_ORACLE オプションのオプショングループへの追加
  - OLEDB_ORACLE バージョンオプションを別のバージョンに変更する
  - オプショングループを DB インスタンスに関連付ける
- OLEDB プロバイダーのプロパティの変更
- OLEDB ドライバープロパティの変更
- Oracle とリンクされたサーバーの非アクティブ化

サポート対象のバージョンとリージョン

RDS for SQL Server は、すべてのリージョンで、SQL Server Standard と Enterprise エディションの次のバージョンについて Oracle OLEDB とリンクしたサーバーをサポートします。

- SQL Server 2025、すべてのバージョン
- SQL Server 2022、すべてのバージョン
- SQL Server 2019、すべてのバージョン
- SQL Server 2017、すべてのバージョン

Oracle OLEDB とリンクされたサーバーは、以下の Oracle Database バージョンでサポートされています。

- Oracle Database 21c、すべてのバージョン
- Oracle Database 19c、すべてのバージョン
- Oracle Database 18c、すべてのバージョン

Oracle OLEDB とリンクされたサーバーは、以下の OLEDB Oracle ドライバーバージョンでサポートされています。

- 21.7
- 21.16

制限と推奨事項

Oracle OLEDB とリンクされたサーバーに適用される次の制約事項および推奨事項に注意してください。

- 各 RDS for SQL Server DB インスタンスのセキュリティグループに適切な TCP ポートを追加して、ネットワークトラフィックを許可します。例えば、EC2 Oracle DB インスタンスと RDS for SQL Server DB インスタンスの間にリンクされたサーバーを設定する場合、EC2 Oracle DB インスタンスの IP アドレスからのトラフィックを許可する必要があります。また、SQL Server がデータベース通信をリッスンするために使用しているポートでのトラフィックを許可する必要があります。セキュリティグループの詳細については、「セキュリティグループによるアクセス制御」を参照してください。

- オプショングループの OLEDB_ORACLE オプションをオン、オフ、または変更した後、RDS for SQL Server DB インスタンスを再起動します。オプショングループのステータスにはこれらのイベントに pending_reboot が表示され、必須です。常時オプションやミラーリングオプションが有効になっている RDS for SQL Server のマルチ AZ インスタンスの場合、インスタンスの新規作成後または復元後にインスタンスを再起動すると、フェイルオーバーが予想されます。
- Oracle データソースのユーザー名とパスワードによる簡易認証のみをサポートします。
- Open Database Connectivity (ODBC) ドライバーはサポートされていません。上記の OLEDB ドライバーバージョンのみがサポートされています。
- 分散トランザクション (XA) はサポートされています。分散トランザクションを有効にするには、DB インスタンスのオプショングループで MSDTC オプションを有効にし、XA トランザクションが有効になっていることを確認します。詳細については、「RDS for SQL Server での Microsoft 分散トランザクションコーディネーターのサポート」を参照してください。
- 接続文字列のショートカットとして使用するデータソース名 (DSN) の作成はサポートされていません。
- OLEDB ドライバーのトレースはサポートされていません。SQL Server 拡張イベントを使用して OLEDB イベントをトレースできます。詳細については、「RDS for SQL Server で拡張イベントを設定する」を参照してください。
- SQL Server Management Studio (SSMS) を使用して Oracle リンクサーバーのカタログフォルダにアクセスすることはサポートされていません。

Oracle とリンクされたサーバーのアクティベーション

RDS for SQL Server DB インスタンスに OLEDB_ORACLE オプションを追加して、Oracle とリンクされたサーバーを有効にします。以下のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. オプショングループに [OLEDB_ORACLE] オプションを追加します。
3. 使用する OLEDB ドライバーのバージョンを選択します。
4. オプショングループを DB インスタンスに関連付けます。
5. DB インスタンスを再起動します。

OLEDB_ORACLE のオプショングループの作成

Oracle とリンクされたサーバーを使用するには、使用する DB インスタンスの SQL Server のエディションとバージョンに対応するオプショングループを作成または変更します。この手順を完了するには、AWS マネジメントコンソール または AWS CLI を使用してください。

コンソール

次の手順では、SQL Server Standard Edition 2019 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create subnet group(オプショングループの作成)] ウィンドウで以下を行います。
   a. [名前] に、AWS アカウント内で一意のオプショングループ名 (**oracle-oledb-se-2019** など) を入力します。名前には、英字、数字、ハイフンのみを使用できます。
   b. [説明] に、オプショングループの簡単な説明 (**OLEDB_ORACLE option group for SQL Server SE 2019** など) を入力します。この説明は表示用に使用されます。
   c. [エンジン] で [sqlserver-se] を選択します。
   d. [Major engine version] (メジャーエンジンのバージョン) で、[15.00] を選択します。
5. [作成] を選択します。

CLI

次の手順では、SQL Server Standard Edition 2019 のオプショングループを作成します。

オプショングループを作成するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-option-group \
```

```
    --option-group-name oracle-oledb-se-2019 \
    --engine-name sqlserver-se \
    --major-engine-version 15.00 \
    --option-group-description "OLEDB_ORACLE option group for SQL Server SE 2019"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --engine-name sqlserver-se ^
    --major-engine-version 15.00 ^
    --option-group-description "OLEDB_ORACLE option group for SQL Server SE 2019"
```

OLEDB_ORACLE オプションのオプショングループへの追加

次に、AWS マネジメントコンソール または AWS CLI を使用して OLEDB_ORACLE オプションをオプショングループに追加します。

コンソール

OLEDB_ORACLE オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 作成したオプショングループ (この例では oracle-oledb-se-2019) を選択します。
4. [オプションの追加] を選択します。
5. [Option details] (オプションの詳細) で、[Option name] (オプション名) として [OLEDB_ORACLE] を選択します。
6. Version で、インストールする OLEDB Oracle ドライバーのバージョンを選択します。
7. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
8. [オプションを追加] を選択します。

CLI

OLEDB_ORACLE オプションを追加するには

- オプショングループに [OLEDB_ORACLE] オプションを追加します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.16 \
    --apply-immediately
```

  Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.16 ^
    --apply-immediately
```

OLEDB_ORACLE バージョンオプションを別のバージョンに変更する

OLEDB_ORACLE オプションバージョンを別のバージョンに変更するには、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

OLEDB_ORACLE オプションを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. OLEDB_ORACLE オプションが含まれているオプショングループ (前の例では oracle-oledb-se-2019) を選択します。
4. [Modify option] (オプションの変更) を選択します。
5. [Option details] (オプションの詳細) で、[Option name] (オプション名) として [OLEDB_ORACLE] を選択します。

6. [バージョン] で、使用する OLEDB Oracle ドライバーのバージョンを選択します。
7. [スケジュール] で、オプションをすぐに変更するか、次のメンテナンスウィンドウで追加するかを選択します。
8. [Modify option] (オプションの変更) を選択します。

CLI

OLEDB_ORACLE オプションバージョンを変更するには、使用するオプショングループおよびオプションバージョンを指定して、rds add-option-to-option-group AWS CLI コマンドを使用します。

OLEDB_ORACLE オプションを変更するには

- Example

  Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.7 \
    --apply-immediately
```

  Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --options OptionName=OLEDB_ORACLE, OptionVersion=21.7 ^
    --apply-immediately
```

オプショングループを DB インスタンスに関連付ける

OLEDB_ORACLE オプショングループおよびパラメータグループを DB インスタンスに関連付けるには、AWS マネジメントコンソールまたは AWS CLI を使用します。

コンソール

Oracle のリンクされたサーバーの有効化を完了するには、OLEDB_ORACLE オプショングループを新規または既存の DB インスタンスに関連付けます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

OLEDB_ORACLE オプショングループおよびパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

OLEDB_ORACLE オプショングループおよびパラメータグループを使用してインスタンスを作成するには

- オプショングループの作成時に使用したのと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mytestsqlserveroracleoledbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 15.0.4236.7.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name oracle-oledb-se-2019 \
    --db-parameter-group-name my-parameter-group-name
```

  Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mytestsqlserveroracleoledbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
```

```
    --engine-version 15.0.4236.7.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name oracle-oledb-se-2019 ^
    --db-parameter-group-name my-parameter-group-name
```

インスタンスを変更して **OLEDB_ORACLE** オプショングループを関連付けるには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mytestsqlserveroracleoledbinstance \
    --option-group-name oracle-oledb-se-2019 \
    --db-parameter-group-name my-parameter-group-name \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mytestsqlserveroracleoledbinstance ^
    --option-group-name oracle-oledb-se-2019 ^
    --db-parameter-group-name my-parameter-group-name ^
    --apply-immediately
```

OLEDB プロバイダーのプロパティの変更

OLEDB プロバイダーのプロパティを表示および変更することができます。master ユーザーのみが、このタスクを実行できます。DB インスタンス上に作成された Oracle のリンクされたサーバーはすべて、その OLEDB プロバイダーの同じプロパティを使用します。sp_MSset_oledb_prop ストアドプロシージャを呼び出して、OLEDB プロバイダーのプロパティを変更します。

OLEDB プロバイダーのプロパティを変更するには

```
USE [master]
GO
EXEC sp_MSset_oledb_prop N'OraOLEDB.Oracle', N'AllowInProcess', 1
EXEC sp_MSset_oledb_prop N'OraOLEDB.Oracle', N'DynamicParameters', 0
GO
```

次のプロパティを変更できます。

プロパティ名	推奨値 (1 = オン、0 = オフ)	Description
Dynamic parameter	1	パラメータ化されたクエリで SQL プレースホルダー ('?' で表されます) を許可します。
Nested queries	1	サブクエリなど、FROM 句内のネストされた SELECT ステートメントを許可します。
Level zero only	0	プロバイダーに対して呼び出されるのは、ベースレベルの OLEDB インターフェイスだけです。
Allow inprocess	1	Microsoft SQL Server を有効にすると、プロバイダーをインプロセスサーバーとしてインスタンス化できます。Oracle リンクサーバーを使用するには、このプロパティを 1 に設定します。
Non transacted updates	0	0 以外の場合、SQL Server は更新を許可します。
Index as access path	誤	0 以外の場合、SQL Server はプロバイダーのインデックスを使用してデータを取得しようとします。
Disallow adhoc access	誤	設定すると、SQL Server は OLEDB プロバイダーに対するパススルークエリの実行を許可しません。このオプションはチェックできますが、パス

プロパティ名	推奨値 (1 = オン、0 = オフ)	Description
		スルークエリを実行するのが適切な場合もあります。
Supports LIKE operator	1	プロバイダーが LIKE キーワードを使用したクエリをサポートしていることを示します。

OLEDB ドライバープロパティの変更

Oracle にリンクされたサーバーを作成するとき、OLEDB ドライバーのプロパティを表示および変更できます。master ユーザーのみが、このタスクを実行できます。[Driver] (ドライバー) プロパティは、リモート Oracle データソースを使用するときに OLEDB ドライバーがデータを処理する方法を定義します。[Driver] (ドライバー) プロパティは、DB インスタンスで作成された各 Oracle リンクサーバーに固有です。master.dbo.sp_addlinkedserver ストアドプロシージャを呼び出して、OLEDB プロバイダーのプロパティを変更します。

例: リンクされたサーバーを作成して OLEDB ドライバー FetchSize プロパティを変更するには

```
EXEC master.dbo.sp_addlinkedserver
@server = N'Oracle_link2',
@srvproduct=N'Oracle',
@provider=N'OraOLEDB.Oracle',
@datasrc=N'my-oracle-test.cnetsipka.us-west-2.rds.amazonaws.com:1521/ORCL',
@provstr='FetchSize=200'
GO
```

```
EXEC master.dbo.sp_addlinkedsrvlogin
@rmtsrvname=N'Oracle_link2',
@useself=N'False',
@locallogin=NULL,
@rmtuser=N'master',
@rmtpassword='Test#1234'
GO
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

Oracle とリンクされたサーバーの非アクティブ化

Oracle でリンクされた Server を無効にするには、オプショングループから `OLEDB_ORACLE` オプションを削除します。

> **Important**
>
> このオプションを削除しても、DB インスタンス上の既存のリンクされたサーバー設定は削除されません。DB インスタンスから削除するには、手動で削除する必要があります。
> 削除後に `OLEDB_ORACLE` オプションを再度有効にすると、DB インスタンスで以前に設定したリンクされたサーバー設定を再利用できます。

コンソール

以下の手順では、`OLEDB_ORACLE` オプションを削除します。

OLEDB_ORACLE オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. `OLEDB_ORACLE` オプションが含まれているオプショングループ (前の例では `oracle-oledb-se-2019`) を選択します。
4. [オプションを削除] を選択します。
5. [Deletion options] (オプションの削除) で、[Options to delete] (削除するオプション) として [OLEDB_ORACLE] を選択します。
6. [Apply immediately] (すぐに適用) で、オプションをすぐに削除する場合は [Yes] (はい) を選択し、次のメンテナンスウィンドウで削除する場合は [No] (いいえ) を選択します。
7. [削除] を選択します。

CLI

以下の手順では、OLEDB_ORACLE オプションを削除します。

OLEDB_ORACLE オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name oracle-oledb-se-2019 \
    --options OLEDB_ORACLE \
    --apply-immediately
```

  Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name oracle-oledb-se-2019 ^
    --options OLEDB_ORACLE ^
    --apply-immediately
```

RDS for SQL Server での Teradata ODBC を使用したリンクサーバー

RDS for SQL Server での Teradata ODBC ドライバーを使用したリンクサーバーのサポートにより、Teradata データベース上の外部データソースにアクセスできます。RDS for SQL Server インスタンスの外部にあるリモート Teradata データベースサーバーからデータを読み取ってコマンドを実行できます。Teradata ODBC を使用したリンクサーバーを使用すると、次の機能が有効になります。

- SQL Server 以外のデータソースに直接アクセスする。
- データを移動することなく、同じクエリでさまざまな Teradata データソースに対してクエリを実行する。
- エンタープライズエコシステム全体のデータソースに対して分散クエリ、更新、コマンド、トランザクションを発行する。
- Microsoft ビジネスインテリジェンススイート (SSIS、SSRS、SSAS) 内から Teradata データベースへの接続を統合する。
- Teradata データベースから RDS for SQL Server に移行する。

既存の RDS for SQL Server DB インスタンスまたは新しい RDS for SQL Server DB インスタンスで、Teradata 用のリンクサーバーを 1 つ以上アクティブ化できます。その後、外部の Teradata データソースを DB インスタンスと統合できます。

トピック

- サポート対象のバージョンとリージョン
- 制限と推奨事項
- マルチ AZ 配置に関する考慮事項
- Teradata とのリンクサーバーのアクティブ化
- Teradata とのリンクサーバーの作成
- Teradata にリンクされたサーバーの非アクティブ化

サポート対象のバージョンとリージョン

RDS for SQL Server は、すべての AWS リージョンで、SQL Server Standard Edition と SQL Server Enterprise Edition の次のバージョンについて Teradata ODBC を使用したリンクサーバーをサポートしています。

- SQL Server 2025、すべてのバージョン
- SQL Server 2022、すべてのバージョン
- SQL Server 2019、すべてのバージョン
- SQL Server 2017、すべてのバージョン

次の Teradata データベースバージョンは、RDS for SQL Server とのリンクをサポートしています。

- Teradata 17.20、すべてのバージョン

制限と推奨事項

Teradata ODBC を使用したリンクサーバーには、次の制限が適用されます。

- RDS for SQL Server は、Teradata ソースのユーザー名とパスワードによる簡易認証のみをサポートします。
- RDS for SQL Server は、Teradata ODBC ドライバーのバージョン 17.20.0.33 のみをサポートします。
- RDS for SQL Server は、接続文字列のショートカットとして使用するデータソース名 (DSN) の作成をサポートしません。
- RDS for SQL Server は、ODBC ドライバーのトレースをサポートしません。ODBC イベントをトレースするには、SQL Server 拡張イベントを使用します。詳細については、「RDS for SQL Server で拡張イベントを設定する」を参照してください。
- RDS for SQL Server は、SQL Server Management Studio (SSMS) の使用時に Teradata リンクサーバーのカタログフォルダにアクセスすることをサポートしていません。

Teradata ODBC を使用したリンクサーバーを使用する際には、次の推奨事項を考慮してください。

- 各 RDS for SQL Server DB インスタンスのセキュリティグループに適切な TCP ポートを追加して、ネットワークトラフィックを許可します。EC2 Teradata DB インスタンスと RDS for SQL Server DB インスタンスの間にリンクサーバーを設定する場合、EC2 Teradata DB インスタンスの IP アドレスからのトラフィックを許可する必要があります。また、RDS for SQL Server DB インスタンスがデータベース通信をリッスンするために使用しているポートでのトラフィックを許可する必要があります。セキュリティグループの詳細については、「セキュリティグループによるアクセス制御」を参照してください。

- 分散トランザクション (XA) はサポートされています。分散トランザクションをアクティブにするには、DB インスタンスのオプショングループで MSDTC オプションを有効にし、XA トランザクションが有効になっていることを確認します。詳細については、「RDS for SQL Server での Microsoft 分散トランザクションコーディネーターのサポート」を参照してください。
- リンクされた Teradata ODBC は、Teradata Server で設定されている限り、SSL/TLS をサポートします。詳細については、「Enable TLS Connectivity on Teradata Vantage」を参照してください。

マルチ AZ 配置に関する考慮事項

RDS for SQL Server は現在、マルチ AZ 配置内のミラーリングされたデータベースサーバー (または Always-On 可用性グループのセカンダリサーバー) にリンクサーバーをレプリケートしません。設定を変更してミラーリングまたは Always-On を追加する前にリンクサーバーを追加した場合、リンクサーバーは既存のリンクサーバーに対してコピーされます。

または、プライマリインスタンスにリンクサーバーを作成し、高可用性サーバーインスタンスにフェイルオーバーしてから、リンクサーバーを再度作成する方法でも、リンクサーバーを両方の RDS for SQL Server インスタンス上に存在させることができます。

Teradata とのリンクサーバーのアクティブ化

RDS for SQL Server DB インスタンスに ODBC_TERADATA オプションを追加して、Teradata とのリンクサーバーをアクティブ化します。以下のプロセスを使用します。

トピック

- ODBC_TERADATA のオプショングループの作成
- ODBC_TERADATA オプションのオプショングループへの追加
- ODBC_TERADATA オプションを DB インスタンスに関連付ける

ODBC_TERADATA のオプショングループの作成

Teradata とのリンクサーバーを使用するには、オプショングループを作成するか、使用する DB インスタンスの SQL Server エディションおよびバージョンに対応するオプショングループを変更します。この手順を完了するには、AWS マネジメントコンソール または AWS CLI を使用してください。

コンソール

次の手順を使用して、SQL Server Standard Edition 2019 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create option group] (オプショングループを作成) ウィンドウで次の操作を行います。
    a. [Name] (名前) に、AWS アカウント 内で一意のオプショングループ名 (`teradata-odbc-se-2019` など) を入力します。名前には、英字、数字、ハイフンのみ使用できます。
    b. [説明] に、オプショングループの簡単な説明を入力します。
    c. [エンジン] で [sqlserver-se] を選択します。
    d. [Major engine version] (メジャーエンジンのバージョン) で、[15.00] を選択します。
5. [作成] を選択します。

AWS CLI

次の手順では、SQL Server Standard Edition 2019 のオプショングループを作成します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name teradata-odbc-se-2019 \
    --engine-name sqlserver-se \
    --major-engine-version 15.00 \
    --option-group-description "ODBC_TERADATA option group for SQL Server SE 2019"
```

Example

Windows の場合:

```
aws rds create-option-group ^
```

```
    --option-group-name teradata-odbc-se-2019 ^
    --engine-name sqlserver-se ^
    --major-engine-version 15.00 ^
    --option-group-description "ODBC_TERADATA option group for SQL Server SE 2019"
```

ODBC_TERADATA オプションのオプショングループへの追加

次に、AWS マネジメントコンソール または AWS CLI を使用して ODBC_Teradata オプションをオプショングループに追加します。

コンソール

次の手順を使用して、SQL Server Standard Edition 2019 のオプショングループを作成します。

ODBC_TERADATA オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 新しいオプショングループを選択します。
4. [オプションを追加] を選択します。
5. [オプションの詳細] で以下を実行します。
   a. [オプション名] で ODBC_TERADATA を選択します。
   b. オプションのバージョンとして 17.20.33.00 を選択します。
6. [スケジューリング] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
7. [オプションを追加] を選択します。

AWS CLI

次の手順では、オプショングループに ODBC_TERADATA オプションを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name teradata-odbc-se-2019 \
```

```
    --options "OptionName=ODBC_TERADATA,OptionVersion=17.20.33.00" \
    --apply-immediately
```

Example

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name teradata-odbc-se-2019 ^
    --options "OptionName=ODBC_TERADATA,OptionVersion=17.20.33.00" ^
    --apply-immediately
```

ODBC_TERADATA オプションを DB インスタンスに関連付ける

ODBC_TERADATA オプショングループを DB インスタンスに関連付けるには、AWS マネジメントコンソールまたは AWS CLI を使用します。

コンソール

Teradata 用のリンクサーバーのアクティブ化を完了するには、オプショングループを新規または既存の DB インスタンスに関連付けます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

オプショングループの作成時に使用したのと同じ DB エンジンのタイプとメジャーバージョンを指定します。

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mytestsqlserverteradataodbcinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 15.00 \
    --license-model license-included \
```

```
    --allocated-storage 100 \
    --master-username admin \
    --master-user-password password \
    --storage-type gp2 \
    --option-group-name teradata-odbc-se-2019
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mytestsqlserverteradataodbcinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 15.00 ^
    --license-model license-included ^
    --allocated-storage 100 ^
    --master-username admin ^
    --master-user-password password ^
    --storage-type gp2 ^
    --option-group-name teradata-odbc-se-2019
```

インスタンスを変更して新しいオプショングループを関連付けるには:

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mytestsqlserverteradataodbcinstance \
    --option-group-name teradata-odbc-se-2019 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mytestsqlserverteradataodbcinstance ^
    --option-group-name teradata-odbc-se-2019 ^
    --apply-immediately
```

Teradata とのリンクサーバーの作成

Teradata とのリンクサーバーを作成するには、次のコマンドを実行します。

```
EXECUTE master.dbo.sp_addlinkedserver
```

```
    @server = N'LinkedServer_NAME',
    @srvproduct=N'',
    @provider=N'MSDASQL',
    @provstr=N'"PROVIDER=MSDASQL;DRIVER={Teradata Database ODBC Driver 17.20};
                DBCName=Server;UID=user_name;PWD=user_password;
                UseDataEncryption=YES/NO;SSLMODE=PREFER/ALLOW/DISABLE>;"',
    @catalog='database'
```

```
EXECUTE master.dbo.sp_addlinkedsrvlogin
    @rmtsrvname = N'LinkedServer_NAME',
    @locallogin = NULL ,
    @useself = N'False',
    @rmtuser = N'user_name',
    @rmtpassword = N'user_password'
```

上のコマンドの例を次に示します。

```
EXECUTE master.dbo.sp_addlinkedserver
    @server = N'LinkedServerToTeradata',
    @srvproduct=N'',
    @provider=N'MSDASQL',
    @provstr=N'"PROVIDER=MSDASQL;DRIVER={Teradata Database ODBC Driver 17.20};
                DBCName=my-teradata-test.cnetsipka.us-west-2.rds.amazonaws.com;
                UID=master;
                PWD=Test#1234;
                UseDataEncryption=YES;
                SSLMODE=PREFER;"',
    @catalog='MyTestTeradataDB'

EXECUTE master.dbo.sp_addlinkedsrvlogin
    @rmtsrvname = N'LinkedServerToTeradata',
    @locallogin = NULL ,
    @useself = N'False',
    @rmtuser = N'master',
    @rmtpassword = N'Test#1234'
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

Teradata にリンクされたサーバーの非アクティブ化

Teradata へのリンクサーバーを非アクティブにするには、オプショングループから ODBC_TERADATA オプションを削除します。

> **Important**
>
> このオプションを削除しても、DB インスタンス上のリンクサーバー設定は削除されません。DB インスタンスから削除するには、手動で削除する必要があります。
> 削除後に ODBC_TERADATA を再度アクティブにすると、DB インスタンスで以前に設定したリンクサーバー設定を再利用できます。

コンソール

オプショングループから ODBC_TERADATA オプションを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. ODBC_TERADATA オプションを含むオプショングループを選択します。
4. [削除] を選択します。
5. [削除オプション] で、[削除するオプション] の下の ODBC_TERADATA を選択します。
6. [Apply immediately] (すぐに適用) で、オプションをすぐに削除する場合は [Yes] (はい) を選択し、次のメンテナンスウィンドウで削除する場合は [No] (いいえ) を選択します。
7. [削除] を選択します。

AWS CLI

以下のコマンドは、ODBC_TERADATA オプションを削除します。

Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name teradata-odbc-se-2019 \
    --options ODBC_TERADATA \
    --apply-immediately
```

Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name teradata-odbc-se-2019 ^
    --options ODBC_TERADATA ^
    --apply-immediately
```

SQL Server のネイティブバックアップおよび復元のサポート

SQL Server データベースのネイティブバックアップおよび復元を使用すると、オンプレミスデータベースの差分バックアップまたは完全バックアップを作成し、バックアップファイルを Amazon S3 に保存できます。これで、SQL Server を実行する既存の Amazon RDS DB インスタンスに復元できます。また、RDS for SQL Server データベースをバックアップして Amazon S3 に保存し、他の場所に復元することもできます。さらに、オンプレミスサーバーや SQL サーバーが実行されている別の Amazon RDS DB インスタンスにバックアップを復元できます。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。

Amazon RDS では、差分および完全バックアップファイル (.bak ファイル) を使用した、Microsoft SQL Server データベースのネイティブバックアップと復元をサポートしています。

ネイティブバックアップおよび復元オプションの追加

DB インスタンスにネイティブバックアップおよび復元オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [SQLSERVER_BACKUP_RESTORE] オプションを追加します。
3. AWS Identity and Access Management (IAM) ロールとオプションを関連付けます。IAM ロールには、データベースバックアップを復元できるように S3 バケットのアクセス権限を付与します。

   つまり、オプションには、有効な Amazon リソースネーム (ARN) を`arn:aws:iam::`*`account-id`*`:role/`*`role-name`* 形式で設定するオプションが必要となります。詳細については、AWS 全般のリファレンス の「Amazon リソースネーム (ARN)」を参照してください。

   IAM ロールには、信頼関係と許可ポリシーがアタッチされている必要もあります。信頼関係は RDS がロールを引き受けることを許可し、許可ポリシーはロールが実行できるアクションを定義します。詳細については、「ネイティブバックアップおよび復元用の IAM ロールの手動作成」を参照してください。
4. オプショングループを DB インスタンスに関連付けます。

ネイティブバックアップおよび復元オプションを追加したら、DB インスタンスを再起動する必要はありません。オプショングループがアクティブになるとすぐ、バックアップと復元をスタートできます。

コンソール

ネイティブバックアップおよび復元オプションを追加

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 新しいオプショングループを作成するか、既存のオプショングループを使用します。カスタム DB オプションの作成方法については、「オプショングループを作成する」を参照してください。

   既存のオプショングループを使用する場合は、次のステップに進んでください。
4. オプショングループに [SQLSERVER_BACKUP_RESTORE] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
5. 次のいずれかを行ってください。

   - 既存の IAM ロールおよびAmazon S3 設定を使用するには、IAM ロールに対して既存の IAM ロールを選択します。既存の IAM ロールを使用すると、RDS は、このロールに対して設定した Amazon S3 設定を使用します。
   - 新規ロールを作成し、Amazon S3 設定を構成するには、次の手順を実行します。
     1. [IAM role] (IAM ロール) は、[Create a new role] (新しいロールの作成) を選択します。
     2. 「S3 バケット」 で、リストからS3 バケットを選択します。
     3. 「S3 プレフィックス (オプション)」 で、Amazon S3 バケットに保存されているファイルに使用するプレフィックスを指定します。

        このプレフィックスにはファイルパスを含めることはできますが、必須ではありません。プレフィックスを提供すると、RDS はそのプレフィックスをすべてのバックアップファイルにアタッチします。その後、復元中に、RDS はプレフィックスを使用して関連ファイルを特定し、非関連ファイルを無視します。例えば、バックアップファイルを保持する以外の目的で、S3 バケットを使用することが可能です。この場合、RDS が、特定のフォルダやサブフォルダでネイティブバックアップおよび復元を実行するため、プレフィックスを使うことが可能です。

        プレフィックスを空白にした場合、RDS はプレフィックスを使用しないでバックアップファイルの特定およびファイルの復元を実行します。その結果、複数ファイルの復元中に、RDS は S3 バケットの各フォルダのすべてのファイルの復元を試みます。

4. バックアップファイルを暗号化する場合は、「暗号化を有効化する」の チェックボックスを選択します。バックアップファイルを暗号化しない場合は、チェックボックスをオフにします (デフォルト)。

   「暗号化を有効化する」 を選択した場合、「AWS KMS key」で暗号化キーを選択します。暗号化キーの詳細については、AWS Key Management Service デベロッパーガイドの「スタート方法」を参照してください。

6. [オプションの追加] を選択します。
7. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。(詳しくは、「Amazon RDS DB インスタンスを変更する」を参照してください。)

CLI

この手順では、以下を前提とします。

- 既存のオプショングループに SQLSERVER_BACKUP_RESTORE オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
- このオプションを既存の IAM ロールに関連付けます。また、バックアップを保存するための S3 バケットへのアクセス権があります。
- オプショングループを既存の DB インスタンスに適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

ネイティブバックアップおよび復元オプションを追加

1. オプショングループに [SQLSERVER_BACKUP_RESTORE] オプションを追加します。

   Example

   Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
```

```
 --apply-immediately \
 --option-group-name mybackupgroup \
 --options "OptionName=SQLSERVER_BACKUP_RESTORE, \
   OptionSettings=[{Name=IAM_ROLE_ARN,Value=arn:aws:iam::account-id:role/role-
name}]"
```

Windows の場合:

```
aws rds add-option-to-option-group ^
 --option-group-name mybackupgroup ^
 --options "[{\"OptionName\": \"SQLSERVER_BACKUP_RESTORE\", ^
 \"OptionSettings\": [{\"Name\": \"IAM_ROLE_ARN\", ^
 \"Value\": \"arn:aws:iam::account-id:role/role-name"}]}]" ^
 --apply-immediately
```

> **Note**
>
> Windows コマンドプロンプトを使用する場合、JSON コードでは、二重引用符 (") の前にバックスラッシュ (\) を付けてエスケープする必要があります。

2. DB インスタンスにオプショングループを適用します。

   Example

   Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
 --db-instance-identifier mydbinstance \
 --option-group-name mybackupgroup \
 --apply-immediately
```

   Windows の場合:

```
aws rds modify-db-instance ^
 --db-instance-identifier mydbinstance ^
 --option-group-name mybackupgroup ^
 --apply-immediately
```

ネイティブバックアップおよび復元オプションの設定の変更

ネイティブバックアップおよび復元オプションを有効にすると、オプションの設定を変更できます。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。

ネイティブバックアップおよび復元オプションの削除

DB インスタンスからオプションを削除することによって、ネイティブバックアップおよび復元をオフにすることができます。ネイティブバックアップおよび復元オプションを削除したら、DB インスタンスを再起動する必要はありません。

DB インスタンスからネイティブバックアップおよび復元オプションを削除するには、次のいずれかを実行します。

- オプションを所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、ネイティブバックアップおよび復元オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

SQL サーバーの透過的なデータの暗号化サポート

Amazon RDS は、透過的なデータ暗号化 (TDE) を使用して、Microsoft SQL Server を実行する DB インスタンスのデータの暗号化をサポートします。TDE は、ストレージへの書き込み前に自動的にデータを暗号化し、ストレージからのデータの読み取り時に自動的にデータを復号します。

Amazon RDS は、次の SQL Server のバージョンおよびエディションの TDE をサポートしています。

- SQL Server 2025: Standard および Enterprise Edition
- SQL Server 2022: Standard および Enterprise Edition
- SQL Server 2019: Standard および Enterprise Edition
- SQL Server 2017 Enterprise Edition
- SQL Server 2016 Enterprise Edition

> **Note**
>
> RDS for SQL Server は、読み取り専用データベースの TDE をサポートしていません。

SQL Server の 透過的データ暗号化では、2 階層キーアーキテクチャを使用して暗号化キーの管理を行っています。証明書は、データベースマスターキーから生成され、データ暗号化キーの保護に使用されます。データベース暗号化キーにより、ユーザーデータベースのデータの実際の暗号化と復号が実行されます。データベースマスターキーと TDE 証明書は、Amazon RDS によりバックアップおよび管理されます。

透過的データ暗号化 (TDE) は、機密データの暗号化が必要なシナリオで使用されます。例えば、データファイルとバックアップをサードパーティーに提供したり、セキュリティ関連の規制遵守の問題に対処したりすることができます。`model` データベースや `master` データベースなど、SQL Server のシステムデータベースを暗号化することはできません。

透過的データ暗号化の詳細はこのガイドの範囲外ですが、暗号化アルゴリズムとキーのそれぞれのセキュリティ上の長所と短所を理解しておく必要があります。SQL Server の透過的データ暗号化の詳細については、Microsoft のドキュメントで「透過的データ暗号化 (TDE)」を参照してください。

トピック

- RDS for SQL Server の TDE をオンにする

- RDS for SQL Server でのデータの暗号化
- RDS for SQL Server での TDE 証明書のバックアップと復元
- オンプレミスデータベースの TDE 証明書のバックアップと復元
- RDS for SQL Server の TDE をオフにする

RDS for SQL Server の TDE をオンにする

RDS for SQL Server DB インスタンスに対して透過的なデータ暗号化 (TDE) をオンにするには、DB インスタンスに関連付けられている RDS オプショングループで TDE オプションを指定します。

1. DB インスタンスが、TDE オプションが含まれているオプショングループにすでに関連付けられているかどうかを確認します。DB インスタンスが関連付けられているオプショングループを表示するには、RDS コンソール、AWS CLI コマンド (describe-db-instance)、または API オペレーション DescribeDBInstances を使用します。
2. TDE がオンになっているオプショングループに DB インスタンスが関連付けられていない場合は、2 つのオプションから選択できます。オプショングループを作成して TDE オプションを追加するか、オプションを追加するように関連するオプショングループを変更することもできます。

   > **Note**
   >
   > RDS コンソールの場合、このオプション名は `TRANSPARENT_DATA_ENCRYPTION` です。AWS CLI と RDS API の場合、名前は `TDE` です。

   オプショングループの作成または変更の詳細については、「オプショングループを使用する」を参照してください。オプショングループへのオプションの追加の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. [TDE] オプションを持つオプショングループに DB インスタンスを関連付けます。オプショングループへの DB インスタンスの関連付けの詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

オプショングループに関する考慮事項

TDE オプションは、永続オプションです。すべての DB インスタンスおよびバックアップがオプショングループに関連付けられていない限り、オプショングループから削除することはできません。オプショングループに TDE オプションを追加したら、そのオプショングループは、TDE を使用する

DB インスタンスにのみ関連付けることができます。オプショングループの永続オプションの詳細については、「オプショングループの概要」を参照してください。

TDE オプションは永続オプションであるため、オプショングループおよび関連付けられている DB インスタンスとの間に競合が生じることがあります。次の状況で競合が生じることがあります。

- TDE オプションを含む現在のオプショングループを、TDE オプションを含まないオプショングループに置き換えた。
- DB スナップショットから復元した先の新しい DB インスタンスに TDE オプションを含むオプショングループがない。このシナリオの詳細については、「オプショングループの考慮事項」を参照してください。

SQL Server のパフォーマンスに関する考慮事項

透過的データ暗号化の使用は、SQL Server DB インスタンスのパフォーマンスに影響を与えることがあります。

暗号化されていないデータベースが DB インスタンスにあり、そのインスタンスに暗号化されたデータベースが 1 つでもあれば、暗号化されていないデータベースのパフォーマンスも低下することがあります。したがって、暗号化されたデータベースと暗号化されていないデータベースは別々の DB インスタンスに維持することをお勧めします。

RDS for SQL Server でのデータの暗号化

TDE オプションがオプショングループに追加されると、暗号化プロセスに使用される証明書が Amazon RDS によって生成されます。その後、証明書を使用して、DB インスタンス上のデータベースのデータを暗号化する SQL ステートメントを実行できます。

以下の例では、RDS によって生成された `RDSTDECertificateName` という証明書を使用して、`myDatabase` というデータベースを暗号化しています。

```
---------- Turning on TDE -------------

-- Find an RDS TDE certificate to use
USE [master]
GO
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'
GO
```

```
USE [myDatabase]
GO
-- Create a database encryption key (DEK) using one of the certificates from the
 previous step
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE [RDSTDECertificateName]
GO

-- Turn on encryption for the database
ALTER DATABASE [myDatabase] SET ENCRYPTION ON
GO

-- Verify that the database is encrypted
USE [master]
GO
SELECT name FROM sys.databases WHERE is_encrypted = 1
GO
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO
```

TDE を使用した SQL Server データベースの暗号化にかかる時間は、いくつかの要因によって異なります。例えば、DB インスタンスのサイズ、プロビジョンド IOPS ストレージがインスタンスに対して有効になっているかどうか、データ量などです。

RDS for SQL Server での TDE 証明書のバックアップと復元

RDS for SQL Server には、TDE 証明書のバックアップ、復元、および削除のためのストアドプロシージャが用意されています。RDS for SQL Server には、復元されたユーザー TDE 証明書を表示するための機能も用意されています。

ユーザー TDE 証明書は、オンプレミスで TDE がオンになっている RDS for SQL Server にデータベースを復元するために使用されます。これらの証明書には、プレフィックス `UserTDECertificate_` が付いています。データベースを復元した後、それらを使用できるようにする前に、RDS は、TDE を オンにしたデータベースを変更し、RDS で生成された TDE 証明書を使用するようにします。これらの証明書には、プレフィックス `RDSTDECertificate` が付いています。

ユーザー TDE 証明書は、`rds_drop_tde_certificate` ストアドプロシージャを使って削除しない限り、RDS for SQL Server DB インスタンスに残ります。(詳しくは、「復元された TDE 証明書の削除」を参照してください。)

ユーザー TDE 証明書を使用して、移行元 DB インスタンスから他のデータベースを復元できます。復元するデータベースは同じ TDE 証明書を使用し、TDE がオンになっている必要があります。同じ証明書を再度インポート (復元) する必要はありません。

トピック

- 前提条件
- 制限事項
- TDE 証明書のバックアップ
- TDE 証明書の復元
- 復元された TDE 証明書の表示
- 復元された TDE 証明書の削除

前提条件

RDS for SQL Server で TDE 証明書をバックアップまたは復元する前に、次のタスクを実行してください。最初の 3 つについては、「ネイティブバックアップおよび復元のセットアップ」を参照してください。

1. バックアップおよび復元するファイルを保存するための Amazon S3 汎用バケットまたはディレクトリバケットを作成します。

   データベースバックアップと TDE 証明書のバックアップには、別々のバケットを使用することをお勧めします。
2. ファイルのバックアップと復元用の IAM ロールを作成します。

   IAM ロールは、AWS KMS key のユーザーおよび管理者の両方である必要があります。

   ディレクトリバケットを使用する場合、ディレクトリバケットで ネイティブバックアップおよび復元用の IAM ロールの手動作成 に必要なアクセス許可以外、追加のアクセス許可は必要ありません。

   S3 リソースを使用する場合、ネイティブバックアップおよび復元用の IAM ロールの手動作成 に必要なアクセス許可に加えて、IAM ロールには次のアクセス許可も必要です。

   - S3 バケットリソースの `s3:GetBucketAcl`、`s3:GetBucketLocation`、および`s3:ListBucket`
3. DB インスタンスのオプショングループに追加された `SQLSERVER_BACKUP_RESTORE` オプション。

これは、TRANSPARENT_DATA_ENCRYPTION (TDE) オプションへの追加です。

4. 対称暗号化 KMS キーであることを確認します。次のオプションがあります。
   - アカウントに既存の KMS キーがある場合は、それを使用できます。これ以上の操作は不要です。
   - アカウントに既存の対称暗号化 KMS キーがない場合は、AWS Key Management Serviceデベロッパーガイドの「キーの作成」の手順に従って KMS キーを作成します。
5. Amazon S3 統合を有効にして、DB インスタンスと Amazon S3 の間でファイルを転送します。

   Amazon S3 統合を有効にするための詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。

   ディレクトリバケットは S3 統合ではサポートされていないことにご注意ください。このステップは、オンプレミスデータベースの TDE 証明書のバックアップと復元 にのみ必要です。

制限事項

ストアドプロシージャを使用して TDE 証明書をバックアップおよび復元する場合、次の制限があります。

- SQLSERVER_BACKUP_RESTORE および TRANSPARENT_DATA_ENCRYPTION (TDE) オプションはどちらも DB インスタンスに関連付けられたオプショングループに追加されている必要があります。
- TDE 証明書のバックアップおよび復元タスクのキャンセルはサポートされていません。
- RDS for SQL Server DB インスタンス上の他のデータベースの TDE 暗号化にユーザー TDE 証明書を使用することはできません。これを使用して復元できるのは、TDE がオンになっていて、同じ TDE 証明書を使用する移行元 DB インスタンスから他のデータベースのみです。
- 削除できるのはユーザー TDE 証明書のみです。
- RDS でサポートされているユーザー TDE 証明書の最大数は 10 です。数が 10 を超える場合は、未使用の TDE 証明書を削除して、もう一度試してください。
- 証明書名を空または null にすることはできません。
- 証明書を復元する場合、証明書名にキーワード RDSTDECERTIFICATE を含めることはできません。また、プレフィックス UserTDECertificate_ で始まる必要があります。
- @certificate_name パラメータには、a ~ z、0 ~ 9、@、$、#、下線 (_) の文字のみを含めることができます。

- @certificate_file_s3_arn のファイル拡張子は .cer (大文字小文字を区別しない) にする必要があります。
- @private_key_file_s3_arn のファイル拡張子は .pvk (大文字小文字を区別しない) にする必要があります。
- プライベートキーファイルの S3 メタデータには、x-amz-meta-rds-tde-pwd タグが含まれる必要があります。詳細については、「オンプレミスデータベースの TDE 証明書のバックアップと復元」を参照してください。
- RDS for SQL Server は、TDE のクロスアカウントキーの使用をサポートしていません。

TDE 証明書のバックアップ

TDE 証明書をバックアップするには、rds_backup_tde_certificate ストアドプロシージャを使用します。これには、以下の構文があります。

```
EXECUTE msdb.dbo.rds_backup_tde_certificate
    @certificate_name='UserTDECertificate_certificate_name |
 RDSTDECertificatetimestamp',
    @certificate_file_s3_arn='arn:aws:s3:::bucket_name/certificate_file_name.cer',
    @private_key_file_s3_arn='arn:aws:s3:::bucket_name/key_file_name.pvk',
    @kms_password_key_arn='arn:aws:kms:region:account-id:key/key-id',
    [@overwrite_s3_files=0|1];
```

以下のパラメータは必須です。

- @certificate_name — バックアップする TDE 証明書の名前。
- @certificate_file_s3_arn — Amazon S3 の証明書バックアップファイルの送信先 Amazon リソースネーム (ARN)。
- @private_key_file_s3_arn — TDE 証明書を保護するぷらいべーとキーファイルの送信先 S3 ARN。
- @kms_password_key_arn — プライベートキーのパスワードの暗号化に使用される対称 KMS キーの ARN。

次のパラメータはオプションです。

- @overwrite_s3_files — S3 内の既存の証明書および秘密キーファイルを上書きするかどうかを示します。

- `0` – 既存のファイルを上書きしません。この値はデフォルト値です。

  設定 `@overwrite_s3_files` を 0 にすると、ファイルが既に存在している場合はエラーが返されます。
- `1` – バックアップファイルではない場合でも、指定された名前を持つ既存のファイルを上書きします。

Example TDE 証明書のバックアップ

```
EXECUTE msdb.dbo.rds_backup_tde_certificate
    @certificate_name='RDSTDECertificate20211115T185333',
    @certificate_file_s3_arn='arn:aws:s3:::TDE_certs/mycertfile.cer',
    @private_key_file_s3_arn='arn:aws:s3:::TDE_certs/mykeyfile.pvk',
    @kms_password_key_arn='arn:aws:kms:us-
west-2:123456789012:key/AKIAIOSFODNN7EXAMPLE',
    @overwrite_s3_files=1;
```

TDE 証明書の復元

ユーザー TDE 証明書を復元 (インポート) するには `rds_restore_tde_certificate` ストアドプロシージャを使用します。これには、以下の構文があります。

```
EXECUTE msdb.dbo.rds_restore_tde_certificate
    @certificate_name='UserTDECertificate_certificate_name',
    @certificate_file_s3_arn='arn:aws:s3:::bucket_name/certificate_file_name.cer',
    @private_key_file_s3_arn='arn:aws:s3:::bucket_name/key_file_name.pvk',
    @kms_password_key_arn='arn:aws:kms:region:account-id:key/key-id';
```

以下のパラメータは必須です。

- `@certificate_name` — 復元する TDE 証明書の名前。名前はプレフィックス `UserTDECertificate_` で開始する必要があります。
- `@certificate_file_s3_arn` — TDE 証明書を復元するために使用されるバックアップファイルの S3 ARN。
- `@private_key_file_s3_arn` — 復元する TDE 証明書のプライベートキーバックアップファイルの S3 ARN。
- `@kms_password_key_arn` — プライベートキーのパスワードの暗号化に使用される対称 KMS キーの ARN。

Example TDE 証明書の復元

```
EXECUTE msdb.dbo.rds_restore_tde_certificate
    @certificate_name='UserTDECertificate_myTDEcertificate',
    @certificate_file_s3_arn='arn:aws:s3:::TDE_certs/mycertfile.cer',
    @private_key_file_s3_arn='arn:aws:s3:::TDE_certs/mykeyfile.pvk',
    @kms_password_key_arn='arn:aws:kms:us-
west-2:123456789012:key/AKIAIOSFODNN7EXAMPLE';
```

復元された TDE 証明書の表示

復元 (インポート) したユーザー TDE 証明書を表示するには rds_fn_list_user_tde_certificates 関数を使用します。これには、以下の構文があります。

```
SELECT * FROM msdb.dbo.rds_fn_list_user_tde_certificates();
```

出力は以下のようになります。すべての列がここに表示されるわけではありません。

name	certif te_id	princi _id	pvt_ke ncrypt _type_ c	issuer me	cert_s al_num	thumbp t	subjec	start_ e	expiry te	pvt_key_l ast_backu p_date
UserTD rtific _tde_c	343	1	ENCRYP _BY_MA R_KEY	AnyCom y Shippi	79 3e 57 a3 69 fd 1d 9e 47 2c 32 67 1d 9c	0x6BB2 341103 80B FE1BA2 C69509 5B5	AnyCom y Shippi	2022-0 5 19:49: 000000	2023-0 5 19:49: 000000	NULL

					ca af					

復元された TDE 証明書の削除

使用していない復元された (インポートされた) ユーザー TDE 証明書を削除するには、rds_drop_tde_certificate ストアドプロシージャを使用します。これには、以下の構文があります。

```
EXECUTE msdb.dbo.rds_drop_tde_certificate
 @certificate_name='UserTDECertificate_certificate_name';
```

以下のパラメータは必須です。

- @certificate_name— 削除する TDE 証明書の名前。

復元された (インポートされた) TDE 証明書のみを削除できます。RDS で作成された証明書は削除できません。

Example TDE 証明書の削除

```
EXECUTE msdb.dbo.rds_drop_tde_certificate
 @certificate_name='UserTDECertificate_myTDEcertificate';
```

オンプレミスデータベースの TDE 証明書のバックアップと復元

オンプレミスデータベースの TDE 証明書をバックアップし、後でそれらを RDS for SQL Server に復元できます。RDS for SQL Server TDE 証明書をオンプレミス DB インスタンスに復元することもできます。

> **Note**
>
> RDS for SQL Server は、TDE のクロスアカウントキーの使用をサポートしていません。

次の手順では、TDE 証明書とプライベートキーをバックアップします。プライベートキーは、対称暗号化 KMS キーから生成されたデータキーを使用して暗号化されます。

オンプレミスの TDE 証明書をバックアップするには

1. AWS CLI generate-data-key コマンドを使用して、データキーを生成します。

```
aws kms generate-data-key \
    --key-id my_KMS_key_ID \
    --key-spec AES_256
```

出力は以下のようになります。

```
{
"CiphertextBlob": "AQIDAHimL2NEoAlOY6Bn7LJfnxi/OZe9kTQo/
XQXduug1rmerwGiL7g5ux4av9GfZLxYTDATAAAAfjB8BgkqhkiG9w0B
BwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMyCxLMi7GRZgKqD65AgEQgDtjvZLJo2cQ31Vetng
2RezQy3sAS6ZHrCjfnfn0c65bFdhsXxjSMnudIY7AKw==",
"Plaintext": "U/fpGtmzGCYBi8A2+0/9qcRQRK2zmG/aOn939ZnKi/0=",
"KeyId": "arn:aws:kms:us-west-2:123456789012:key/1234abcd-00ee-99ff-88dd-
aa11bb22cc33"
}
```

次のステップで、プレーンテキスト出力をプライベートキーのパスワードとして使用します。

2. 次の例に示すように、TDE 証明書をバックアップします。

```
BACKUP CERTIFICATE myOnPremTDEcertificate TO FILE = 'D:\tde-cert-backup.cer'
WITH PRIVATE KEY (
FILE = 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\cert-
backup-key.pvk',
ENCRYPTION BY PASSWORD = 'U/fpGtmzGCYBi8A2+0/9qcRQRK2zmG/aOn939ZnKi/0=');
```

3. 証明書のバックアップファイルを Amazon S3 証明書バケットに保存します。
4. プライベートキーのバックアップファイルを S3 証明書バケットに保存し、ファイルのメタデータに次のタグを付けます。

   - キー – x-amz-meta-rds-tde-pwd
   - 値 — データキーの生成による CiphertextBlob 値、以下の例を参照。

```
AQIDAHimL2NEoAlOY6Bn7LJfnxi/OZe9kTQo/
XQXduug1rmerwGiL7g5ux4av9GfZLxYTDATAAAAfjB8BgkqhkiG9w0B
BwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMyCxLMi7GRZgKqD65AgEQgDtjvZLJo2cQ31Vet
2RezQy3sAS6ZHrCjfnfn0c65bFdhsXxjSMnudIY7AKw==
```

次の手順では、RDS for SQL Server TDE 証明書をオンプレミス DB インスタンスに復元します。証明書のバックアップ、対応するプライベートキーファイル、およびデータキーを使用して、移行先 DB インスタンスに TDE 証明書をコピーして復元します。復元された証明書は、新しいサーバーのデータベースマスターキーによって暗号化されます。

TDE 証明書を復元するには

1. TDE 証明書のバックアップファイルとプライベートキーファイルを Amazon S3 から移行先インスタンスにコピーします。Amazon S3 からのファイルコピーの詳細については、「RDS for SQL Server と Amazon S3 間のファイル転送」を参照してください。
2. KMS キーを使用して出力暗号テキストを復号化し、データキーのプレーンテキストを取得します。暗号テキストは、プライベートキーバックアップファイルの S3 メタデータにあります。

   ```
   aws kms decrypt \
       --key-id my_KMS_key_ID \
       --ciphertext-blob fileb://exampleCiphertextFile | base64 -d \
       --output text \
       --query Plaintext
   ```

   次のステップで、プレーンテキスト出力をプライベートキーのパスワードとして使用します。
3. 次の SQL コマンドを使用して、TDE 証明書を復元します。

   ```
   CREATE CERTIFICATE myOnPremTDEcertificate FROM FILE='D:\tde-cert-backup.cer'
   WITH PRIVATE KEY (FILE = N'D:\tde-cert-key.pvk',
   DECRYPTION BY PASSWORD = 'plain_text_output');
   ```

KMS の復号化の詳細については、「AWS CLI コマンドリファレンス」の KMS セクションの「復号」を参照してください。

TDE 証明書が移行先 DB インスタンスで復元された後、その証明書を使用して暗号化されたデータベースを復元できます。

> **Note**
>
> 同じ TDE 証明書を使用して、移行元 DB インスタンス上の複数の SQL Server データベースを暗号化できます。複数のデータベースを移行先インスタンスに移行するには、それらに関連付けられた TDE 証明書を移行先インスタンスに一度だけコピーします。

RDS for SQL Server の TDE をオフにする

RDS for SQL Server DB インスタンスの TDE をオフにするには、まず、DB インスタンスに暗号化されたオブジェクトが残っていないようにします。これを行うには、オブジェクトを復号化するか、削除します。暗号化されたオブジェクトが DB インスタンスに残っている場合は、DB インスタンスに対して TDE をオフにすることはできません。暗号化用のユーザー TDE 証明書が復元 (インポート) された場合は、削除する必要があります。コンソールを使用してオプショングループから TDE オプションを削除すると、処理中であることがコンソールに示されます。さらに、オプショングループが暗号化された DB インスタンスまたは DB スナップショットに関連付けられている場合は、エラーイベントが作成されます。

以下の例では、customerDatabase というデータベースから TDE 暗号化を削除しています。

```
------------- Removing TDE ----------------

USE [customerDatabase]
GO

-- Turn off encryption of the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION OFF
GO

-- Wait until the encryption state of the database becomes 1. The state is 5
 (Decryption in progress) for a while
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO

-- Drop the DEK used for encryption
DROP DATABASE ENCRYPTION KEY
GO

-- Drop a user TDE certificate if it was restored (imported)
EXECUTE msdb.dbo.rds_drop_tde_certificate
 @certificate_name='UserTDECertificate_certificate_name';

-- Alter to SIMPLE Recovery mode so that your encrypted log gets truncated
USE [master]
GO
ALTER DATABASE [customerDatabase] SET RECOVERY SIMPLE
GO
```

すべてのオブジェクトを復号すると、2 つのオプションを使用できます。

1. DB インスタンスを変更して TDE オプションが含まれていないオプショングループに関連付けることができます。
2. オプショングループから TDE オプションを削除できます。

SQL Server Audit

Amazon RDS では、組み込みの SQL Server 監査メカニズムを使用して、Microsoft SQL Server データベースを監査することができます。監査と監査の仕様は、オンプレミスのデータベースサーバー用にそれらを作成するのと同じ方法で作成することができます。

RDS は、完了した監査ログを S3 バケットにアップロードします。この際、作成されたロールを IAM 使用します。保存を有効にしている場合、RDS は設定された期間、監査ログを DB インスタンスに保存します。さらに、CloudWatch (ログストリーム) または両方の送信先に同時に転送することもできます。

詳細については、Microsoft SQL Server ドキュメントの「SQL Server Audit (database engine)」を参照してください。

データベースアクティビティストリームを使用した SQL Server Audit

RDS のデータベースアクティビティストリームを使用すると、SQL Server Audit イベントを Imperva、McAfee、IBM のデータベースアクティビティ監視ツールと統合できます。RDS SQL Server のデータベースアクティビティストリームによる監査の詳細については、「Microsoft SQL Server での監査」を参照してください。

トピック

- SQL Server Audit のサポート
- SQL Server Audit を DB インスタンスオプションに追加する
- SQL Server Audit を使用する
- 監査ログの表示
- マルチ AZ インスタンスで SQL Server Audit を使用する
- S3 バケットを設定する
- CloudWatch ログストリームの設定
- SQL Server Audit の IAM ロールを手動で作成する

SQL Server Audit のサポート

Amazon RDS では、SQL Server 2016 以降、SQL Server のすべてのエディションでサーバーレベルの監査がサポートされており、Enterprise エディションでもデータベースレベルの監査がサポートされています。SQL サーバー 2016 (13.x) SP1 以降では、すべてのエディションでサーバーレベルと

データベースレベルの両方の監査がサポートされています。詳細については、SQL Server ドキュメントの「SQL Server Audit (database engine)」を参照してください。

RDS では、SQL Server Audit に関する次のオプション設定の構成をサポートしています。

オプション設定	有効な値	説明
IAM_ROLE_ARN	形式 arn:aws:iam:: *account-id* :role/*role-name* の有効な Amazon リソースネーム (ARN)。	監査ログを保存する S3 バケットへのアクセスを許可する IAM ロールの ARN。詳細については、AWS 全般のリファレンス の「Amazon リソースネーム (ARN)」を参照してください。
S3_BUCKET_ARN	arn:aws:s3::: *amzn-s3-demo-bucket* 形式または arn:aws:s3::: *amzn-s3-demo-bucket* /key-prefix 形式の有効な ARN。	監査ログを保存する S3 バケットの ARN。
ENABLE_COMPRESSION	true または false	監査ログの圧縮を行います。デフォルトでは、圧縮は有効です (true に設定されている)。
RETENTION_TIME	0 ~ 840	SQL Server 監査レコードが RDS インスタンスに保持される保持期間 (時間単位)。保持設定は、デフォルトでは無効になります。

SQL Server Audit を DB インスタンスオプションに追加する

SQL Server 監査を有効にするには、DB インスタンスでオプションを有効にすることと、SQL Server 内の機能を有効にすることの 2 つのステップを行う必要があります。SQL Server Audit オプションを DB インスタンスに追加するプロセスは次のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. すべての必須オプションを追加して構成します。
3. オプショングループを DB インスタンスに関連付けます。

SQL Server Audit オプションを追加後、DB インスタンスを再起動する必要はありません。オプショングループがアクティブになった時点で、監査を作成して監査ログを S3 バケットに保存できるようになります。

DB インスタンスのオプショングループに SQL Server Audit を追加して設定するには

1. 次のいずれかを選択します。
   - 既存のオプショングループを使用します。
   - カスタムの DB オプショングループを作成し、そのオプショングループを使用します。詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [SQLSERVER_AUDIT] オプションを追加し、オプション設定を構成します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
   - 必要なポリシーがアタッチされた IAM ロールがある場合は、[IAM ロール] で、そのロールを選択できます。新しい IAM ロールを作成するには、[新しいロールの作成] を選択します。必要なポリシーの詳細については、SQL Server Audit の IAM ロールを手動で作成する を参照してください。
   - 使用する S3 バケットが既にある場合は、[S3 送信先の選択] で、そのバケットを選択します。S3 バケットを作成するには、[新しい S3 バケットの作成] を選択します。
   - 監査ファイルを圧縮する場合は、[圧縮の有効化] オプションをオンにします。デフォルトでは、圧縮は有効になっています。圧縮を無効にするには、[Enable Compression (圧縮の有効化)] をオフにします。
   - DB インスタンスの監査レコードを保持するには、[Audit log retention (監査ログの保持)] オプションをオンにします。保持期間 (時間) を指定します。最大保持期間は 35 日間です。
3. 必要に応じて、PUBLISH_TO_CLOUDWATCH オプションをオプショングループに追加して、監査ログに CloudWatch (ログストリーム) 経由でアクセスできるようにします。
4. 新規または既存の DB インスタンスに、DB オプショングループを適用します。次のいずれかを選択します。

- 新しい DB インスタンスを作成する場合は、インスタンスを起動するときにオプショングループを適用します。
- 既存の DB インスタンスで、インスタンスを変更し、新しいオプショングループをアタッチして、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

SQL Server Audit オプションを変更する

SQL Server Audit オプションを有効にしたら、設定を変更できます。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。

DB インスタンスオプションから SQL Server Audit を削除する

SQL Server Audit 機能をオフにするには、監査を無効にし、オプションを削除します。

監査を削除するには

1. SQL Server 内の監査設定をすべて無効にします。監査が実行されている場所を確認するには、SQL Server セキュリティカタログビューをクエリします。詳細については、Microsoft SQL Server ドキュメントの「Security catalog views」を参照してください。
2. DB インスタンスから SQL Server Audit オプションを削除します。次のいずれかを選択します。
   - DB インスタンスで使用しているオプショングループから SQL Server Audit オプションを削除します。この変更は同じオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
   - DB インスタンスを変更し、オプショングループを選択します (SQL Server Audit オプションはオフ)。この変更は、変更した DB インスタンスにのみ影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
3. DB インスタンスから SQL Server Audit オプションを削除後、インスタンスを再起動する必要はありません。S3 バケットから不要な監査ファイルを削除します。

SQL Server Audit を使用する

サーバー監査、サーバー監査の仕様、およびデータベース監査の仕様は、オンプレミスデータベースサーバーの場合と同じ方法で制御することができます。

監査の作成

サーバーの監査は、オンプレミスデータベースサーバーで作成した方法と同じ方法で作成することができます。サーバー監査の作成方法については、Microsoft SQL Server ドキュメントの「CREATE SERVER AUDIT」を参照してください。

エラーを回避するには、次の制限に準拠します。

- インスタンスごとにサポートされるサーバー監査の最大数 50 を超えない。
- データをバイナリファイルに書き込むように SQL Server に指示する。
- サーバーの監査名のプリフィックスとして RDS_ を使用しない。
- FILEPATH で、D:\rdsdbdata\SQLAudit を指定する。
- MAXSIZE で、2 MB ~ 50 MB の間のサイズを指定する。
- MAX_ROLLOVER_FILES または MAX_FILES を設定しない。
- 監査レコードの書き込みに失敗した場合に DB インスタンスをシャットダウンするように SQL Server を構成しない。

監査仕様の作成

サーバー監査仕様とデータベース監査仕様は、オンプレミスデータベースサーバーで作成するのと同じ方法で作成します。監査仕様の作成方法については、Microsoft SQL Server のドキュメントの「CREATE SERVER AUDIT SPECIFICATION」および「CREATE DATABASE AUDIT SPECIFICATION」を参照してください。

エラーを回避するために、データベース監査仕様またはサーバー監査仕様の名前のプリフィックスとして RDS_ を使用しないでください。

監査ログの表示

監査ログは D:\rdsdbdata\SQLAudit に格納されます。

ファイルがサイズ制限に達している場合に SQL Server が監査ログファイルへの書き込みを完了すると、Amazon RDS はファイルを S3 バケットにアップロードします。保持設定が有効になっている場合、Amazon RDS は、このファイルを保持フォルダ (D:\rdsdbdata\SQLAudit\transmitted) に移動します。

保持設定については、SQL Server Audit を DB インスタンスオプションに追加する を参照してください。

監査ログファイルがアップロードされるまで、監査レコードは DB インスタンスに維持されます。監査レコードを表示するには、次のコマンドを実行します。

```
SELECT   *
 FROM     msdb.dbo.rds_fn_get_audit_file
              ('D:\rdsdbdata\SQLAudit\*.sqlaudit'
              , default
              , default )
```

同じコマンドを使用して、保持フォルダの監査レコードを表示するには、フィルタを `D:\rdsdbdata\SQLAudit\transmitted\*.sqlaudit` に変更します。

```
SELECT   *
 FROM     msdb.dbo.rds_fn_get_audit_file
              ('D:\rdsdbdata\SQLAudit\transmitted\*.sqlaudit'
              , default
              , default )
```

CloudWatch での監査ログレコードの表示

監査ログを CloudWatch にアップロードすることを選択した場合は、以下の手順に従ってログレコード (JSON 形式) にアクセスできます。

CloudWatch で監査ログを表示するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで、[ログ]、[ログ管理] の順に選択します。
3. [ロググループ] 検索ボックスに DB インスタンス名を入力します。
4. 適切なロググループを選択します。
5. [ログストリーム] タブで、ログストリームを選択します。

あるいは、AWS CLI を使用してログストリームをクエリすることもできます。次の例では、監査ログイベントを抽出し、CSV ファイルにエクスポートします。

```
# Parameters
# -----------
LOG_GROUP="/aws/rds/instance/<your-db-instance-id>/sqlaudit"
START_DATE="2026-06-18"
```

```
END_DATE="2026-06-19"
OUTPUT_FILE="sql_audit_extract.csv"

# Convert dates to epoch milliseconds
# -----------------------------------
START_TIME=$(date -d "${START_DATE}" +%s)000
END_TIME=$(date -d "${END_DATE}" +%s)000

# Fetch log events
# ----------------
# Note: --no-paginate returns all results in a single call.
# For large result sets, consider using --page-size or manual pagination.
RESPONSE=$(aws logs filter-log-events \
  --log-group-name "${LOG_GROUP}" \
  --start-time "${START_TIME}" \
  --end-time "${END_TIME}" \
  --output json \
  --no-paginate)

# Parse and export to CSV
# Extract the 'message' field from each event (JSON formatted messages),
# then convert to CSV using jq.
# Get CSV headers from the first record
# -------------------------------------
HEADERS=$(echo "${RESPONSE}" | jq -r '
  [.events[0].message | fromjson | keys[]] | @csv
')

# Write headers to output file
# ----------------------------
echo "${HEADERS}" > "${OUTPUT_FILE}"

# Write all records to output file
# --------------------------------
echo "${RESPONSE}" | jq -r '
  .events[].message | fromjson | [.[]] | @csv
' >> "${OUTPUT_FILE}"

# Script execution summary
# ------------------------
RECORD_COUNT=$(echo "${RESPONSE}" | jq '.events | length')
echo "Exported ${RECORD_COUNT} records to ${OUTPUT_FILE}"
```

マルチ AZ インスタンスで SQL Server Audit を使用する

マルチ AZ インスタンスで、監査ログファイルを Amazon S3 に送信するためのプロセスは、単一 AZ インスタンスのプロセスと似ています。ただし、重要な相違点がいくつかあります。

- データベース監査仕様オブジェクトはすべてのノードにレプリケートされます。
- サーバー監査およびサーバー監査仕様は、セカンダリノードにはレプリケートされません。その代わりに、手動で作成または変更する必要があります。

サーバー監査またはサーバー監査仕様を両方のノードからキャプチャするには

1. サーバー監査またはサーバー監査仕様をプライマリノードに作成します。
2. セカンダリノードにフェイルオーバーし、セカンダリノードに同じ名前と GUID を持つサーバー監査仕様またはサーバー監査仕様を作成します。GUID を指定するには、`AUDIT_GUID` パラメータを使用します。

S3 バケットを設定する

監査ログファイルは、DB インスタンスから S3 バケットに自動的にアップロードされます。監査ファイルのターゲットとして使用する S3 バケットには、以下の制限が適用されます。

- DB インスタンスと同じ AWS リージョンおよび AWS アカウントにある必要があります。
- 公開することは禁止されています。
- バケット所有者は IAM ロール所有者でもある必要があります。
- IAM ロールには、S3 バケットのサーバー側の暗号化に関連付けられたカスタマー管理の KMS キーに対するアクセス許可が必要です。

データを格納するために使用されるターゲットキーは、次の命名スキーマに従います: *amzn-s3-demo-bucket*`/key-prefix/instance-name/audit-name/node_file-name.ext`

> **Note**
>
> バケット名とキープレフィックス値の両方を (`S3_BUCKET_ARN`) オプションで設定します。

このスキーマは、以下の要素で構成されています。

- ***amzn-s3-demo-bucket*** – S3 バケットの名前。
- **key-prefix** - 監査ログに使用するカスタムキープレフィックス。
- **instance-name** - Amazon RDS インスタンスの名前。
- **audit-name** - 監査の名前。
- **node** - 監査ログの出典であるノードの識別子 (node1 または node2)。シングル AZ インスタンスには 1 つのノード、マルチ AZ インスタンスには 2 つのレプリケーションノードがあります。プライマリノードとセカンダリノードのロールは時間とともに変化するため、これらはプライマリノードとセカンダリノードではありません。その代わりに、ノード識別子はシンプルなラベルです。
  - **node1** - 初期のレプリケーションノード (シングル AZ には 1 つのノードのみがあります)。
  - 2 **node2** -番目のレプリケーションノード (マルチ AZ には 2 つのノードがあります)。
- **file-name** - ターゲットファイルの名前。ファイル名は SQL Server からそのまま取得されます。
- **ext** - ファイルの拡張子 (zip または sqlaudit)。
  - **zip** - 圧縮が有効になっているかどうか (デフォルト)。
  - **sqlaudit** - 圧縮が無効になっているかどうか。

CloudWatch ログストリームの設定

デフォルトでは、Amazon RDS for SQL Server Audit は完了した監査ファイルを Amazon S3 バケットに配信します。また、監査ログを Amazon S3 の代わりに、または Amazon S3 に加えて Amazon CloudWatch Logs にストリーミングすることもできます。

監査ログを Amazon CloudWatch Logs にストリーミングすると、次のことが可能になります。

- 監査レコードを、ユーザーが定義する保持期間で耐久性の高いストレージに保存する。
- 監査ログデータを検索してフィルタリングし、メトリクスフィルターと CloudWatch アラームを構築する。
- アカウント間でログデータを共有し、Amazon S3 にエクスポートする。
- データをストリーミングするか、リアルタイムで処理する。

CloudWatch ストリーミングの有効化

CloudWatch ストリーミングは、SQLSERVER_AUDIT オプションの単一のブール設定 PUBLISH_TO_CLOUDWATCH によって制御されます。監査ログを Amazon CloudWatch Logs にストリーミングするには true に設定し、停止するには false に設定します。

> **Note**
>
> CloudWatch パブリッシングでは、カスタマーの IAM ロールは必要ありません。PUBLISH_TO_CLOUDWATCH が true の場合、Amazon RDS は Amazon CloudWatch Logs への監査ログの配信を管理します。その IAM ロールやアクセス許可を作成またはアタッチする必要はありません。CloudWatch はスタンドアロンの送信先とすることができるため、IAM_ROLE_ARN と S3_BUCKET_ARN はオプションになります。S3 の送信先も必要な場合にのみ指定してください。少なくとも 1 つの送信先 (CloudWatch、S3、またはその両方) を設定する必要があります。

新しいオプショングループの場合

SQLSERVER_AUDIT オプションをオプショングループに追加する場合 (詳細な手順については「SQL Server Audit を DB インスタンスオプションに追加する」を参照)、PUBLISH_TO_CLOUDWATCH 設定を true に設定します。

既存のオプショングループの場合

既存のオプショングループで CloudWatch ストリーミングを有効にするには

1. Amazon RDS コンソールを開き、[オプショングループ] を選択して、DB インスタンスにアタッチされているオプショングループを開きます。
2. [オプショングループの変更] を選択します。
3. 既存の SQLSERVER_AUDIT オプションで、PUBLISH_TO_CLOUDWATCH 設定を true に設定します。
4. 変更を保存し、今すぐ適用するか、次のメンテナンスウィンドウ中に適用するかを選択します。

1 つのオプショングループを複数の DB インスタンスで共有できるため、この変更は、そのオプショングループを使用するすべての DB インスタンスに影響します。インスタンスを 1 つだけ変更するには、別のオプショングループを使用します。

オプショングループがアクティブになったら、DB インスタンスを再起動する必要はありません。変更が適用されるとすぐに、監査ログが Amazon CloudWatch Logs へのストリーミングを開始します。ストリーミングを停止するには、PUBLISH_TO_CLOUDWATCH を false に戻します。

監査ログが CloudWatch に表示される場所

Amazon RDS は、SQL Server 監査ログを次の名前の RDS 管理 CloudWatch Logs ロググループに発行します。

```
/aws/rds/instance/db_instance_name/sqlaudit
```

このロググループ内では、各ノードはノード識別子によって識別される独自のログストリームに発行されます。マルチ AZ (常時オン) インスタンスの場合、プライマリとセカンダリはそれぞれ、ノード識別子で識別される同じsqlaudit ロググループ内の個別のログストリームに書き込むため、両方のレプリカからの監査イベントは個別にキャプチャされます。

バージョンの可用性

CloudWatch への監査ログのストリーミングは、SQL Server Audit に対応しているすべてのエンジンバージョンとエディションで利用できます。SQL Server Audit に対応していれば、追加のバージョン制限はありません。

SQL Server Audit の IAM ロールを手動で作成する

通常、新しいオプションを作成すると、AWS マネジメントコンソール は IAM ロールと IAM 信頼ポリシーを作成します。ただし、SQL Server Audits で使用する新しい IAM ロールを手動で作成して、必要に応じてその他の要件に合わせてカスタマイズできます。そのためには、Amazon RDS サービスで Amazon S3 バケットを使用できるように、IAM ロールを作成して、アクセス許可を委任します。この IAM ロールを作成したら、信頼ポリシーとアクセス許可ポリシーをアタッチします。信頼ポリシーを使用することで、Amazon RDS はこのロールを引き受けることができます。アクセス許可ポリシーでは、このロールが実行できるアクションを定義します。詳細については、AWS Identity and Access Management ユーザーガイドのAWS のサービスに許可を委任するロールの作成を参照してください。

必要な信頼関係とアクセス許可ポリシーは、このセクションの例を使用して作成できます。

以下の例は、SQL Server Audit の信頼関係を示しています。この例では、サービスプリンシパル rds.amazonaws.com を使用して、RDS で S3 バケットに書き込めるようにしています。サービスプリンシパルは、サービスにアクセス許可を付与するために使用される識別子です。このように、

いつでも rds.amazonaws.com にアクセス許可を付与できます。これにより、RDS はお客様に代わってアクションを実行できるようになります。サービスプリンシパルの詳細については、AWS JSON ポリシーの要素: プリンシパルを参照してください。

Example SQL Server Audit の信頼関係

JSON

```
{
     "Version":"2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Principal": {
                 "Service": "rds.amazonaws.com"
             },
             "Action": "sts:AssumeRole"
         }
     ]
 }
```

リソースベースの信頼関係では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与する特定のリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

両方のグローバル条件コンテキストキーを使用し、aws:SourceArn 値にアカウント ID を含めます。この場合は、aws:SourceAccount 値と aws:SourceArn 値のアカウントは、同じステートメントで使用する場合、同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

信頼関係では、aws:SourceArn グローバル条件コンテキストキーに、必ず、ロールにアクセスするリソースの完全な Amazon リソースネーム (ARN) を使用します。SQL Server Audit の場合は、次の例に示すように、DB オプショングループと DB インスタンスの両方を必ず含めるようにしてください。

Example SQL Server Audit のグローバル条件コンテキストキーとの信頼関係

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [

 "arn:aws:rds:Region:my_account_ID:db:db_instance_identifier",
                        "arn:aws:rds:Region:my_account_ID:og:option_group_name"
                    ]
                }
            }
        }
    ]
}
```

SQL Server Audit のアクセス許可ポリシーの次の例では、Amazon S3 バケットの ARN を指定します。ARN を使用して、アクセス許可を付与する特定のアカウント、ユーザー、またはロールを識別します。ARN の使用の詳細については、Amazon リソースネーム (ARN) を参照してください。

Example SQL Server Audit のアクセス許可ポリシー

JSON

```
{
     "Version":"2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
```

```
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketACL",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/key_prefix/*"
        }
    ]
}
```

> **Note**
>
> `s3:ListAllMyBuckets` アクションは、同じ AWS アカウントが S3 バケットと SQL Server DB インスタンスの両方を所有していることを確認するために必要です。このアクションは、アカウント内のバケットの名前を一覧表示します。
> S3 バケットの名前空間はグローバルです。誤ってバケットを削除した場合は、別のユーザーが同じ名前のバケットを別のアカウントで作成できます。次に、SQL Server 監査データが新しいバケットに書き込まれます。

> **Note**
>
> この IAM ロールは、S3 監査先でのみ使用されます。また、監査ログを Amazon CloudWatch Logs にストリーミングする場合 (`PUBLISH_TO_CLOUDWATCH` オプション – 「CloudWatch ログストリームの設定」を参照)、このロールに CloudWatch Logs アクセス許可 (`logs:*`) を追加する必要はありません。CloudWatch 配信は、RDS マネージドサービスにリンクされた

ロール (AWSServiceRoleForRDS) によって処理されます。上記の S3 アクセス許可で十分です。

Amazon RDS for SQL Server での SQL Server Analysis Services のサポート

Microsoft SQL Server Analysis Services (SSAS) は、Microsoft ビジネスインテリジェンス (MSBI) スイートの一部です。SSAS は、SQL Server 内にインストールされているオンライン分析処理 (OLAP) およびデータマイニングツールです。SSAS を使用してデータを分析し、ビジネス上の意思決定に役立てます。SSAS は、ビジネスインテリジェンス環境の一般的なクエリや計算用に最適化されているという点で、SQL Server リレーショナルデータベースとは異なります。

SSAS は、既存または新規の DB インスタンスに対してオンにすることができます。データベースエンジンと同じ DB インスタンスにインストールされます。SSAS の詳細については、Microsoft の Analysis Services のドキュメントを参照してください。

Amazon RDS は、次のバージョンで SQL Server スタンダードエディションおよびエンタープライズエディションの SSAS をサポートします。

- 表形式モード:
  - SQL Server 2019、バージョン 15.00.4043.16.v1 以降
  - SQL Server 2017、バージョン 14.00.3223.3.v1 以降
  - SQL Server 2016、バージョン 13.00.5426.0.v1 以降
- 多次元モード:
  - SQL Server 2019、バージョン 15.00.4153.1.v1 以降
  - SQL Server 2017、バージョン 14.00.3381.3.v1 以降
  - SQL Server 2016、バージョン 13.00.5882.1.v1 以降

> **Note**
>
> Amazon RDS は SSAS for SQL Server 2025 をサポートしていません。サポートは、今後のリリースで予定されています。

目次

- 制限事項
- SSAS をオンにする
  - SSAS のオプショングループの作成

  - SSAS オプションをオプショングループに追加する
  - オプショングループを DB インスタンスに関連付ける
  - VPC セキュリティグループへのインバウンドアクセスの許可
  - Amazon S3 統合の有効化
- Amazon RDS への SSAS プロジェクトのデプロイ
- デプロイタスクのステータスのモニタリング
- Amazon RDS での SSAS の使用
  - SSAS 用の Windows 認証ユーザーの設定
  - データベース管理者としてのドメインユーザーの追加
  - SSAS プロキシの作成
  - SQL Server エージェントを使用した SSAS データベース処理のスケジュール
  - プロキシからの SSAS アクセスの取り消し
- SSAS データベースのバックアップ
- SSAS データベースの復元
  - 特定の時点への DB インスタンスの復元
- SSAS モードの変更
- SSAS をオフにする
- SSAS の問題のトラブルシューティング

制限事項

RDS for SQL Server で SSAS を使用する場合は、次の制限が適用されます。

- RDS for SQL Server は、表形式モードまたは多次元モードでの SSAS の実行をサポートしています。詳細については、Microsoft のドキュメントの「テーブルソリューションと多次元ソリューションの比較」を参照してください。
- 一度に使用できる SSAS モードは 1 つだけです。モードを変更する前に、すべての SSAS データベースを必ず削除してください。

  詳細については、「SSAS モードの変更」を参照してください。
- マルチ AZ インスタンスはサポートされていません。

- インスタンスは自己管理型の Active Directory または AWS Directory Service for Microsoft Active Directory for SSAS 認証を使用する必要があります。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。
- ユーザーには、SSAS サーバーの管理者アクセス権は付与されませんが、データベースレベルの管理者アクセス権が付与されます。
- SSAS へのアクセスにサポートされているポートは 2383 のみです。
- プロジェクトを直接デプロイすることはできません。これを行うには、用意されている RDS ストアドプロシージャを使用できます。詳細については、「Amazon RDS への SSAS プロジェクトのデプロイ」を参照してください。
- デプロイ中の処理はサポートされていません。
- デプロイでの .xmla ファイルの使用はサポートされていません。
- SSAS のプロジェクト入力ファイルとデータベースバックアップ出力ファイルは、DB インスタンスの `D:\S3` フォルダにのみ配置できます。

SSAS をオンにする

DB インスタンスの SSAS をオンにするには、次のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. オプショングループに [SSAS] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。
4. SSAS リスナーポートの仮想プライベートクラウド (VPC) セキュリティグループへのインバウンドアクセスを許可します。
5. Amazon S3 統合をオンにします。

SSAS のオプショングループの作成

AWS マネジメントコンソール または AWS CLI を使用して、使用する DB インスタンスの SQL Server エンジンおよびバージョンに対応するオプショングループを作成します。

> **Note**
>
> 既存のオプショングループが正しい SQL Server エンジンおよびバージョンに対応している場合は、それを使用することもできます。

コンソール

次の手順では、コンソールを使用して SQL Server Standard Edition 2017 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [オプショングループの作成] ウィンドウで、次の操作を行います。
   a. [名前] に、AWS アカウント内で一意のオプショングループ名 (**ssas-se-2017** など) を入力します。名前には、英字、数字、ハイフンのみを使用できます。
   b. [説明] に、オプショングループの簡単な説明 (**SSAS option group for SQL Server SE 2017** など) を入力します。この説明は表示用に使用されます。
   c. [エンジン] で [sqlserver-se] を選択します。
   d. [メジャーエンジンのバージョン] で、[14.00] を選択します。
5. [Create] (作成) を選択します。

CLI

次の例では、CLI を使用して SQL Server Standard Edition 2017 のオプショングループを作成します。

オプショングループを作成するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name ssas-se-2017 \
    --engine-name sqlserver-se \
    --major-engine-version 14.00 \
```

```
    --option-group-description "SSAS option group for SQL Server SE 2017"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name ssas-se-2017 ^
    --engine-name sqlserver-se ^
    --major-engine-version 14.00 ^
    --option-group-description "SSAS option group for SQL Server SE 2017"
```

SSAS オプションをオプショングループに追加する

次に、AWS マネジメントコンソール または AWS CLI を使用して SSAS オプションをオプショングループに追加します。

コンソール

SSAS オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 先ほど作成したオプショングループを選択します。
4. [オプションの追加] を選択します。
5. [オプションの詳細] で、[オプション名] として SSAS を選択します。
6. [オプション設定] で、次の操作を行います。

    a. [Max memory] (最大メモリ) に、10 ~ 80 の範囲の値を入力します。

    [最大メモリ] は、メモリの上限しきい値を指定します。この値を超えると、実行中のリクエスト用と新しい優先順位の高いリクエスト用のスペースを確保するために、SSAS がメモリの解放を積極的にスタートします。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10〜80 で、デフォルトは 45 です。

    b. [Mode] (モード) で、SSAS サーバーモードとして [Tabular] (表形式) または [Multidimensional] (多次元) を選択します。

[Mode] (モード) オプション設定が表示されない場合は、自身の AWS リージョンで多次元モードがサポートされていないことを意味します。詳細については、「制限事項」を参照してください。

[Tabular] (表形式) はデフォルトです。

c. [セキュリティグループ] で、オプションに関連付ける VPC セキュリティグループを選択します。

> **Note**
>
> SSAS にアクセスするためのポート 2383 は事前に設定されています。

7. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
8. [オプションを追加] を選択します。

CLI

SSAS オプションを追加するには

1. 次のパラメータを使用して JSON ファイル (`ssas-option.json` など) を作成します。

   - `OptionGroupName` - 以前に作成または選択したオプショングループの名前 (次の例では `ssas-se-2017`)。
   - `Port` - SSAS にアクセスするために使用するポート。サポートされているポートは 2383 のみです。
   - `VpcSecurityGroupMemberships` – RDS DB インスタンスの VPC セキュリティグループのメンバーシップ。
   - `MAX_MEMORY` - メモリの上限しきい値。この値を超えると、実行中のリクエスト用および新しい優先順位の高いリクエスト用のスペースを確保するために、SSAS がメモリの解放を積極的にスタートします。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10〜80 で、デフォルトは 45 です。
   - `MODE` – SSAS サーバモード。`Tabular` または `Multidimensional`。`Tabular` がデフォルトです。

MODE オプション設定が有効ではないというエラーは、多次元モードが自身の AWS リージョンでサポートされていないという意味です。詳細については、「制限事項」を参照してください。

SSAS オプション設定を含む JSON ファイルの例を次に示します。

```
{
"OptionGroupName": "ssas-se-2017",
"OptionsToInclude": [
 {
 "OptionName": "SSAS",
 "Port": 2383,
 "VpcSecurityGroupMemberships": ["sg-0abcdef123"],
 "OptionSettings": [{"Name":"MAX_MEMORY","Value":"60"},
{"Name":"MODE","Value":"Multidimensional"}]
 }],
"ApplyImmediately": true
}
```

2. オプショングループに [SSAS] オプションを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --cli-input-json file://ssas-option.json \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://ssas-option.json ^
    --apply-immediately
```

オプショングループを DB インスタンスに関連付ける

コンソールまたは CLI を使用して、オプショングループを DB インスタンスに関連付けることができます。

コンソール

オプショングループを新規または既存の DB インスタンスに関連付けます。

- 新規の DB インスタンスの場合は、DB インスタンスの起動時にオプショングループを DB インスタンスと関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、DB インスタンスを変更してから、新しいオプショングループと関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> 既存のインスタンスを使用する場合は、このインスタンスに Active Directory ドメインと AWS Identity and Access Management (IAM) ロールが既に関連付けられている必要があります。新しいインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。

CLI

オプショングループを新規または既存の DB インスタンスに関連付けることができます。

> **Note**
>
> 既存のインスタンスを使用する場合は、このインスタンスに Active Directory ドメインと IAM ロールが既に関連付けられている必要があります。新しいインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。

オプショングループを使用する DB インスタンスを作成するには

- オプショングループの作成時に使用したのと同じ DB エンジンのタイプとメジャーバージョンを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier myssasinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 14.00.3223.3.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssas-se-2017
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier myssasinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 14.00.3223.3.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssas-se-2017
```

DB インスタンスを変更してオプショングループを関連付けるには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier myssasinstance \
    --option-group-name ssas-se-2017 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssasinstance ^
    --option-group-name ssas-se-2017 ^
    --apply-immediately
```

VPC セキュリティグループへのインバウンドアクセスの許可

DB インスタンスに関連付けられた VPC セキュリティグループで、指定した SSAS リスナーポートのインバウンドルールを作成します。セキュリティグループの設定の詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

Amazon S3 統合の有効化

モデル設定ファイルをホストにダウンロードしてデプロイするには、Amazon S3 統合を使用します。詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。

Amazon RDS への SSAS プロジェクトのデプロイ

RDS では、SQL Server Management Studio (SSMS) を使用して SSAS プロジェクトを直接デプロイすることはできません。プロジェクトをデプロイするには、RDS ストアドプロシージャを使用します。

> **Note**
>
> デプロイでの .xmla ファイルの使用はサポートされていません。

プロジェクトをデプロイする前に、以下を確認してください。

- Amazon S3 統合がオンになっている。詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。
- `Processing Option` 構成設定が `Do Not Process` に設定されている。この設定は、デプロイ後に処理が実行されないことを意味します。
- *`myssasproject`*`.asdatabase` ファイルと *`myssasproject`*`.deploymentoptions` ファイルの両方がある。これらは、SSAS プロジェクトの構築時に自動的に生成されます。

SSAS プロジェクトを RDS にデプロイするには

1. 次の例に示すように、`.asdatabase` (SSAS モデル) を S3 バケットから DB インスタンスにダウンロードします。ダウンロードパラメータの詳細については、「Amazon S3 バケットから SQL Server DB インスタンスにファイルをダウンロードする」を参照してください。

```
exec msdb.dbo.rds_download_from_s3
@s3_arn_of_file='arn:aws:s3:::bucket_name/myssasproject.asdatabase',
[@rds_file_path='D:\S3\myssasproject.asdatabase'],
[@overwrite_file=1];
```

2. `.deploymentoptions` ファイルを S3 バケットから DB インスタンスにダウンロードします。

```
exec msdb.dbo.rds_download_from_s3
@s3_arn_of_file='arn:aws:s3:::bucket_name/myssasproject.deploymentoptions',
[@rds_file_path='D:\S3\myssasproject.deploymentoptions'],
[@overwrite_file=1];
```

3. プロジェクトをデプロイします。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSAS_DEPLOY_PROJECT',
@file_path='D:\S3\myssasproject.asdatabase';
```

デプロイタスクのステータスのモニタリング

デプロイ (またはダウンロード) タスクのステータスを追跡するには、`rds_fn_task_status` 関数を呼び出します。2 つのパラメータを使用します。1 つめのパラメータは、SSAS に適用されないため、常に `NULL` を設定してください。2 つめのパラメータは、タスク ID を受け入れます。

全タスクのリストを見るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータを 0 に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

特定のタスクを受け取るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータをタスク ID に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 機能は次の情報を返します。

出力パラメータ	説明
task_id	タスクの ID。
task_type	SSAS では、次のタスクタイプを使用できます。 • SSAS_DEPLOY_PROJECT • SSAS_ADD_DB_ADMIN_MEMBER • SSAS_BACKUP_DB • SSAS_RESTORE_DB
database_name	SSAS タスクは該当しません。
% complete	タスクの進行状況の割合。
duration (mins)	タスクにかかった時間 (分単位)。
lifecycle	タスクのステータス。有効な状態には以下のものがあります。 • CREATED - SSAS ストアドプロシージャのいずれかを呼び出すと、タスクが作成され、ステータスが に設定されます。CREATED •

出力パラメータ	説明
	IN_PROGRESS - タスクがスタートすると、ステータスが に設定されます。IN_PROGRESS ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。 • SUCCESS - タスクが完了すると、ステータスが に設定されます。SUCCESS • ERROR - タスクが失敗すると、ステータスがに設定されます。ERROR工エラーの詳細については、task_info 列を参照してください。 • CANCEL_REQUESTED - rds_cancel_task を呼び出すと、タスクのステータスが CANCEL_REQUESTED になります。 • CANCELLED - タスクが正常にキャンセルされると、タスクのステータスが に設定されます。CANCELLED
task_info	タスクに関する追加情報。処理中にエラーが発生した場合、この列にエラーに関する情報が含まれます。 詳細については、「SSAS の問題のトラブルシューティング」を参照してください。
last_updated	タスクのステータスが最後に更新された日時。
created_at	タスクが作成された日時。
S3_object_arn	SSAS タスクは該当しません。

出力パラメータ	説明
overwrite_S3_backup_file	SSAS タスクは該当しません。
KMS_master_key_arn	SSAS タスクは該当しません。
filepath	SSAS タスクは該当しません。
overwrite_file	SSAS タスクは該当しません。
task_metadata	SSAS タスクに関連付けられたメタデータ。

Amazon RDS での SSAS の使用

SSAS プロジェクトをデプロイすると、SSMS で OLAP データベースを直接処理できます。

RDS で SSAS を使用するには

1. SSMS で、Active Directory ドメインのユーザー名とパスワードを使用して SSAS に接続します。
2. [データベース] を展開します。新しくデプロイした SSAS データベースが表示されます。
3. 接続文字列を見つけて、ユーザー名とパスワードを、ソース SQL データベースにアクセスできるように更新します。この操作は、SSAS オブジェクトを処理するために必要です。

    a. 表形式モードの場合、次の操作を行います。

        1. [Connections] (接続) タブを展開します。
        2. 右クリックで接続オブジェクトのメニューを開き、[Properties] (プロパティ) を選択します。
        3. 接続文字列のユーザー名とパスワードを更新します。

    b. 多次元モードの場合、次の操作を行います。

        1. [Data Sources] (データソース) タブを展開します。
        2. 右クリックでデータソースオブジェクトのメニューを開き、[Properties] (プロパティ) を選択します。
        3. 接続文字列のユーザー名とパスワードを更新します。

4. 作成した SSAS データベースのコンテキスト (右クリック) メニューを開き、[データベースの処理] を選択します。

   入力データのサイズによっては、処理オペレーションが完了するまでに数分かかる場合があります。

トピック

- SSAS 用の Windows 認証ユーザーの設定
- データベース管理者としてのドメインユーザーの追加
- SSAS プロキシの作成
- SQL Server エージェントを使用した SSAS データベース処理のスケジュール
- プロキシからの SSAS アクセスの取り消し

SSAS 用の Windows 認証ユーザーの設定

メイン管理者ユーザー (マスターユーザーと呼ばれることもあります) は、以下のコード例を使用して、Windows 認証ログインを設定し、必要な手順に対するアクセス許可を付与できます。これにより、ドメインユーザーに SSAS カスタマータスクの実行、S3 ファイル転送手順の使用、認証情報の作成、および SQL Server エージェントプロキシの操作を行うアクセス許可が付与されます。詳細については、Microsoft ドキュメントの「Credentials (Database Engine)」および「Create a SQL Server Agent Proxy」を参照してください。

必要に応じて、Windows 認証ユーザーに以下のアクセス許可の一部またはすべてを付与できます。

Example

```
-- Create a server-level domain user login, if it doesn't already exist
USE [master]
GO
CREATE LOGIN [mydomain\user_name] FROM WINDOWS
GO

-- Create domain user, if it doesn't already exist
USE [msdb]
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]
GO

-- Grant necessary privileges to the domain user
```

```
USE [master]
GO
GRANT ALTER ANY CREDENTIAL TO [mydomain\user_name]
GO

USE [msdb]
GO
GRANT EXEC ON msdb.dbo.rds_msbi_task TO [mydomain\user_name] with grant option
GRANT SELECT ON msdb.dbo.rds_fn_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_cancel_task TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_download_from_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_upload_to_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_delete_from_filesystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_gather_file_details TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_add_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_update_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_grant_login_to_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_revoke_login_from_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_delete_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_enum_login_for_proxy to [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_enum_proxy_for_subsystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_sqlagent_proxy TO [mydomain\user_name] with grant option
ALTER ROLE [SQLAgentUserRole] ADD MEMBER [mydomain\user_name]
GO
```

データベース管理者としてのドメインユーザーの追加

次の方法で、ドメインユーザーを SSAS データベース管理者として追加できます。

- データベース管理者は、SSMS を使用して admin 権限を持つロールを作成し、そのロールにユーザーを追加できます。
- 次のストアドプロシージャを使用できます。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSAS_ADD_DB_ADMIN_MEMBER',
@database_name='myssasdb',
```

```
@ssas_role_name='exampleRole',
@ssas_role_member='domain_name\domain_user_name';
```

以下のパラメータは必須です。

- @task_type - MSBI タスクのタイプ (この例では SSAS_ADD_DB_ADMIN_MEMBER)。
- @database_name - 管理者権限を付与する先の SSAS データベースの名前。
- @ssas_role_name - SSAS データベース管理者ロールの名前。ロールがまだ存在しない場合は、作成されます。
- @ssas_role_member - 管理者ロールに追加する SSAS データベースユーザー。

SSAS プロキシの作成

SQL Server エージェントを使用して SSAS データベース処理をスケジュールできるようにするには、SSAS 認証情報と SSAS プロキシを作成します。これらの手順を Windows 認証ユーザーとして実行します。

SSAS 認証情報を作成するには

- プロキシの認証情報を作成します。そのためには、SSMS または以下の SQL ステートメントを使用できます。

```
USE [master]
GO
CREATE CREDENTIAL [SSAS_Credential] WITH IDENTITY = N'mydomain\user_name', SECRET =
 N'mysecret'
GO
```

> **Note**
>
> IDENTITY はドメイン認証ログインであることが必要です。*mysecret* をドメイン認証ログインのパスワードに置き換えます。

SSAS プロキシを作成するには

1. 以下の SQL ステートメントを使用して、プロキシを作成します。

```
USE [msdb]
```

```
GO
EXEC msdb.dbo.sp_add_proxy
 @proxy_name=N'SSAS_Proxy',@credential_name=N'SSAS_Credential',@description=N''
GO
```

2. 以下の SQL ステートメントを使用して、他のユーザーにプロキシへのアクセスを許可します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_grant_login_to_proxy
 @proxy_name=N'SSAS_Proxy',@login_name=N'mydomain\user_name'
GO
```

3. 以下の SQL ステートメントを使用して、SSAS サブシステムにプロキシへのアクセスを許可します。

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='GRANT_SUBSYSTEM_ACCESS',@proxy_name='SSAS_Proxy',@proxy_subsystem='SSAS'
GO
```

プロキシとそのプロキシに対する許可を表示するには

1. 以下の SQL ステートメントを使用して、プロキシの被付与者を表示します。

```
USE [msdb]
GO
EXEC sp_help_proxy
GO
```

2. 以下の SQL ステートメントを使用して、サブシステムの許可を表示します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_enum_proxy_for_subsystem
GO
```

SQL Server エージェントを使用した SSAS データベース処理のスケジュール

認証情報とプロキシを作成し、SSAS にプロキシへのアクセスを許可したら、SQL Server エージェントジョブを作成して SSAS データベース処理をスケジュールできます。

SSAS データベース処理をスケジュールするには

- SSMS または T-SQL を使用して、SQL Server エージェントジョブを作成します。以下の例では T-SQL を使用しています。SSMS または T-SQL を使用して、ジョブスケジュールをさらに設定できます。

  - `@command` パラメータは、SQL Server エージェントジョブによって実行される XML for Analysis (XMLA) コマンドの概要を示します。この例では、SSAS 多次元データベース処理を設定します。
  - `@server` パラメータは、SQL Server エージェントジョブのターゲット SSAS サーバー名の概要を示しています。

    SQL Server エージェントジョブが存在する同じ RDS DB インスタンス内で SSAS サービスを呼び出すには、`localhost:2383` を使用します。

    RDS DB インスタンスの外部から SSAS サービスを呼び出すには、RDS エンドポイントを使用します。RDS DB インスタンスが同じドメインで参加している場合、Kerberos Active Directory (AD) エンドポイント (*`your-DB-instance-name`*.*`your-AD-domain-name`*) も使用できます。外部 DB インスタンスの場合は、RDS DB インスタンスに関連付けられた VPC セキュリティグループをセキュアな接続用に適切に設定してください。

  クエリをさらに編集して、さまざまな XMLA オペレーションをサポートできます。T-SQL クエリを直接変更するか、SQL Server エージェントジョブの作成後に SSMS UI を使用して編集を行います。

```
USE [msdb]
GO
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'SSAS_Job',
    @enabled=1,
    @notify_level_eventlog=0,
    @notify_level_email=0,
    @notify_level_netsend=0,
    @notify_level_page=0,
    @delete_level=0,
```

```
    @category_name=N'[Uncategorized (Local)]',
    @job_id = @jobId OUTPUT
GO
EXEC msdb.dbo.sp_add_jobserver
    @job_name=N'SSAS_Job',
    @server_name = N'(local)'
GO
EXEC msdb.dbo.sp_add_jobstep @job_name=N'SSAS_Job',
 @step_name=N'Process_SSAS_Object',
    @step_id=1,
    @cmdexec_success_code=0,
    @on_success_action=1,
    @on_success_step_id=0,
    @on_fail_action=2,
    @on_fail_step_id=0,
    @retry_attempts=0,
    @retry_interval=0,
    @os_run_priority=0, @subsystem=N'ANALYSISCOMMAND',
    @command=N'<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/
engine">
        <Parallel>
            <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/
engine/2" xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/
engine/2/2"
                xmlns:ddl100_100="http://schemas.microsoft.com/
analysisservices/2008/engine/100/100" xmlns:ddl200="http://schemas.microsoft.com/
analysisservices/2010/engine/200"
                xmlns:ddl200_200="http://schemas.microsoft.com/
analysisservices/2010/engine/200/200" xmlns:ddl300="http://schemas.microsoft.com/
analysisservices/2011/engine/300"
                xmlns:ddl300_300="http://schemas.microsoft.com/
analysisservices/2011/engine/300/300" xmlns:ddl400="http://schemas.microsoft.com/
analysisservices/2012/engine/400"
                xmlns:ddl400_400="http://schemas.microsoft.com/
analysisservices/2012/engine/400/400" xmlns:ddl500="http://schemas.microsoft.com/
analysisservices/2013/engine/500"
                xmlns:ddl500_500="http://schemas.microsoft.com/
analysisservices/2013/engine/500/500">
                <Object>
                    <DatabaseID>Your_SSAS_Database_ID</DatabaseID>
                </Object>
                <Type>ProcessFull</Type>
```

```
                <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
            </Process>
        </Parallel>
    </Batch>',
    @server=N'localhost:2383',
    @database_name=N'master',
    @flags=0,
    @proxy_name=N'SSAS_Proxy'
GO
```

プロキシからの SSAS アクセスの取り消し

以下のストアドプロシージャを使用して、SSAS サブシステムへのアクセスを取り消し、SSAS プロキシを削除できます。

アクセスを取り消してプロキシを削除するには

1. サブシステムのアクセスを取り消します。

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='REVOKE_SUBSYSTEM_ACCESS',@proxy_name='SSAS_Proxy',@proxy_subsystem='SSAS'
GO
```

2. プロキシに対する許可を取り消します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_revoke_login_from_proxy
 @proxy_name=N'SSAS_Proxy',@name=N'mydomain\user_name'
GO
```

3. プロキシを削除します。

```
USE [msdb]
GO
EXEC dbo.sp_delete_proxy @proxy_name = N'SSAS_Proxy'
GO
```

SSAS データベースのバックアップ

SSAS データベースのバックアップファイルは、DB インスタンスの D:\S3 フォルダにのみ作成できます。バックアップファイルを S3 バケットに移動するには、Amazon S3 を使用します。

SSAS データベースをバックアップするには、次のようにします。

- 特定のデータベースに対する admin ロールを持つドメインユーザーは、SSMS を使用してデータベースを D:\S3 フォルダにバックアップできます。

  詳細については、「データベース管理者としてのドメインユーザーの追加」を参照してください。
- 次のストアドプロシージャを使用できます。このストアドプロシージャは、暗号化をサポートしていません。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSAS_BACKUP_DB',
@database_name='myssasdb',
@file_path='D:\S3\ssas_db_backup.abf',
[@ssas_apply_compression=1],
[@ssas_overwrite_file=1];
```

  以下のパラメータは必須です。

  - @task_type - MSBI タスクのタイプ (この例では SSAS_BACKUP_DB)。
  - @database_name - バックアップする SSAS データベースの名前。
  - @file_path - SSAS バックアップファイルのパス。.abf 拡張子は必須です。

  以下のパラメータはオプションです。

  - @ssas_apply_compression - SSAS バックアップを圧縮するかどうか。有効な値は 1 (はい) と 0 (いいえ) です。
  - @ssas_overwrite_file - SSAS バックアップファイルを上書きするかどうか。有効な値は 1 (はい) と 0 (いいえ) です。

SSAS データベースの復元

SSAS データベースをバックアップから復元するには、次のストアドプロシージャを使用します。

同じ名前の既存の SSAS データベースがある場合は、データベースを復元できません。復元用のストアドプロシージャは、暗号化されたバックアップファイルをサポートしていません。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSAS_RESTORE_DB',
@database_name='mynewssasdb',
@file_path='D:\S3\ssas_db_backup.abf';
```

以下のパラメータは必須です。

- `@task_type` - MSBI タスクのタイプ (この例では `SSAS_RESTORE_DB`)。
- `@database_name` - 復元先の新しい SSAS データベースの名前。
- `@file_path` - SSAS バックアップファイルへのパス。

特定の時点への DB インスタンスの復元

ポイントインタイムリカバリ (PITR) は SSAS データベースには適用されません。PITR を実行すると、復元されたインスタンスでは、リクエストした時刻より前の最後のスナップショットの SSAS データのみを使用できます。

復元された DB インスタンスで最新の SSAS データベースを使用するには

1. SSAS データベースを出典インスタンスの `D:\S3` フォルダにバックアップします。
2. バックアップファイルを S3 バケットに転送します。
3. S3 バケットから復元されたインスタンスの `D:\S3` フォルダにバックアップファイルを転送します。
4. 復元されたインスタンスに SSAS データベースを復元するためのストアドプロシージャを実行します。

   SSAS プロジェクトを再処理してデータベースを復元することもできます。

SSAS モードの変更

SSAS の実行モードは、表形式または多次元に変更できます。モードを変更するには、AWS マネジメントコンソール または AWS CLI を使用して、SSAS オプションのオプション設定を変更します。

> **Important**
>
> 一度に使用できる SSAS モードは 1 つだけです。モードを変更する前に、必ずすべての SSAS データベースを削除してください。そうしないと、エラーが発生します。

コンソール

次の Amazon RDS コンソールの手順では、SSAS モードを表形式に変更し、MAX_MEMORY パラメータを 70% にします。

SSAS オプションを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. SSAS オプションで、変更するオプショングループ (前の例では`ssas-se-2017`) を選択します。
4. [Modify option] (オプションの変更) を選択します。
5. オプション設定を次のように変更します。
   a. [Max memory] (最大メモリ) に、**`70`** を入力します。
   b. [Mode] (モード) で、[Tabular] (表形式) を選択します。
6. [Modify option] (オプションの変更) を選択します。

AWS CLI

次の AWS CLI の例では、SSAS モードを表形式に変更し、MAX_MEMORY パラメータを 70% に設定します。

CLI コマンドが機能するためには、変更していない場合でも、必要なパラメータをすべて含めてください。

SSAS オプションを変更するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name ssas-se-2017 \
```

```
    --options
 "OptionName=SSAS,VpcSecurityGroupMemberships=sg-12345e67,OptionSettings=[{Name=MAX_MEMORY,
{Name=MODE,Value=Tabular}]" \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name ssas-se-2017 ^
    --options
 OptionName=SSAS,VpcSecurityGroupMemberships=sg-12345e67,OptionSettings=[{Name=MAX_MEMORY,V
{Name=MODE,Value=Tabular}] ^
    --apply-immediately
```

SSAS をオフにする

SSAS をオフにするには、オプショングループから SSAS オプションを削除します。

> **Important**
>
> SSAS オプションを削除する前に、SSAS データベースを削除します。
> SSAS データベースを削除して SSAS オプションを削除する前に、SSAS データベースをバックアップすることを強くお勧めします。

コンソール

SSAS オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. SSAS オプションで、削除するオプショングループ (前の例では `ssas-se-2017`) を選択します。
4. [オプションの削除] を選択します。
5. [オプションの削除] で、[削除するオプション] として [SSAS] を選択します。
6. [すぐに適用] で、オプションをすぐに削除する場合は [はい] を選択し、次のメンテナンスウィンドウで削除する場合は [いいえ] を選択します。

7. [削除] を選択します。

AWS CLI

SSAS オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name ssas-se-2017 \
    --options SSAS \
    --apply-immediately
```

  Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssas-se-2017 ^
    --options SSAS ^
    --apply-immediately
```

SSAS の問題のトラブルシューティング

SSAS の使用時に、次の問題が発生することがあります。

問題	タイプ	トラブルシューティングの推奨事項
SSAS オプションを設定できません。要求された SSAS モードは *new_mode* ですが、現在の DB インスタンスには *number* 個の *current_mode* データベースがあります。*new_mode* モードに切り替える前に既存のデータベースを削除します。データベースを削除するた	RDS イベント	現在のモードを使用する SSAS データベースがまだある場合は、SSAS モードを変更できません。SSAS データベースを削除してから、もう一度試してください。

問題	タイプ	トラブルシューティングの推奨事項
めに *current_mode* モードへのアクセスを取り戻すには、現在の DB オプショングループを更新するか、SSAS オプションの MODE オプション設定値として %s を使用して新しいオプショングループをアタッチします。		
number 個の既存 *mode* のデータベースがあるため、SSAS オプションを削除できません。SSAS オプションは、すべての SSAS データベースが削除されるまで削除できません。SSAS オプションを再度追加し、すべての SSAS データベースを削除して、もう一度試してください。	RDS イベント	SSAS データベースがまだある場合は、SSAS をオフにすることはできません。SSAS データベースを削除してから、もう一度試してください。
SSAS オプションが有効になっていないか、有効化中です。後ほどもう一度試してください。」	RDS ストアドプロシージャ	SSAS ストアドプロシージャは、このオプションがオフになっているときやオンになる途中のときには実行できません。

問題	タイプ	トラブルシューティングの推奨事項
SSAS オプションが正しく設定されていません。オプショングループのメンバーシップステータスが「in-sync」であることを確認し、RDS イベントログで関連する SSAS 設定エラーメッセージを確認します。これらを調べてから、もう一度試してください。エラーが引き続き発生する場合は、AWS サポートにお問い合わせください。	RDS ストアドプロシージャ	オプショングループのメンバーシップが `in-sync` ステータスでないときは、SSAS ストアドプロシージャを実行できません。これにより、SSAS オプションは誤った設定状態になります。 SSAS オプションの変更により、オプショングループのメンバーシップステータスが `failed` に変更された場合、次の 2 つの理由が考えられます。 1. SSAS オプションが、SSAS データベースを削除せずに削除された。 2. SSAS モードが、既存の SSAS データベースを削除せずに、表形式から多次元、または多次元から表形式に更新された。 RDS では、一度に 1 つの SSAS モードのみが可能で、SSAS データベースが存在する場合の SSAS オプションの削除がサポートされていないので、SSAS オプションを再構成します。 RDS イベントログで SSAS インスタンスの設定エラーを確認し、それに応じて問題を解決します。

問題	タイプ	トラブルシューティングの推奨事項
デプロイに失敗しました。変更は、*deployment_file_mode* モードで実行されているサーバーにのみデプロイできます。現在のサーバーモードは *current_mode* です。	RDS ストアドプロシージャ	表形式データベースを多次元サーバーにデプロイしたり、多次元データベースを表形式サーバーにデプロイしたりすることはできません。 正しいモードのファイルを使用していることを確認し、MODE オプション設定が適切な値に設定されていることを確認します。
復元に失敗しました。バックアップファイルは、*restore_file_mode* モードで実行されているサーバー上でのみ復元できます。現在のサーバーモードは *current_mode* です。	RDS ストアドプロシージャ	表形式データベースを多次元サーバーに復元したり、多次元データベースを表形式サーバーに復元することはできません。 正しいモードのファイルを使用していることを確認し、MODE オプション設定が適切な値に設定されていることを確認します。
復元に失敗しました。バックアップファイルと RDS DB インスタンスのバージョンに互換性がありません。	RDS ストアドプロシージャ	SQL Server インスタンスのバージョンと互換性のないバージョンで SSAS データベースを復元することはできません。 詳細については、Microsoft のドキュメントの「テーブルモデルの互換性レベル」と「多次元データベースの互換性レベル」を参照してください。
復元に失敗しました。復元オペレーションで指定されたバックアップファイルが破損しているか、SSAS バックアップファイルではありません。@rds_file_path が正しいフォーマットであることを確認してください。	RDS ストアドプロシージャ	破損したファイルで SSAS データベースを復元することはできません。 ファイルが破損したり、破壊されていたりしていないことを確認してください。 このエラーは、`@rds_file_path` が正しくフォーマットされていない場合 (例えば、`D:\S3\\incorrect_format.abf` のように二重のバックスラッシュがある場合) にも発生します。

問題	タイプ	トラブルシューティングの推奨事項
復元に失敗しました。復元されたデータベース名には、予約語や無効な文字 (. , ; ' ` : / \\ * | ? \" & % $! + = ( ) [ ] { } <>)、または 100 文字を超える文字を含めることはできません。	RDS ストアドプロシージャ	復元されたデータベース名には、予約語や有効でない文字、または 100 文字を超える文字を含めることはできません。 SSAS オブジェクトの命名規則については、Microsoft のドキュメントの「オブジェクトの名前付け規則」を参照してください。
無効なロール名が指定されました。ロール名に予約文字列を含めることはできません。	RDS ストアドプロシージャ	ロール名に予約文字列を含めることはできません。 SSAS オブジェクトの命名規則については、Microsoft のドキュメントの「オブジェクトの名前付け規則」を参照してください。
無効なロール名が指定されました。ロール名には、予約文字 (. , ; ' ` : / \\ * | ? \" & % $! + = ( ) [ ] { } <>) を含めることはできません。	RDS ストアドプロシージャ	ロール名に予約文字を含めることはできません。 SSAS オブジェクトの命名規則については、Microsoft のドキュメントの「オブジェクトの名前付け規則」を参照してください。

Amazon RDS for SQL Server での SQL Server Integration Services のサポート

Microsoft SQL Server Integration Services (SSIS) は、幅広いデータ移行タスクの実行に使用できるコンポーネントです。SSIS は、データ統合およびワークフローアプリケーションに対応したプラットフォームです。データの抽出、変換、ロード (ETL) に使用されるデータウェアハウジングツールを備えています。このツールを使用して、SQL Server データベースのメンテナンスと多次元キューブデータの更新を自動化することもできます。

SSIS プロジェクトはパッケージに編成されて、XML ベースの .dtsx ファイルとして保存されます。パッケージには、制御フローとデータフローを含めることができます。データフローを使用して、ETL オペレーションを表します。デプロイ後、パッケージは SQL Server の SSISDB データベースに保存されます。SSISDB は、完全復旧モードで動作するオンライントランザクション処理 (OLTP) データベースです。

Amazon RDS for SQL Server は RDS DB インスタンスでの SSIS の直接実行をサポートしています。既存または新規の DB インスタンスで SSIS を有効にできます。SSIS はデータベースエンジンと同じ DB インスタンスにインストールされます。

RDS は、次のバージョンで SQL Server スタンダードエディションおよびエンタープライズエディションの SSIS をサポートします。

- SQL Server 2022、すべてのバージョン
- SQL Server 2019、バージョン 15.00.4043.16.v1 以降
- SQL Server 2017、バージョン 14.00.3223.3.v1 以降
- SQL Server 2016、バージョン 13.00.5426.0.v1 以降

> **Note**
>
> Amazon RDS は SSIS for SQL Server 2025 をサポートしていません。サポートは、今後のリリースで予定されています。

目次

- 制限と推奨事項
- SSIS の有効化

- SSIS のオプショングループの作成
- オプショングループへの SSIS オプションの追加
- SSIS のパラメータグループの作成
- SSIS のパラメータの変更
- オプショングループとパラメータグループを DB インスタンスに関連付ける
- S3 統合を有効にする
- SSISDB の管理権限
  - SSIS 用の Windows 認証ユーザーの設定
- SSIS プロジェクトのデプロイ
- デプロイタスクのステータスのモニタリング
- SSIS の使用
  - SSIS プロジェクトのデータベース接続マネージャーの設定
  - SSIS プロキシの作成
  - SQL Server エージェントを使用した SSIS パッケージのスケジュール
  - プロキシからの SSIS アクセスの取り消し
- SSIS データベースの無効化と削除
  - SSIS の無効化
  - SSISDB データベースの削除

制限と推奨事項

RDS for SQL Server で SSIS を実行する場合は、以下の制限と推奨事項が適用されます。

- また、DB インスタンスには、`clr enabled` パラメータが 1 に設定されたパラメータグループが関連付けられている必要があります。詳細については、「SSIS のパラメータの変更」を参照してください。

  > **Note**
  >
  > SQL Server 2017 または 2019 で `clr enabled` パラメータを有効にしている場合、DB インスタンスで共通言語ランタイム (CLR) を使用できません。詳細については、「サポート対象外の機能とサポートが制限されている機能」を参照してください。

- 以下の制御フロータスクがサポートされています。

  - Analysis Services DDL 実行タスク
  - Analysis Services 処理タスク
  - 一括挿入タスク
  - データベース整合性チェックタスク
  - データフロータスク
  - データマイニングクエリタスク
  - データプロファイリングタスク
  - パッケージ実行タスク
  - SQL Server エージェントジョブ実行タスク
  - SQL 実行タスク
  - T-SQL ステートメント実行タスク
  - オペレーター通知タスク
  - インデックス再構築タスク
  - インデックス再編成タスク
  - データベース縮小タスク
  - データベース転送タスク
  - ジョブ転送タスク
  - ログイン転送タスク
  - SQL Server オブジェクト転送タスク
  - 統計更新タスク
- プロジェクトのデプロイのみがサポートされています。
- SQL Server エージェントを使用した SSIS パッケージの実行がサポートされています。
- SSIS ログレコードは、ユーザーが作成したデータベースにのみ挿入できます。
- ファイルの操作には D:\S3 フォルダのみを使用します。他のディレクトリにあるファイルは削除されます。ファイルの場所について、そのほか以下の詳細にも注意してください。
  - SSIS プロジェクトの入力ファイルと出力ファイルは D:\S3 フォルダに配置します。
  - データフロータスクの場合、BLOBTempStoragePath と BufferTempStoragePath の場所を D:\S3 フォルダ内のファイルに変更します。ファイルパスは D:\S3\ で始まる必要があります。
  - ファイル接続に使用されるすべてのパラメータ、可変、表現が D:\S3 フォルダを指していることを確認します。

  - マルチ AZ インスタンスでは、SSIS によって D:\S3 フォルダに作成されたファイルは、フェイルオーバー後に削除されます。詳細については、「S3 統合のマルチ AZ の制限」を参照してください。
  - SSIS によって D:\S3 フォルダに作成されたファイルは、耐久性を高めるために、Amazon S3 バケットにアップロードします。
- 列のインポートと列のエクスポートの変換、およびデータフロータスクのスクリプトコンポーネントはサポートされていません。
- SSIS パッケージの実行時にダンプを有効にしたり、SSIS パッケージにデータタップを追加したりすることはできません。
- SSIS スケールアウト機能はサポートされていません。
- プロジェクトを直接デプロイすることはできません。この機能を実行するための RDS ストアドプロシージャが提供されています。詳細については、「SSIS プロジェクトのデプロイ」を参照してください。
- RDS にデプロイする SSIS プロジェクト (.ispac) ファイルは DoNotSavePasswords 保護モードで構築します。
- SSIS は、リードレプリカを使用する Always On インスタンスではサポートされていません。
- SSIS オプションに関連付けられている SSISDB データベースをバックアップすることはできません。
- SSIS の他のインスタンスからの SSISDB データベースのインポートと復元はサポートされていません。
- 他の SQL Server DB インスタンスまたは Oracle データソースに接続できます。RDS for SQL Server 上の SSIS では、MySQL または PostgreSQL などの他のデータベースエンジンへの接続はサポートされていません。Oracle データソースへの接続に関する詳細については、Oracle OLEDB とリンクされたサーバー を参照してください。
- SSIS は、オンプレミスドメインへの送信の信頼を持つドメイン参加インスタンスをサポートしていません。送信の信頼を使用する場合は、ローカル AWS ドメインのアカウントから SSIS ジョブを実行します。
- ファイルシステムベースのパッケージの実行はサポートされていません。

SSIS の有効化

SSIS を有効にするには、DB インスタンスに SSIS オプションを追加します。以下のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. オプショングループに [SSIS] オプションを追加します。
3. 新しいパラメータグループを作成するか、既存のパラメータグループを選択します。
4. パラメータグループを変更して、`clr enabled` パラメータを 1 に設定します。
5. オプショングループとパラメータグループを DB インスタンスに関連付けます。
6. Amazon S3 統合を有効にする

> **Note**
>
> DB インスタンスに既に SSISDB という名前のデータベースがある場合や、SSIS ログインが予約されている場合、そのインスタンスで SSIS を有効にすることはできません。

SSIS のオプショングループの作成

SSIS を使用するには、使用する DB インスタンスの SQL Server のエディションとバージョンに対応するオプショングループを作成または変更します。そのためには、AWS マネジメントコンソールまたは AWS CLI を使用します。

コンソール

次の手順では、SQL Server Standard Edition 2016 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create subnet group(オプショングループの作成)] ウィンドウで以下を行います。
   a. [名前] に、AWS アカウント内で一意のオプショングループ名 (**`ssis-se-2016`** など) を入力します。名前には、英字、数字、ハイフンのみを使用できます。
   b. [説明] に、オプショングループの簡単な説明 (**`SSIS option group for SQL Server SE 2016`** など) を入力します。この説明は表示用に使用されます。
   c. [エンジン] で [sqlserver-se] を選択します。

d. [メジャーエンジンのバージョン] で、[13.00] を選択します。

5. [作成] を選択します。

CLI

次の手順では、SQL Server Standard Edition 2016 のオプショングループを作成します。

オプショングループを作成するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name ssis-se-2016 \
    --engine-name sqlserver-se \
    --major-engine-version 13.00 \
    --option-group-description "SSIS option group for SQL Server SE 2016"
```

  Windows の場合:

```
aws rds create-option-group ^
    --option-group-name ssis-se-2016 ^
    --engine-name sqlserver-se ^
    --major-engine-version 13.00 ^
    --option-group-description "SSIS option group for SQL Server SE 2016"
```

オプショングループへの SSIS オプションの追加

次に、AWS マネジメントコンソール または AWS CLI を使用して SSIS オプションをオプショングループに追加します。

コンソール

SSIS オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 作成したオプショングループ (この例では ssis-se-2016) を選択します。
4. [オプションの追加] を選択します。
5. [オプションの詳細] で、[オプション名] として [SSRS] を選択します。
6. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
7. [オプションを追加] を選択します。

CLI

SSIS オプションを追加するには

- オプショングループに [SSIS] オプションを追加します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name ssis-se-2016 \
    --options OptionName=SSIS \
    --apply-immediately
```

  Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name ssis-se-2016 ^
    --options OptionName=SSIS ^
    --apply-immediately
```

SSIS のパラメータグループの作成

SSIS で使用する DB インスタンスの SQL Server のエディションとバージョンに対応する `clr enabled` パラメータのパラメータグループを作成または変更します。

コンソール

以下の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [Create parameter group] を選択します。
4. [パラメータグループの作成] ペインで、次の操作を行います。
   a. [パラメータグループファミリー] で、[sqlserver-se-13.0] を選択します。
   b. [グループ名] に、パラメータグループの識別子 (**ssis-sqlserver-se-13** など) を入力します。
   c. [説明] に「**clr enabled parameter group**」と入力します。
5. [作成] を選択します。

CLI

以下の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "clr enabled parameter group"
```

  Windows の場合:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "clr enabled parameter group"
```

SSIS のパラメータの変更

DB インスタンスの SQL Server のエディションとバージョンに対応するパラメータグループの `clr enabled` パラメータを変更します。SSIS の場合、`clr enabled` パラメータを 1 に設定します。

コンソール

以下の手順では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [ssis-sqlserver-se-13] などのパラメータグループを選択します。
4. [パラメータ] で、パラメータのリストを **`clr`** でフィルタ処理します。
5. [clr enabled (clr 有効化)] を選択します。
6. [Edit parameters] を選択します。
7. [Values (値)] から [1] を選択します。
8. [Save changes] (変更の保存) をクリックします。

CLI

以下の手順では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

オプショングループとパラメータグループを DB インスタンスに関連付ける

SSIS オプショングループおよびパラメータグループを DB インスタンスに関連付けるには、AWS マネジメントコンソール または AWS CLI を使用します。

> **Note**
>
> 既存のインスタンスを使用する場合は、このインスタンスに Active Directory ドメインと AWS Identity and Access Management (IAM) ロールが既に関連付けられている必要があります。新しいインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。

コンソール

SSIS の有効化を完了するには、SSIS オプショングループおよびパラメータグループを新規または既存の DB インスタンスに関連付けます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

SSIS オプショングループおよびパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

SSIS オプショングループおよびパラメータグループを使用してインスタンスを作成するには

- オプショングループの作成時に使用したものと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier myssisinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssis-se-2016 \
    --db-parameter-group-name ssis-sqlserver-se-13
```

  Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier myssisinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssis-se-2016 ^
    --db-parameter-group-name ssis-sqlserver-se-13
```

インスタンスを変更し、SSIS オプショングループおよびパラメータグループを関連付けるには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier myssisinstance \
    --option-group-name ssis-se-2016 \
    --db-parameter-group-name ssis-sqlserver-se-13 \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssisinstance ^
    --option-group-name ssis-se-2016 ^
    --db-parameter-group-name ssis-sqlserver-se-13 ^
    --apply-immediately
```

S3 統合を有効にする

SSIS プロジェクト (.ispac) ファイルをデプロイのためにホストにダウンロードするには、S3 ファイル統合を使用します。詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。

SSISDB の管理権限

SSIS オプションを使用してインスタンスを作成または変更すると、その結果、SSISDB データベースのマスターユーザーには、ssis_admin および ssis_logリーダー ロールが付与されます。マスターユーザーには SSISDB に対する以下の権限があります。

- ssis_admin ロールの変更
- ssis_logリーダー ロールの変更
- 任意のユーザーの変更

マスターユーザーは SQL 認証ユーザーであるため、マスターユーザーを使用して SSIS パッケージを実行することはできません。マスターユーザーはこれらの権限を使用して新しい SSISDB ユーザーを作成し、それらのユーザーを ssis_admin および ssis_logリーダー ロールに追加できます。これは、ドメインユーザーに SSIS を使用するアクセス許可を付与するのに役立ちます。

SSIS 用の Windows 認証ユーザーの設定

マスターユーザーは、以下のコード例を使用して、SSISDB に Windows 認証ログインを設定し、必要な手順に対するアクセス許可を付与できます。これにより、ドメインユーザーに SSIS パッケージのデプロイと実行、S3 ファイル転送手順の使用、認証情報の作成、および SQL Server エージェントプロキシの操作を行うアクセス許可が付与されます。詳細については、Microsoft ドキュメントの「Credentials (Database Engine)」および「Create a SQL Server Agent Proxy」を参照してください。

> **Note**
>
> 必要に応じて、Windows 認証ユーザーに以下のアクセス許可の一部またはすべてを付与できます。

Example

```
-- Create a server-level SQL login for the domain user, if it doesn't already exist
USE [master]
GO
CREATE LOGIN [mydomain\user_name] FROM WINDOWS
GO

-- Create a database-level account for the domain user, if it doesn't already exist

USE [SSISDB]
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]

-- Add SSIS role membership to the domain user
ALTER ROLE [ssis_admin] ADD MEMBER [mydomain\user_name]
ALTER ROLE [ssis_logreader] ADD MEMBER [mydomain\user_name]
GO

-- Add MSDB role membership to the domain user
USE [msdb]
```

```
GO
CREATE USER [mydomain\user_name] FOR LOGIN [mydomain\user_name]

-- Grant MSDB stored procedure privileges to the domain user
GRANT EXEC ON msdb.dbo.rds_msbi_task TO [mydomain\user_name] with grant option
GRANT SELECT ON msdb.dbo.rds_fn_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_task_status TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_cancel_task TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_download_from_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_upload_to_s3 TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.rds_delete_from_filesystem TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.rds_gather_file_details TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_add_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_update_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_grant_login_to_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_revoke_login_from_proxy TO [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_delete_proxy TO [mydomain\user_name] with grant option
GRANT EXEC ON msdb.dbo.sp_enum_login_for_proxy to [mydomain\user_name] with grant
 option
GRANT EXEC ON msdb.dbo.sp_enum_proxy_for_subsystem TO [mydomain\user_name]  with grant
 option
GRANT EXEC ON msdb.dbo.rds_sqlagent_proxy TO [mydomain\user_name] WITH GRANT OPTION


-- Add the SQLAgentUserRole privilege to the domain user
USE [msdb]
GO
ALTER ROLE [SQLAgentUserRole] ADD MEMBER [mydomain\user_name]
GO

-- Grant the ALTER ANY CREDENTIAL privilege to the domain user
USE [master]
GO
GRANT ALTER ANY CREDENTIAL TO [mydomain\user_name]
GO
```

SSIS プロジェクトのデプロイ

RDS では、SQL Server Management Studio (SSMS) または SSIS の手順を使用して SSIS プロジェクトを直接デプロイすることはできません。Amazon S3 からプロジェクトファイルをダウンロードしてデプロイするには、RDS ストアドプロシージャを使用します。

ストアドプロシージャを実行するには、ストアドプロシージャの実行アクセス許可を付与した任意のユーザーとしてログインします。詳細については、「SSIS 用の Windows 認証ユーザーの設定」を参照してください。

SSIS プロジェクトをデプロイするには

1. プロジェクト (.ispac) ファイルをダウンロードします。

```
exec msdb.dbo.rds_download_from_s3
@s3_arn_of_file='arn:aws:s3:::bucket_name/ssisproject.ispac',
@rds_file_path='D:\S3\ssisproject.ispac',
@overwrite_file=1;
```

2. 以下のことを確認してから、デプロイタスクを送信します。

   - フォルダが SSIS カタログに存在する。
   - プロジェクト名が、SSIS プロジェクトの開発中に使用したプロジェクト名と一致する。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSIS_DEPLOY_PROJECT',
@folder_name='DEMO',
@project_name='ssisproject',
@file_path='D:\S3\ssisproject.ispac';
```

デプロイタスクのステータスのモニタリング

デプロイタスクのステータスを追跡するには、rds_fn_task_status 関数を呼び出します。2 つのパラメータを使用します。1 つめのパラメータは、SSIS に適用されないため、常に NULL を設定してください。2 つめのパラメータは、タスク ID を受け入れます。

全タスクのリストを見るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータを 0 に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

特定のタスクを受け取るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータをタスク ID に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 機能は次の情報を返します。

出力パラメータ	説明
task_id	タスクの ID。
task_type	SSIS_DEPLOY_PROJECT
database_name	SSIS タスクには該当しません。
% complete	タスクの進行状況の割合。
duration (mins)	タスクにかかった時間 (分単位)。
lifecycle	タスクのステータス。有効な状態には以下のものがあります。 • CREATED - msdb.dbo.rds_msbi_task ストアドプロシージャを呼び出すと、タスクが作成され、ステータスが CREATED に設定されます。 • IN_PROGRESS - タスクがスタートすると、ステータスが に設定されます。IN_PROGRESS ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。 • SUCCESS - タスクが完了すると、ステータスが に設定されます。SUCCESS

出力パラメータ	説明
	• ERROR - タスクが失敗すると、ステータスがに設定されます。ERROR エラーの詳細については、task_info 列を参照してください。 • CANCEL_REQUESTED - rds_cancel_task を呼び出すと、タスクのステータスが CANCEL_REQUESTED になります。 • CANCELLED - タスクが正常にキャンセルされると、タスクのステータスが に設定されます。CANCELLED
task_info	タスクに関する追加情報。処理中にエラーが発生した場合、この列にエラーに関する情報が含まれます。
last_updated	タスクのステータスが最後に更新された日時。
created_at	タスクが作成された日時。
S3_object_arn	SSIS タスクには該当しません。
overwrite_S3_backup_file	SSIS タスクには該当しません。
KMS_master_key_arn	SSIS タスクには該当しません。
filepath	SSIS タスクには該当しません。
overwrite_file	SSIS タスクには該当しません。
task_metadata	SSIS タスクに関連付けられたメタデータ。

SSIS の使用

SSIS プロジェクトを SSIS カタログにデプロイした後、SSMS から直接パッケージを実行するか、SQL Server エージェントを使用してパッケージをスケジュールできます。SSIS パッケージを実行するには、Windows 認証のログインを使用する必要があります。詳細については、「SSIS 用の Windows 認証ユーザーの設定」を参照してください。

トピック

- SSIS プロジェクトのデータベース接続マネージャーの設定
- SSIS プロキシの作成
- SQL Server エージェントを使用した SSIS パッケージのスケジュール
- プロキシからの SSIS アクセスの取り消し

SSIS プロジェクトのデータベース接続マネージャーの設定

接続マネージャーを使用する場合、以下のタイプの認証を使用できます。

- AWS Managed Active Directory を使用したローカルデータベース接続の場合、SQL 認証または Windows 認証を使用できます。Windows 認証の場合、接続文字列のサーバー名として *DB_instance_name*.*fully_qualified_domain_name* を使用します。

  例えば、`myssisinstance.corp-ad.example.com` を使用します。ここで、`myssisinstance` は DB インスタンス名、`corp-ad.example.com` は完全修飾ドメイン名です。
- リモート接続の場合は、常に SQL 認証を使用します。
- セルフマネージド Active Directory を使用したローカルデータベース接続の場合、SQL 認証または Windows 認証を使用できます。Windows 認証の場合、接続文字列のサーバー名として `.` または *`LocalHost`* を使用します。

SSIS プロキシの作成

SQL Server エージェントを使用して SSIS パッケージをスケジュールできるようにするには、SSIS 認証情報と SSIS プロキシを作成します。これらの手順を Windows 認証ユーザーとして実行します。

SSIS 認証情報を作成するには

- プロキシの認証情報を作成します。そのためには、SSMS または以下の SQL ステートメントを使用できます。

```
USE [master]
GO
CREATE CREDENTIAL [SSIS_Credential] WITH IDENTITY = N'mydomain\user_name', SECRET =
 N'mysecret'
GO
```

> **Note**
>
> IDENTITY はドメイン認証ログインであることが必要です。*mysecret* をドメイン認証ログインのパスワードに置き換えます。
> SSISDB プライマリホストが変更されるたびに、SSIS プロキシ認証情報を変更して、新しいホストがそれらのホストにアクセスできるようにします。

SSIS プロキシを作成するには

1. 以下の SQL ステートメントを使用して、プロキシを作成します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_add_proxy
 @proxy_name=N'SSIS_Proxy',@credential_name=N'SSIS_Credential',@description=N''
GO
```

2. 以下の SQL ステートメントを使用して、他のユーザーにプロキシへのアクセスを許可します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_grant_login_to_proxy
 @proxy_name=N'SSIS_Proxy',@login_name=N'mydomain\user_name'
GO
```

3. 以下の SQL ステートメントを使用して、SSIS サブシステムにプロキシへのアクセスを許可します。

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='GRANT_SUBSYSTEM_ACCESS',@proxy_name='SSIS_Proxy',@proxy_subsystem='SSIS'
GO
```

プロキシとそのプロキシに対する許可を表示するには

1. 以下の SQL ステートメントを使用して、プロキシの被付与者を表示します。

```
USE [msdb]
GO
EXEC sp_help_proxy
GO
```

2. 以下の SQL ステートメントを使用して、サブシステムの許可を表示します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_enum_proxy_for_subsystem
GO
```

SQL Server エージェントを使用した SSIS パッケージのスケジュール

認証情報とプロキシを作成し、SSIS にプロキシへのアクセスを許可したら、SQL Server エージェントジョブを作成して SSIS パッケージをスケジュールできます。

SSIS パッケージをスケジュールするには

- SSMS または T-SQL を使用して、SQL Server エージェントジョブを作成できます。以下の例では T-SQL を使用しています。

```
USE [msdb]
GO
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_name=N'MYSSISJob',
@enabled=1,
@notify_level_eventlog=0,
@notify_level_email=2,
```

```
@notify_level_page=2,
@delete_level=0,
@category_name=N'[Uncategorized (Local)]',
@job_id = @jobId OUTPUT
GO
EXEC msdb.dbo.sp_add_jobserver @job_name=N'MYSSISJob',@server_name=N'(local)'
GO
EXEC msdb.dbo.sp_add_jobstep
 @job_name=N'MYSSISJob',@step_name=N'ExecuteSSISPackage',
@step_id=1,
@cmdexec_success_code=0,
@on_success_action=1,
@on_fail_action=2,
@retry_attempts=0,
@retry_interval=0,
@os_run_priority=0,
@subsystem=N'SSIS',
@command=N'/ISSERVER "\"\SSISDB\MySSISFolder\MySSISProject\MySSISPackage.dtsx\"" /
SERVER "\"my-rds-ssis-instance.corp-ad.company.com/\""
/Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";1 /Par
 "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING
 E',
@database_name=N'master',
@flags=0,
@proxy_name=N'SSIS_Proxy'
GO
```

プロキシからの SSIS アクセスの取り消し

以下のストアドプロシージャを使用して、SSIS サブシステムへのアクセスを取り消し、SSIS プロキシを削除できます。

アクセスを取り消してプロキシを削除するには

1. サブシステムのアクセスを取り消します。

```
USE [msdb]
GO
EXEC msdb.dbo.rds_sqlagent_proxy
 @task_type='REVOKE_SUBSYSTEM_ACCESS',@proxy_name='SSIS_Proxy',@proxy_subsystem='SSIS'
GO
```

2. プロキシに対する許可を取り消します。

```
USE [msdb]
GO
EXEC msdb.dbo.sp_revoke_login_from_proxy
 @proxy_name=N'SSIS_Proxy',@name=N'mydomain\user_name'
GO
```

3. プロキシを削除します。

```
USE [msdb]
GO
EXEC dbo.sp_delete_proxy @proxy_name = N'SSIS_Proxy'
GO
```

SSIS データベースの無効化と削除

SSIS データベースを無効化または削除するには、次の手順を実行します。

トピック

- SSIS の無効化
- SSISDB データベースの削除

SSIS の無効化

SSIS を無効にするには、オプショングループから SSIS オプションを削除します。

> **Important**
>
> SSIS オプションを削除しても SSISDB データベースは削除されないため、SSIS プロジェクトを失うことなくこのオプションを安全に削除できます。
> 削除後に SSIS オプションを再度有効にして、前に SSIS カタログにデプロイされていた SSIS プロジェクトを再利用できます。

コンソール

以下の手順では、SSIS オプションを削除します。

SSIS オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. SSIS オプションが含まれているオプショングループ (前の例では ssis-se-2016) を選択します。
4. [オプションの削除] を選択します。
5. [オプションの削除] で、[削除するオプション] として [SSIS] を選択します。
6. [すぐに適用] で、オプションをすぐに削除する場合は [はい] を選択し、次のメンテナンスウィンドウで削除する場合は [いいえ] を選択します。
7. [削除] を選択します。

CLI

以下の手順では、SSIS オプションを削除します。

SSIS オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name ssis-se-2016 \
    --options SSIS \
    --apply-immediately
```

  Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssis-se-2016 ^
    --options SSIS ^
    --apply-immediately
```

SSISDB データベースの削除

SSIS オプションを削除しても、SSISDB データベースは削除されません。SSISDB データベースを削除するには、SSIS オプションを削除した後、`rds_drop_ssis_database` ストアドプロシージャを使用します。

SSIS データベースを削除するには

- 次のストアドプロシージャを使用します。

```
USE [msdb]
GO
EXEC dbo.rds_drop_ssis_database
GO
```

SSISDB データベースを削除した後、SSIS オプションを再度有効にすると、新しい SSISDB カタログが取得されます。

Amazon RDS for SQL Server での SQL Server Reporting Services と Power BI Report Server のサポート

SQL Server 2025 以降、Microsoft はすべてのレポートサービスを Power BI Report Server (PBIRS) に統合し、SSRS に取って代わりました。SSRS で既存の RDS DB インスタンスを SQL Server 2025 にアップグレードすると、SSRS は自動的に PBIRS に置き換えられます。アップグレードプロセスの詳細については、「SSRS から PBIRS へのアップグレード」を参照してください。SSRS と同様に、PBIRS は、レポートと KPI を表示および管理できるウェブポータルを備えたレポートサーバーを提供しています。PBIRS を使用すると、Power BI レポート (.pbix) とページ分割レポート (.rdl) を実行できるため、PBIRS は SSRS 機能のスーパーセットとなっています。SSRS でできることはすべて PBIRS でも実行できます。さらに PBIRS には、インタラクティブな Power BI レポートタイプが追加されています。

Amazon RDS は、次のバージョンで SQL Server スタンダードエディションおよびエンタープライズエディションの PBIRS をサポートしています。

- SQL Server 2025、すべてのバージョン

Microsoft SQL Server Reporting Services (SSRS) は、レポートの生成とディストリビューションに使用されるサーバーベースのアプリケーションです。これは、SQL Server Analysis Services (SSAS) および SQL Server Integration Services (SSIS) を含む SQL Server サービスのスイートの一部です。SSRS は、SQL Server 上に構築されたサービスです。これを使用すると、さまざまなデータソースからデータを収集して、それを簡単に理解でき、分析の準備が整っている方法で提示できます。

Amazon RDS for SQL Server は、RDS DB インスタンスで SSRS と PBIRS を直接実行する操作をサポートしています。SSRS または PBIRS は、既存または新規の DB インスタンスで使用できます。

Amazon RDS は、次のバージョンで SQL Server スタンダードエディションおよびエンタープライズエディションの SSRS をサポートしています。

- SQL Server 2022、すべてのバージョン
- SQL Server 2019、バージョン 15.00.4043.16.v1 以降
- SQL Server 2017、バージョン 14.00.3223.3.v1 以降
- SQL Server 2016、バージョン 13.00.5820.21.v1 以降

目次

- SSRS から PBIRS へのアップグレード
- 制限と推奨事項
- SSRS または PBIRS をオンにする
  - SSRS または PBIRS のオプショングループの作成
  - オプショングループへの SSRS または PBIRS オプションの追加
  - オプショングループと DB インスタンスの関連付け
  - VPC セキュリティグループへのインバウンドアクセスの許可
- レポートサーバーデータベース
- SSRS および PBIRS ログファイル
- SSRS または PBIRS ウェブポータルへのアクセス
  - RDS での SSL の使用
  - ドメインユーザーへのアクセスの許可
  - ウェブポータルへのアクセス
- レポートのデプロイとレポートデータソースの設定
  - SSRS へのレポートのデプロイ
  - レポートのデータソースの設定
- レポートサービスの E メールを使用したレポートの送信
- システムレベルのアクセス許可の取り消し
- タスクのステータスのモニタリング
- SSRS または PBIRS データベースの無効化と削除
  - SSRS または PBIRS をオフにする
  - SSRS または PBIRS データベースの削除

SSRS から PBIRS へのアップグレード

SSRS を実行している既存の RDS DB インスタンスを SQL Server 2025 にアップグレードすると、SSRS は Power BI Report Server (PBIRS) に置き換えられます。移行を完了するには、PBIRS オプションを SQL Server 2025 オプショングループ (新しいオプショングループまたは既存のオプショングループ) に追加し、DB インスタンスに関連付ける必要があります。詳細については、「SSRS または PBIRS をオンにする」を参照してください。この移行は、既存のレポートには影響しません。ページ分割レポート (.rdl) とレポートデータソースは保持され、アップグレードの完了後

も PBIRS ウェブポータルからアクセスできます。ウェブポータルエンドポイントは、アップグレード後も同じままです。レポート URL を参照する既存のブックマークとアプリケーション統合は、変更することなく引き続き機能します。

アップグレード後、SSRS_GRANT_PORTAL_PERMISSION や SSRS_REVOKE_PORTAL_PERMISSION などの SSRS タスクタイプと @ssrs_group_or_username パラメータで rds_msbi_task ストアドプロシージャを使用する自動化処理がある場合は、@pbirs_group_or_username パラメータで同等の PBIRS タスクタイプ PBIRS_GRANT_PORTAL_PERMISSION と PBIRS_REVOKE_PORTAL_PERMISSION を使用するように更新する必要があります。

制限と推奨事項

RDS for SQL Server で SSRS および Power BI Report Server (PBIRS) を実行する場合は、次の制限と推奨事項が適用されます。

- リードレプリカを持つ DB インスタンスではレポートサービスを使用することはできません。
- インスタンスは、レポートサービスウェブポータルおよびウェブサーバー認証にセルフマネージドの Active Directory または AWS Directory Service for Microsoft Active Directory を使用する必要があります。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。
- レポートサービスオプションを使用して作成されたレポートサーバーデータベースをバックアップすることはできません。
- レポートサービスの他のインスタンスからのレポートサーバーデータベースのインポートと復元はサポートされていません。詳細については、「レポートサーバーデータベース」を参照してください。
- デフォルトの SSL ポート (443) でリッスンするようにレポートサービスを構成することはできません。指定できる値は、1234、1434、3260、3343、3389、47001 を除く、1150～49511 です。
- Microsoft Windows ファイル共有によるサブスクリプションはサポートされていません。
- Reporting Services Configuration Manager の使用はサポートされていません。
- ロールの作成と変更はサポートされていません。
- レポートサーバーのプロパティの変更はサポートされていません。
- システム管理者とシステムユーザーのロールは付与されません。
- ウェブポータルを使用してシステムレベルのロールの割り当てを編集することはできません。

SSRS または PBIRS をオンにする

DB インスタンスの SSRS または PBIRS をオンにするには、次のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. SSRS オプション (SQL Server 2016～2022 の場合) または PBIRS オプション (SQL Server 2025 以降の場合) をオプショングループに追加します。
3. オプショングループを DB インスタンスに関連付けます。
4. SSRS または PBIRS リスナーポートの仮想プライベートクラウド (VPC) セキュリティグループへのインバウンドアクセスを許可します。

SSRS または PBIRS のオプショングループの作成

SSRS または PBIRS を使用するには、使用する DB インスタンスの SQL Server エンジンおよびバージョンに対応するオプショングループを作成します。そのためには、AWS マネジメントコンソール または AWS CLI を使用します。

> **Note**
>
> 既存のオプショングループが正しい SQL Server エンジンおよびバージョンに対応している場合は、それを使用することもできます。

コンソール

次の手順では、SQL Server Standard Edition 2017 のオプショングループを作成します。SQL Server 2025 の場合、同じ手順に従いますが、適切なエンジンバージョンを選択し、PBIRS オプションを使用します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group (グループの作成)] を選択します。
4. [オプショングループの作成] ウィンドウで、次の操作を行います。

a. [名前] に、AWS アカウント内の一意のオプショングループ名、**ssrs-se-2017** (SSRS の場合) または**pbirs-se-2025** (PBIRS の場合) などを入力します。名前には、英字、数字、ハイフンのみ使用できます。

b. [説明] に、オプショングループの簡単な説明 (**SSRS option group for SQL Server SE 2017** または **PBIRS option group for SQL Server SE 2025** など) を入力します。この説明は表示用に使用されます。

c. [エンジン] で [sqlserver-se] を選択します。

d. [メジャーエンジンバージョン] で SSRS オプションを追加する場合は 14.00、PBIRS オプションを追加する場合は 17.00 を選択します。

5. [作成] を選択します。

CLI

次の手順では、SQL Server Standard Edition 2017 (SSRS) と SQL Server Standard Edition 2025 (PBIRS) のオプショングループを作成します。

SSRS のオプショングループを作成するには

- 以下のいずれかのコマンドを実行します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name ssrs-se-2017 \
    --engine-name sqlserver-se \
    --major-engine-version 14.00 \
    --option-group-description "SSRS option group for SQL Server SE 2017"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name ssrs-se-2017 ^
    --engine-name sqlserver-se ^
    --major-engine-version 14.00 ^
    --option-group-description "SSRS option group for SQL Server SE 2017"
```

PBIRS のオプショングループを作成するには

- 以下のいずれかのコマンドを実行します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name pbirs-se-2025 \
    --engine-name sqlserver-se \
    --major-engine-version 17.00 \
    --option-group-description "PBIRS option group for SQL Server SE 2025"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name pbirs-se-2025 ^
    --engine-name sqlserver-se ^
    --major-engine-version 17.00 ^
    --option-group-description "PBIRS option group for SQL Server SE 2025"
```

オプショングループへの SSRS または PBIRS オプションの追加

次に、AWS マネジメントコンソールまたは AWS CLI を使用して、SSRS オプション (SQL Server 2016～2022 の場合) または PBIRS オプション (SQL Server 2025 以降の場合) をオプショングループに追加します。

コンソール

SSRS オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 先ほど作成したオプショングループを選択し、[Add option] (オプションの追加) を選択します。
4. [オプションの詳細] で、[オプション名] として SSRS を選択します。
5. [オプション設定] で、次の操作を行います。

a. SSRS サービスがリッスンするポートを入力します。デフォルトは 8443 です。許可される値のリストについては、「制限と推奨事項」を参照してください。

b. [最大メモリ] に値を入力します。

   [最大メモリ] は、レポートサーバーアプリケーションに対して新しいメモリ割り当て要求を許可しない上限しきい値を指定します。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10 ~ 80 です。

c. [セキュリティグループ] で、オプションに関連付ける VPC セキュリティグループを選択します。DB インスタンスに関連付けられているものと同じセキュリティグループを使用します。

6. SSRS Email を使用してレポートを送信するには、[Email delivery in reporting services] (レポートサービスでのメール配信) の [Configure email delivery options] (E メール配信オプションの設定) チェックボックスを選択し、以下の操作を実行します。

   a. [Sender email address] (送信者メールアドレス) で、SSRS Email で送信されるメッセージの [From] フィールドに使用するメールアドレスを入力します。

      SMTP サーバーからメールを送信するアクセス許可を持つユーザーアカウントを指定します。

   b. SMTP server] (SMTP サーバー) で、使用する SMTP サーバーまたはゲートウェイを指定します。

      IP アドレス、社内イントラネット上のコンピュータの NetBIOS 名、または完全修飾ドメイン名を使用できます。

   c. [SMTP port] (SMTP ポート) で、メールサーバーに接続するために使用するポートを入力します。デフォルトは 25 です。

   d. 認証を使用するには:

      i. [Use authentication] (認証の使用) チェックボックスを選択します。

      ii. [Secret Amazon Resource Name (ARN)] (シークレットの Amazon リソースネーム (ARN)) で、ユーザー認証情報の AWS Secrets Manager ARN を入力します。

         次の形式を使用します。

         ```
         arn:aws:secretsmanager:Region:AccountId:secret:SecretName-6RandomChara
         ```

         例:

arn:aws:secretsmanager:*us-west-2*:*123456789012*:secret:*MySecret-a1b2c3*

シークレットを作成するための詳細については、「レポートサービスの E メールを使用したレポートの送信」を参照してください。

e. [Use Secure Sockets Layer (SSL)] (Secure Sockets Layer (SSL) を使用する)) チェックボックスを選択して、SSL を使用して電子メールメッセージを暗号化します。

7. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
8. [オプションを追加] を選択します。

PBIRS オプションを追加するには (SQL Server 2025 以降)

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 先ほど作成した PBIRS のオプショングループを選択し、[オプションの追加] を選択します。
4. [オプションの詳細] で、[オプション名] として [PBIRS] を選択します。
5. [オプション設定] で、次の操作を行います。

   a. PBIRS サービスがリッスンするポートを入力します。デフォルトは 8443 です。許可される値のリストについては、「制限と推奨事項」を参照してください。

   b. [最大メモリ] に値を入力します。

      [最大メモリ] は、レポートサーバーアプリケーションに対して新しいメモリ割り当て要求を許可しない上限しきい値を指定します。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10 ~ 80 です。

   c. [セキュリティグループ] で、オプションに関連付ける VPC セキュリティグループを選択します。DB インスタンスに関連付けられているものと同じセキュリティグループを使用します。

6. レポートサービスの E メールを使用してレポートを送信するには、[E メール配信オプションの設定] の [レポートサービスでの E メール配信] チェックボックスを選択し、以下の操作を実行します。

a. [送信者の E メールアドレス] で、レポートサービスの E メールで送信されるメッセージの [From] フィールドに使用する E メールアドレスを入力します。

   SMTP サーバーからメールを送信するアクセス許可を持つユーザーアカウントを指定します。

b. SMTP server] (SMTP サーバー) で、使用する SMTP サーバーまたはゲートウェイを指定します。

   IP アドレス、社内イントラネット上のコンピュータの NetBIOS 名、または完全修飾ドメイン名を使用できます。

c. [SMTP port] (SMTP ポート) で、メールサーバーに接続するために使用するポートを入力します。デフォルトは 25 です。

d. 認証を使用するには:

   i. [Use authentication] (認証の使用) チェックボックスを選択します。

   ii. [Secret Amazon Resource Name (ARN)] (シークレットの Amazon リソースネーム (ARN)) で、ユーザー認証情報の AWS Secrets Manager ARN を入力します。

   次の形式を使用します。

   arn:aws:secretsmanager:*Region*:*AccountId*:secret:*SecretName*-*6RandomChara*

   例:

   arn:aws:secretsmanager:*us-west-2*:*123456789012*:secret:*MySecret*-*a1b2c3*

   シークレットを作成するための詳細については、「レポートサービスの E メールを使用したレポートの送信」を参照してください。

e. [Use Secure Sockets Layer (SSL)] (Secure Sockets Layer (SSL) を使用する)) チェックボックスを選択して、SSL を使用して電子メールメッセージを暗号化します。

7. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。

8. [オプションを追加] を選択します。

CLI

SSRS オプションを追加するには

1. `ssrs-option.json` などの JSON ファイル を作成します。

   a. 以下の必須パラメータを設定します。

      - `OptionGroupName` - 以前に作成または選択したオプショングループの名前 (次の例では `ssrs-se-2017`)。
      - `Port` - SSRS サービスがリッスンするポート。デフォルトは 8443 です。許可される値のリストについては、「制限と推奨事項」を参照してください。
      - `VpcSecurityGroupMemberships` - RDS DB インスタンスの VPC セキュリティグループのメンバーシップ。
      - `MAX_MEMORY` - レポートサーバーアプリケーションに対して新しいメモリ割り当て要求を許可しない上限しきい値。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10 ~ 80 です。

   b. (オプション) SSRS Email を使用するには、次のパラメータを設定します。

      - `SMTP_ENABLE_EMAIL` — SSRS E メールを使用するには `true` に設定します。デフォルトは `false` です。
      - `SMTP_SENDER_EMAIL_ADDRESS` — SSRS E メールで送信されるメッセージの[From] フィールドに使用するメールアドレスです。SMTP サーバーからメールを送信するアクセス許可を持つユーザーアカウントを指定します。
      - `SMTP_SERVER` — 使用する SMTP サーバーまたはゲートウェイ。IP アドレス、社内イントラネット上のコンピュータの NetBIOS 名、または完全修飾ドメイン名を使用できます。
      - `SMTP_PORT` — メールサーバーに接続するために使用するポート。デフォルトは 25 です。
      - `SMTP_USE_SSL` — SSL を使用して E メールメッセージを暗号化するには、`true` に設定します。デフォルトは `true` です。
      - `SMTP_EMAIL_CREDENTIALS_SECRET_ARN` — ユーザー認証情報を保持する Secrets Manager ARN。次の形式を使用します。

        **arn:aws:secretsmanager:*Region*:*AccountId*:secret:*SecretName*-*6RandomCharact*

シークレットを作成するための詳細については、「レポートサービスの E メールを使用したレポートの送信」を参照してください。

- SMTP_USE_ANONYMOUS_AUTHENTICATION — 認証を使用しない場合は、true に設定し、SMTP_EMAIL_CREDENTIALS_SECRET_ARN を含めないでください。

  SMTP_ENABLE_EMAIL が true の場合、デフォルトは false です。

次の例には、シークレット ARN を使用した SSRS E メールパラメータが含まれています。

```
{
"OptionGroupName": "ssrs-se-2017",
"OptionsToInclude": [
 {
 "OptionName": "SSRS",
 "Port": 8443,
 "VpcSecurityGroupMemberships": ["sg-0abcdef123"],
 "OptionSettings": [
            {"Name": "MAX_MEMORY","Value": "60"},
            {"Name": "SMTP_ENABLE_EMAIL","Value": "true"}
            {"Name": "SMTP_SENDER_EMAIL_ADDRESS","Value": "nobody@example.com"},
            {"Name": "SMTP_SERVER","Value": "email-smtp.us-west-2.amazonaws.com"},
            {"Name": "SMTP_PORT","Value": "25"},
            {"Name": "SMTP_USE_SSL","Value": "true"},
            {"Name": "SMTP_EMAIL_CREDENTIALS_SECRET_ARN","Value":
 "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret-a1b2c3"}
            ]
 }],
"ApplyImmediately": true
}
```

2. オプショングループに [SSRS] オプションを追加します。

   Example

   Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --cli-input-json file://ssrs-option.json \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://ssrs-option.json ^
    --apply-immediately
```

PBIRS オプションを追加するには (SQL Server 2025 以降)

1. pbirs-option.json などの JSON ファイル を作成します。

   a. 以下の必須パラメータを設定します。

      - OptionGroupName - 以前に作成または選択したオプショングループの名前 (次の例では pbirs-se-2025)。
      - Port - PBIRS サービスがリッスンするポート。デフォルトは 8443 です。許可される値のリストについては、「制限と推奨事項」を参照してください。
      - VpcSecurityGroupMemberships - RDS DB インスタンスの VPC セキュリティグループのメンバーシップ。
      - MAX_MEMORY - レポートサーバーアプリケーションに対して新しいメモリ割り当て要求を許可しない上限しきい値。この値は、DB インスタンスの合計メモリに対する割合 (%) です。指定できる値は 10 ~ 80 です。

   b. (オプション) レポートサービスの E メールを使用するには、次のパラメータを設定します。

      - SMTP_ENABLE_EMAIL – レポートサービスの E メールを使用するには、true に設定します。デフォルトは false です。
      - SMTP_SENDER_EMAIL_ADDRESS – レポートサービスの E メールで送信されるメッセージの [From] フィールドに使用する E メールアドレスです。SMTP サーバーからメールを送信するアクセス許可を持つユーザーアカウントを指定します。
      - SMTP_SERVER — 使用する SMTP サーバーまたはゲートウェイ。IP アドレス、社内イントラネット上のコンピュータの NetBIOS 名、または完全修飾ドメイン名を使用できます。
      - SMTP_PORT — メールサーバーに接続するために使用するポート。デフォルトは 25 です。

- SMTP_USE_SSL — SSL を使用して E メールメッセージを暗号化するには、true に設定します。デフォルトは true です。
- SMTP_EMAIL_CREDENTIALS_SECRET_ARN — ユーザー認証情報を保持する Secrets Manager ARN。次の形式を使用します。

  arn:aws:secretsmanager:*Region*:*AccountId*:secret:*SecretName*-*6RandomCharact*

  シークレットを作成するための詳細については、「レポートサービスの E メールを使用したレポートの送信」を参照してください。
- SMTP_USE_ANONYMOUS_AUTHENTICATION — 認証を使用しない場合は、true に設定し、SMTP_EMAIL_CREDENTIALS_SECRET_ARN を含めないでください。

  SMTP_ENABLE_EMAIL が true の場合、デフォルトは false です。

次の例には、シークレット ARN を使用したレポートサービスの E メールパラメータが含まれています。

```
{
"OptionGroupName": "pbirs-se-2025",
"OptionsToInclude": [
 {
 "OptionName": "PBIRS",
 "Port": 8443,
 "VpcSecurityGroupMemberships": ["sg-0abcdef123"],
 "OptionSettings": [
            {"Name": "MAX_MEMORY","Value": "60"},
            {"Name": "SMTP_ENABLE_EMAIL","Value": "true"},
            {"Name": "SMTP_SENDER_EMAIL_ADDRESS","Value": "nobody@example.com"},
            {"Name": "SMTP_SERVER","Value": "email-smtp.us-west-2.amazonaws.com"},
            {"Name": "SMTP_PORT","Value": "25"},
            {"Name": "SMTP_USE_SSL","Value": "true"},
            {"Name": "SMTP_EMAIL_CREDENTIALS_SECRET_ARN","Value":
 "arn:aws:secretsmanager:us-west-2:123456789012:secret:MySecret-a1b2c3"}
            ]
 }],
"ApplyImmediately": true
}
```

2. オプショングループに [PBIRS] オプションを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --cli-input-json file://pbirs-option.json \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://pbirs-option.json ^
    --apply-immediately
```

オプショングループと DB インスタンスの関連付け

AWS マネジメントコンソール または AWS CLI を使用して、オプショングループを DB インスタンスに関連付けます。

> **Note**
>
> 既存の DB インスタンスを使用する場合は、このインスタンスに Active Directory ドメインと AWS Identity and Access Management (IAM) ロールが既に関連付けられている必要があります。新しいインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による Active Directory の操作」を参照してください。

コンソール

オプショングループを新規または既存の DB インスタンスに関連付けることができます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更してから、新しいオプショングループを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

オプショングループを新規または既存の DB インスタンスに関連付けることができます。

オプショングループを使用する DB インスタンスを作成するには

- オプショングループの作成時に使用したものと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier myssrsinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 14.00.3223.3.v1 \
    --allocated-storage 100 \
    --manage-master-user-password  \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name ssrs-se-2017
```

  Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier myssrsinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 14.00.3223.3.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name ssrs-se-2017
```

オプショングループを使用するように DB インスタンスを変更するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier myssrsinstance \
    --option-group-name ssrs-se-2017 \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier myssrsinstance ^
    --option-group-name ssrs-se-2017 ^
    --apply-immediately
```

VPC セキュリティグループへのインバウンドアクセスの許可

DB インスタンスに関連付けられた VPC セキュリティグループへのインバウンドアクセスを許可するには、指定された SSRS または PBIRS リスナーポートのインバウンドルールを作成します。セキュリティグループの設定の詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

レポートサーバーデータベース

DB インスタンスが SSRS オプションに関連付けられている場合、DB インスタンスに 2 つの新しいデータベースが作成されます。

- `rdsadmin_ReportServer`
- `rdsadmin_ReportServerTempDB`

これらのデータベースは、ReportServer および ReportServerTempDB データベースとして機能します。SSRS および PBIRS は、ReportServer データベースにデータを保存し、ReportServerTempDB データベースにそのデータをキャッシュします。詳細については、Microsoft ドキュメントの「レポートサーバーデータベース」を参照してください。

RDS はこれらのデータベースを所有および管理するため、ALTER や DROP などのデータベース操作は許可されません。rdsadmin_ReportServerTempDB データベースへのアクセスは許可されていません。ただし、rdsadmin_ReportServer データベースに対して読み取りオペレーションを実行することはできます。

SSRS および PBIRS ログファイル

SSRS および PBIRS ログファイルをリスト、表示、ダウンロードできます。ログファイルは ReportServerService_*timestamp*.log の命名規則に従っており、D:\rdsdbdata\Log\SSRS ディレクトリにあります。(D:\rdsdbdata\Log ディレクトリは、エラーログと SQL Server エージェントログの親ディレクトリでもあります。)

既存の SSRS または PBIRS インスタンスの場合、レポートサーバーログにアクセスするためにサービスを再起動する必要がある場合があります。SSRS オプションを更新して SSRS サービスを再起動するか、PBIRS オプションを更新して PBIRS サービスを再起動できます。

詳細については、「Amazon RDS for Microsoft SQL Server ログの使用方法」を参照してください。

SSRS または PBIRS ウェブポータルへのアクセス

SSRS または PBIRS ウェブポータルにアクセスするには、次のプロセスを使用します。

1. Secure Sockets Layer (SSL) をオンにします。
2. ドメインユーザーへのアクセス権を付与します。
3. ブラウザとドメインユーザーの認証情報を使用してウェブポータルにアクセスします。

RDS での SSL の使用

SSRS は、接続に HTTPS SSL プロトコルを使用します。このプロトコルを使用するには、クライアントコンピュータの Microsoft Windows オペレーティングシステムに SSL 証明書をインポートします。

SSL 証明書の詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。SQL Server での SSL の使用の詳細については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。

ドメインユーザーへのアクセスの許可

新しい SSRS または PBIRS アクティベーションでは、ロールの割り当てはありません。ドメインユーザーまたはユーザーグループにウェブポータルへのアクセス権を付与するために、RDS にはストアドプロシージャが用意されています。

SSRS ウェブポータルでドメインユーザーにアクセス権を付与するには

- 次のストアドプロシージャを使用します。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSRS_GRANT_PORTAL_PERMISSION',
@ssrs_group_or_username=N'AD_domain\user';
```

PBIRS ウェブポータル (SQL Server 2025 以降) でドメインユーザーにアクセス権を付与するには

- 次のストアドプロシージャを使用します。

```
exec msdb.dbo.rds_msbi_task
@task_type='PBIRS_GRANT_PORTAL_PERMISSION',
@pbirs_group_or_username=N'AD_domain\user';
```

ドメインユーザーまたはユーザーグループには、SSRS の RDS_SSRS_ROLE システムロール、または PBIRS の同等のロールが付与されます。このロールには、次のシステムレベルのタスクが付与されています。

- レポートを実行
- ジョブの管理
- 共有スケジュールの管理
- 共有スケジュールの表示

ルートフォルダでの Content Manager のアイテムレベルのロールも付与されます。

ウェブポータルへのアクセス

SSRS_GRANT_PORTAL_PERMISSION または PBIRS_GRANT_PORTAL_PERMISSION タスクが正常に終了すると、ウェブブラウザを使用してポータルにアクセスできるようになります。ウェブポータル URL 形式とエンドポイントは、SSRS と PBIRS の両方で同じです。

```
https://rds_endpoint:port/Reports
```

この形式では、以下が適用されます。

- *rds_endpoint* – RDS DB インスタンスのエンドポイント。エンドポイントは、DB インスタンスの [接続とセキュリティ] タブにあります。詳細については、「SQL Server DB インスタンスへの接続」を参照してください。
- *port* - SSRS または PBIRS オプションで設定したリスナーポート。

ウェブポータルにアクセスするには

1. ブラウザにウェブポータルの URL を入力します。例えば、次のようになります。

   ```
   https://myssrsinstance.cg034itsfake.us-east-1.rds.amazonaws.com:8443/Reports
   ```

2. SSRS_GRANT_PORTAL_PERMISSION タスク (SSRS の場合) または PBIRS_GRANT_PORTAL_PERMISSION タスク (PBIRS の場合) でアクセス権を付与したドメインユーザーの認証情報を使用してログインします。

レポートのデプロイとレポートデータソースの設定

SSRS にレポートをデプロイし、レポートデータソースを設定するには、次の手順に従います。

トピック

- SSRS へのレポートのデプロイ
- レポートのデータソースの設定

SSRS へのレポートのデプロイ

ウェブポータルにアクセスした後は、ウェブポータルにレポートをデプロイできます。ウェブポータルのアップロードツールを使用して、レポートをアップロードしたり、SQL Server データツール (SSDT) から直接デプロイしたりできます。SSDT からデプロイする場合は、次の点を確認してください。

- SSDT を起動したユーザーは、SSRS ウェブポータルにアクセスできます。

- SSRS プロジェクトプロパティの TargetServerURL の値は、ReportServer サフィックスが付加された RDS DB インスタンスの HTTPS エンドポイントに設定されます。次に例を示します。

```
https://myssrsinstance.cg034itsfake.us-east-1.rds.amazonaws.com:8443/ReportServer
```

レポートのデータソースの設定

レポートを SSRS または PBIRS にデプロイした後、レポートのデータソースを設定する必要があります。レポートのデータソースを設定する際は、次の点を確認してください。

- RDS for SQL Server DB インスタンスが AWS Directory Service for Microsoft Active Directory に結合されている場合、接続文字列のデータソース名として完全修飾ドメイン名 (FQDN) を使用してください。例えば、*myssrsinstance.corp-ad.example.com* を使用します。ここで、*myssrsinstance* は DB インスタンス名、*corp-ad.example.com* は完全修飾ドメイン名です。
- 自己管理型の Active Directory に結合された RDS for SQL Server DB インスタンスの場合、接続文字列のデータソース名として . または *LocalHost* を使用します。

レポートサービスの E メールを使用したレポートの送信

SSRS と PBIRS には、ユーザーにレポートを送信するために使用できる E メール拡張機能が含まれています。

SSRS で E メールを設定するには、SSRS オプション設定を使用します。PBIRS で E メールを設定するには、PBIRS オプション設定を使用します。詳細については、「オプショングループへの SSRS または PBIRS オプションの追加」を参照してください。

E メールを設定すると、レポートサーバー上のレポートをサブスクライブできます。詳細については、Microsoft ドキュメントの「レポートサービスでの E メール配信」を参照してください。

レポートサービスの E メールが RDS で機能するためには、AWS Secrets Manager との統合が必要です。Secrets Manager と統合するには、シークレットを作成します。

> **Note**
>
> シークレットを後で変更する場合は、オプショングループで SSRS または PBIRS オプションも更新する必要があります。

レポートサービスの E メール用にシークレットを作成するには

1. AWS Secrets Manager ユーザーガイドにある「シークレットを作成する」のステップに従います。

   a. [Select secret type] (シークレットタイプの選択) で、[Other type of secrets] (他の種類のシークレット) を選択します。

   b. [Key/value pairs] (キー/値ペア) で、次のように入力します。

      - **SMTP_USERNAME** — SMTP サーバーからメールを送信するアクセス許可を持つユーザーを入力します。
      - **SMTP_PASSWORD** — SMTP ユーザーのパスワードを入力します。

   c. 暗号化キーでは、デフォルトの AWS KMS key を使用しないでください。独自の既存のキーを使用するか、新しく作成します。

      KMS キーポリシーでは、`kms:Decrypt` アクションを許可する必要があります。例:

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "rds.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

2. AWS Secrets Managerユーザーガイドの「アクセス許可ポリシーをシークレットにアタッチする」のステップに従います。アクセス許可ポリシーは、`rds.amazonaws.com` サービスプリンシパルに `secretsmanager:GetSecretValue` アクションを与えます。

   Confused Deputy Problem (混乱した代理の問題) を回避するために、ポリシーの `aws:sourceAccount` および `aws:sourceArn` 条件を使用することをお勧めします。`aws:sourceAccount` には AWS アカウント を使用し、`aws:sourceArn` にはオプショングループ ARN を使用します。(詳しくは、「サービス間での混乱した代理問題の防止」を参照してください。)

以下の例に示しているのは、アクセス許可ポリシーです。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "Service" : "rds.amazonaws.com"
    },
    "Action" : "secretsmanager:GetSecretValue",
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "aws:sourceAccount" : "123456789012"
      },
      "ArnLike" : {
        "aws:sourceArn" : "arn:aws:rds:us-west-2:123456789012:og:ssrs-
se-2017"
      }
    }
  } ]
}
```

その他の例については、「AWS Secrets Manager ユーザーガイド」の「AWS Secrets Manager のアクセス許可ポリシーの例」を参照してください。

システムレベルのアクセス許可の取り消し

RDS_SSRS_ROLE システムロールには、システムレベルのロールの割り当てを削除するための十分なアクセス許可がありません。RDS_SSRS_ROLE からユーザーまたはユーザーグループを削除するには、ロールの付与に使用したものと同じストアドプロシージャを使用します。ただし、SSRS_REVOKE_PORTAL_PERMISSION タスクタイプを使用します。

SSRS ウェブポータルのドメインユーザーからアクセス権を取り消すには

- 次のストアドプロシージャを使用します。

```
exec msdb.dbo.rds_msbi_task
@task_type='SSRS_REVOKE_PORTAL_PERMISSION',
@ssrs_group_or_username=N'AD_domain\user';
```

PBIRS ウェブポータル (SQL Server 2025 以降) のドメインユーザーからアクセス権を取り消すには

- 次のストアドプロシージャを使用します。

```
exec msdb.dbo.rds_msbi_task
@task_type='PBIRS_REVOKE_PORTAL_PERMISSION',
@pbirs_group_or_username=N'AD_domain\user';
```

これにより、RDS_SSRS_ROLE システムロール (SSRS の場合) または同等の PBIRS ロールからユーザーが削除されます。また、ユーザーが Content Manager アイテムレベルのロールを持っている場合、このロールからも削除されます。

タスクのステータスのモニタリング

タスクの権限付与または取り消しのステータスを追跡するには、rds_fn_task_status 関数を呼び出します。2 つのパラメータを使用します。1 つめのパラメータは、SSRS に適用されないため、常に NULL を設定してください。2 つめのパラメータは、タスク ID を受け入れます。

全タスクのリストを見るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータを 0 に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,0);
```

特定のタスクを受け取るには、以下の例にあるように、初期のパラメータを NULL に設定し、2 つめのパラメータをタスク ID に設定します。

```
SELECT * FROM msdb.dbo.rds_fn_task_status(NULL,42);
```

rds_fn_task_status 機能は次の情報を返します。

出力パラメータ	説明
task_id	タスクの ID。

出力パラメータ	説明
task_type	SSRS では、次のタスクタイプを使用できます。 • SSRS_GRANT_PORTAL_PERMISSION • SSRS_REVOKE_PORTAL_PERMISSION PBIRS では、次のタスクタイプを使用できます。 • PBIRS_GRANT_PORTAL_PERMISSION • PBIRS_REVOKE_PORTAL_PERMISSION
database_name	SSRS または PBIRS タスクは該当しません。
% complete	タスクの進行状況の割合。
duration (mins)	タスクにかかった時間 (分単位)。

出力パラメータ	説明
lifecycle	タスクのステータス。有効な状態には以下のものがあります。 • CREATED - SSRS ストアドプロシージャのいずれかを呼び出すと、タスクが作成され、ステータスが に設定されます。CREATED • IN_PROGRESS - タスクがスタートすると、ステータスが に設定されます。IN_PROGRESS ステータスが CREATED から IN_PROGRESS に変わるまで、最大 5 分かかることがあります。 • SUCCESS - タスクが完了すると、ステータスが に設定されます。SUCCESS • ERROR - タスクが失敗すると、ステータスがに設定されます。ERRORエラーの詳細については、task_info 列を参照してください。 • CANCEL_REQUESTED - rds_cancel_task ストアドプロシージャを呼び出すと、タスクのステータスが CANCEL_REQUESTED に設定されます。 • CANCELLED - タスクが正常にキャンセルされると、タスクのステータスが に設定されます。CANCELLED
task_info	タスクに関する追加情報。処理中にエラーが発生した場合、この列にエラーに関する情報が含まれます。
last_updated	タスクのステータスが最後に更新された日時。

出力パラメータ	説明
created_at	タスクが作成された日時。
S3_object_arn	SSRS または PBIRS タスクは該当しません。
overwrite_S3_backup_file	SSRS または PBIRS タスクは該当しません。
KMS_master_key_arn	SSRS または PBIRS タスクは該当しません。
filepath	SSRS または PBIRS タスクは該当しません。
overwrite_file	SSRS または PBIRS タスクは該当しません。
task_metadata	SSRS または PBIRS タスクに関連付けられたメタデータ。

SSRS または PBIRS データベースの無効化と削除

SSRS または PBIRS を無効にし、データベースを削除するには、次の手順に従います。

トピック

- SSRS または PBIRS をオフにする
- SSRS または PBIRS データベースの削除

SSRS または PBIRS をオフにする

SSRS をオフにするには、オプショングループから SSRS オプションを削除します。PBIRS をオフにするには、オプショングループから PBIRS オプションを削除します。 オプションを削除しても、レポートサーバーデータベースは削除されません。詳細については、「SSRS または PBIRS データベースの削除」を参照してください。

SSRS を再度オンにするには、SSRS オプションを追加するか、PBIRS の場合は PBIRS オプションを再度追加します。さらに、レポートサーバーデータベースも削除した場合は、同じ DB インスタンスでオプションを再度有効にすると、新しいレポートサーバーデータベースが作成されます。

コンソール

SSRS オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. SSRS オプションが含まれているオプショングループ (前の例では `ssrs-se-2017`) を選択します。
4. [オプションの削除] を選択します。
5. [オプションの削除] で、[削除するオプション] として [SSRS] を選択します。
6. [すぐに適用] で、オプションをすぐに削除する場合は [はい] を選択し、次のメンテナンスウィンドウで削除する場合は [いいえ] を選択します。
7. [削除] を選択します。

PBIRS オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. PBIRS オプションが含まれているオプショングループ (前の例では `pbirs-se-2025`) を選択します。
4. [オプションを削除] を選択します。
5. [削除オプション] で、[削除するオプション] として [PBIRS] を選択します。
6. [すぐに適用] で、オプションをすぐに削除する場合は [はい] を選択し、次のメンテナンスウィンドウで削除する場合は [いいえ] を選択します。
7. [削除] を選択します。

CLI

SSRS オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを実行します。

Example

Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name ssrs-se-2017 \
    --options SSRS \
    --apply-immediately
```

Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name ssrs-se-2017 ^
    --options SSRS ^
    --apply-immediately
```

PBIRS オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを実行します。

Example

Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name pbirs-se-2025 \
    --options PBIRS \
    --apply-immediately
```

Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name pbirs-se-2025 ^
    --options PBIRS ^
    --apply-immediately
```

SSRS または PBIRS データベースの削除

SSRS または PBIRS オプションを削除しても、レポートサーバーデータベースは削除されません。それらを削除するには、次のストアドプロシージャを使用します。

レポートサーバーデータベースを削除するには、必ず最初に SSRS または PBIRS オプションを削除してください。

SSRS または PBIRS データベースを削除するには

- 次のストアドプロシージャを使用します。このプロシージャでは、SSRS データベースと PBIRS データベースの両方を処理します。

```
exec msdb.dbo.rds_drop_ssrs_databases
```

RDS for SQL Server での Microsoft 分散トランザクションコーディネーターのサポート

分散トランザクションは、2 つ以上のネットワークホストが関与するデータベーストランザクションです。RDS for SQL Server では、ホスト間の分散トランザクションをサポートしています。この場合、ホストの 1 つとして次のいずれかを含みます。

- RDS for SQL Server の DB インスタンス
- オンプレミスの SQL Server ホスト
- SQL Server がインストールされている Amazon EC2 ホスト
- 分散トランザクションをサポートするデータベースエンジンを備えた他の EC2 ホストまたは RDS DB インスタンス

RDS では、SQL Server 2012 以降 (バージョン 11.00.5058.0.v1 以降)、RDS for SQL Server のすべてのエディションで分散トランザクションがサポートされています。サポートは、Microsoft 分散トランザクションコーディネーター (MSDTC) を使用して提供されます。MSDTC の詳細については、Microsoft のドキュメントの「分散トランザクションコーディネーター」を参照してください。

目次

- 制限事項
- MSDTC の有効化
  - MSDTC のオプショングループの作成
  - オプショングループへの MSDTC オプションの追加
  - MSDTC のパラメータグループの作成
  - MSDTC のパラメータの変更
  - DB インスタンスとのオプショングループおよびパラメータグループの関連付け
  - MSDTC オプションの変更
- トランザクションの使用
  - 分散トランザクションの使用
  - XA トランザクションの使用
  - トランザクションの追跡の使用
- MSDTC の無効化
- RDS for SQL Server の MSDTC のトラブルシューティング

制限事項

RDS for SQL Server で MSDTC を使用する場合は、次の制限が適用されます。

- MSDTC は、SQL Server データベースミラーリングを使用するインスタンスではサポートされません。詳細については、「トランザクション - 可用性グループとデータベースミラーリング」を参照してください。
- `in-doubt xact resolution` パラメータは 1 または 2 に設定する必要があります。詳細については、「MSDTC のパラメータの変更」を参照してください。
- MSDTC では、分散トランザクションに参加するすべてのホストをホスト名を使用して解決可能にする必要があります。RDS は、ドメインに参加しているインスタンスに対してこの機能を自動的に維持します。ただし、スタンドアロンインスタンスの場合は、必ず DNS サーバーを手動で設定してください。
- Java Database Connectivity (JDBC) XA トランザクションは、SQL Server 2017 バージョン 14.00.3223.3 以降、SQL Server 2019、SQL Server 2022、および SQL Server 2025 でサポートされています。
- RDS インスタンス上のクライアント動的リンクライブラリ (DLL) に依存する分散トランザクションはサポートされていません。
- カスタム XA 動的リンクライブラリの使用はサポートされていません。

MSDTC の有効化

DB インスタンスに対して MSDTC を有効にするには、次のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. オプショングループに [MSDTC] オプションを追加します。
3. 新しいパラメータグループを作成するか、既存のパラメータグループを選択します。
4. パラメータグループを変更して、`in-doubt xact resolution` パラメータを 1 または 2 に設定します。
5. オプショングループとパラメータグループを DB インスタンスに関連付けます。

MSDTC のオプショングループの作成

AWS マネジメントコンソール または AWS CLI を使用して、使用する DB インスタンスの SQL Server エンジンおよびバージョンに対応するオプショングループを作成します。

> **Note**
>
> 既存のオプショングループが正しい SQL Server エンジンおよびバージョンに対応している場合は、それを使用することもできます。

コンソール

次の手順では、SQL Server Standard Edition 2016 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [オプショングループの作成] ウィンドウで、次の操作を行います。
   a. [名前] に、AWS アカウント内で一意のオプショングループ名 (**msdtc-se-2016** など) を入力します。名前には、英字、数字、ハイフンのみを使用できます。
   b. [説明] に、オプショングループの簡単な説明 (**MSDTC option group for SQL Server SE 2016** など) を入力します。この説明は表示用に使用されます。
   c. [エンジン] で [sqlserver-se] を選択します。
   d. [メジャーエンジンのバージョン] で、[13.00] を選択します。
5. [Create] (作成) を選択します。

CLI

次の例では、SQL Server Standard Edition 2016 のオプショングループを作成します。

オプショングループを作成するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-option-group \
    --option-group-name msdtc-se-2016 \
    --engine-name sqlserver-se \
    --major-engine-version 13.00 \
    --option-group-description "MSDTC option group for SQL Server SE 2016"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name msdtc-se-2016 ^
    --engine-name sqlserver-se ^
    --major-engine-version 13.00 ^
    --option-group-description "MSDTC option group for SQL Server SE 2016"
```

オプショングループへの MSDTC オプションの追加

次に、AWS マネジメントコンソール または AWS CLI を使用して MSDTC オプションをオプショングループに追加します。

次のオプション設定が必要です。

- ポート - MSDTC にアクセスするために使用するポート。指定できる値は、1234、1434、3260、3343、3389、47001 を除く、1150～49151 です。デフォルト値は 5000 です。

  使用するポートがファイアウォールルールで有効になっていることを確認します。また、必要に応じて、DB インスタンスに関連付けられているセキュリティグループのインバウンドルールとアウトバウンドルールでこのポートが有効になっていることを確認します。(詳しくは、「Amazon RDS DB インスタンスに接続できない」を参照してください。)
- セキュリティグループ - RDS DB インスタンスの VPC セキュリティグループのメンバーシップ。
- 認証タイプ - ホスト間の認証モード。次の認証タイプがサポートされています。
  - 相互 - RDS インスタンスは、統合認証を使用して相互に認証されます。このオプションを選択した場合、このオプショングループに関連付けられているすべてのインスタンスがドメインに参加している必要があります。
  - なし - ホスト間での認証は実行されません。本稼働環境でこのモードを使用することは推奨されていません。

- トランザクションログサイズ - MSDTC トランザクションログのサイズ。許容値は、4 ~ 1024 MB です。デフォルトサイズは 4 MB です。

次のオプション設定はオプションです。

- [インバウンド接続を有効にする] - このオプショングループに関連付けられているインスタンスへのインバウンド MSDTC 接続を許可するかどうかを指定します。
- [アウトバウンド接続を有効にする] - このオプショングループに関連付けられているインスタンスからのアウトバウンド MSDTC 接続を許可するかどうかを指定します。
- XA を有効にする X - XA トランザクションを許可するかどうかを指定します。XA プロトコルの詳細については、「XA 仕様」を参照してください。
- SNA LU を有効にする - SNA LU プロトコルを分散トランザクションに使用できるようにするかどうかを指定します。SNA LU プロトコルのサポートの詳細については、Microsoft ドキュメントの「Managing IBM CICS LU 6.2 transactions」を参照してください。

コンソール

MSDTC オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 先ほど作成したオプショングループを選択します。
4. [オプションの追加] を選択します。
5. [オプションの詳細] で、[オプション名] として MSDTC を選択します。
6. [オプション設定] で、次の操作を行います。
    a. [ポート] に、MSDTC にアクセスするためのポート番号を入力します。デフォルトは 5000 です。
    b. [セキュリティグループ] で、オプションに関連付ける VPC セキュリティグループを選択します。
    c. [認証タイプ] で [相互] または [なし] を選択します。
    d. [トランザクションログのサイズ] に、4 ~ 24 の値を入力します。デフォルト値は 4 です。
7. [追加設定] で、次の操作を行います。

a. [接続] で、必要に応じて [インバウンド接続を有効にする] と [アウトバウンド接続を有効にする] を選択します。

b. [許可されたプロトコル] で、必要に応じて [XA を有効にする] と [SNA LU を有効にする] を選択します。

8. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。

9. [オプションの追加] を選択します。

このオプションを追加するにあたって再起動は必要ありません。

CLI

MSDTC オプションを追加するには

1. 次の必須のパラメータを使用して、JSON ファイル (msdtc-option.json など) を作成します。

```
{
"OptionGroupName":"msdtc-se-2016",
"OptionsToInclude": [
 {
 "OptionName":"MSDTC",
 "Port":5000,
 "VpcSecurityGroupMemberships":["sg-0abcdef123"],
 "OptionSettings":[{"Name":"AUTHENTICATION","Value":"MUTUAL"},
{"Name":"TRANSACTION_LOG_SIZE","Value":"4"}]
 }],
"ApplyImmediately": true
}
```

2. オプショングループに [MSDTC] オプションを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --cli-input-json file://msdtc-option.json \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --cli-input-json file://msdtc-option.json ^
    --apply-immediately
```

再起動は必要ありません。

MSDTC のパラメータグループの作成

DB インスタンスの SQL Server のエディションとバージョンに対応する `in-doubt xact resolution` パラメータのパラメータグループを作成または変更します。

コンソール

次の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. [パラメータグループの作成] を選択します。
4. [パラメータグループの作成] ペインで、次の操作を行います。
   a. [パラメータグループファミリー] で、[sqlserver-se-13.0] を選択します。
   b. [グループ名] に、パラメータグループの識別子 (**msdtc-sqlserver-se-13** など) を入力します。
   c. [説明] に「**in-doubt xact resolution**」と入力します。
5. [Create] (作成) を選択します。

CLI

次の例では、SQL Server Standard Edition 2016 のパラメータグループを作成します。

パラメータグループを作成するには

- 以下のいずれかのコマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --db-parameter-group-family "sqlserver-se-13.0" \
    --description "in-doubt xact resolution"
```

Windows の場合:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --db-parameter-group-family "sqlserver-se-13.0" ^
    --description "in-doubt xact resolution"
```

MSDTC のパラメータの変更

DB インスタンスの SQL Server のエディションとバージョンに対応するパラメータグループの `in-doubt xact resolution` パラメータを変更します。

MSDTC の場合、`in-doubt xact resolution` パラメータに次のいずれかの値を設定します。

- 1 - `Presume commit`。MSDTC の in-doubt トランザクションは、コミットされたものと見なされます。
- 2 - `Presume abort`。MSDTC の in-doubt トランザクションは、停止されたものと見なされます。

詳細については、Microsoft ドキュメント の「in-doubt xact resolution server configuration option」を参照してください。

コンソール

次の例では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[パラメータグループ] を選択します。
3. msdtc-sqlserver-se-13 などのパラメータグループを選択します。
4. [パラメータ] で、パラメータのリストを **xact** でフィルタ処理します。
5. in-doubt xact resolution を選択します。
6. [Edit parameters] を選択します。
7. **1** または **2** を入力します。
8. [Save changes] (変更の保存) をクリックします。

CLI

次の例では、SQL Server Standard Edition 2016 用に作成したパラメータグループを変更します。

パラメータグループを変更するには

- 以下のいずれかのコマンドを使用します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --parameters "ParameterName='in-doubt xact
 resolution',ParameterValue=1,ApplyMethod=immediate"
```

  Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --parameters "ParameterName='in-doubt xact
 resolution',ParameterValue=1,ApplyMethod=immediate"
```

DB インスタンスとのオプショングループおよびパラメータグループの関連付け

AWS マネジメントコンソール または AWS CLI を使用すると、MSDTC オプショングループおよびパラメータグループを DB インスタンスに関連付けることができます。

コンソール

MSDTC オプショングループおよびパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> ドメイン参加済みの既存の DB インスタンスを使用する場合は、このインスタンスに Active Directory ドメインおよび AWS Identity and Access Management (IAM) ロールが既に関連付けられている必要があります。新しいドメイン参加済みのインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による AWS Managed Active Directory の操作」を参照してください。

CLI

MSDTC オプショングループおよびパラメータグループを新規または既存の DB インスタンスに関連付けることができます。

> **Note**
>
> 既存のドメイン参加済みの DB インスタンスを使用する場合は、このインスタンスに Active Directory ドメインと IAM ロールが既に関連付けられている必要があります。新しいドメイン参加済みのインスタンスを作成する場合は、既存の Active Directory ドメインと IAM ロールを指定します。詳細については、「RDS for SQL Server による AWS Managed Active Directory の操作」を参照してください。

MSDTC オプショングループおよびパラメータグループを使用して DB インスタンスを作成するには

- オプショングループの作成時に使用したものと同じ DB エンジンのタイプとメジャーバージョンを指定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-se \
    --engine-version 13.00.5426.0.v1 \
    --allocated-storage 100 \
    --manage-master-user-password \
    --master-username admin \
    --storage-type gp2 \
    --license-model li \
    --domain-iam-role-name my-directory-iam-role \
    --domain my-domain-id \
    --option-group-name msdtc-se-2016 \
    --db-parameter-group-name msdtc-sqlserver-se-13
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-se ^
    --engine-version 13.00.5426.0.v1 ^
    --allocated-storage 100 ^
    --manage-master-user-password ^
    --master-username admin ^
    --storage-type gp2 ^
    --license-model li ^
    --domain-iam-role-name my-directory-iam-role ^
    --domain my-domain-id ^
    --option-group-name msdtc-se-2016 ^
    --db-parameter-group-name msdtc-sqlserver-se-13
```

DB インスタンスを変更し、MSDTC オプショングループとパラメータグループを関連付けるには

- 以下のいずれかのコマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --option-group-name msdtc-se-2016 \
    --db-parameter-group-name msdtc-sqlserver-se-13 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --option-group-name msdtc-se-2016 ^
    --db-parameter-group-name msdtc-sqlserver-se-13 ^
    --apply-immediately
```

MSDTC オプションの変更

MSDTC オプションを有効にすると、その設定を変更できます。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。

> **Note**
>
> MSDTC オプション設定の一部の変更では、MSDTC サービスを再起動する必要があります。この要件は、実行中の分散トランザクションに影響を与える可能性があります。

トランザクションの使用

分散トランザクションの使用

Amazon RDS for SQL Server では、オンプレミスで実行されている分散トランザクションと同じ方法で分散トランザクションを実行します。

- .NET フレームワークの System.Transactions 昇格可能トランザクションを使用します。これにより、必要になるまで作成を延期することで分散トランザクションが最適化されます。

この場合、昇格は自動的に行われるため、ユーザーが介入する必要はありません。トランザクション内にリソースマネージャーが 1 つのみ存在する場合、昇格は実行されません。暗黙的なトランザクションスコープの詳細については、Microsoft ドキュメントの「トランザクションスコープを使用した暗黙的なトランザクションの実装」を参照してください。

昇格可能なトランザクションは、次の .NET 実装でサポートされています。

- ADO.NET 2.0 以降、`System.Data.SqlClient` では SQL Server での昇格可能トランザクションをサポートしています。詳細については、Microsoft のドキュメントの「SQL Server と System.Transations の統合」を参照してください。
- ODP.NET は `System.Transactions` をサポートしています。ローカルトランザクションは、Oracle Database 11g リリース 1 (バージョン11.1) 以降への `TransactionsScope` スコープで開かれた初期の接続に対して作成されます。2 番目の接続が開かれると、このトランザクションは自動的に分散トランザクションに昇格されます。ODP.NET での分散トランザクションのサポートの詳細については、Oracle ドキュメントの「Microsoft Distributed Transaction Coordinator Integration」を参照してください。

- `BEGIN DISTRIBUTED TRANSACTION` ステートメントを使用します。詳細については、Microsoft のドキュメントの「BEGIN DISTRIBUTED TRANSACTION (Transact-SQL)」を参照してください。

XA トランザクションの使用

RDS for SQL Server 2017 バージョン 14.00.3223.3 以降、JDBC を使用して分散トランザクションを制御できます。`MSDTC` オプションで `Enable XA` オプション設定を`true` に設定すると、RDS では JDBC トランザクションが自動的に有効化され、`SqlJDBCXAUser` ロールが `guest` ユーザーに付与されます。これにより、JDBC を使用した分散トランザクションの実行が可能になります。コード例も含めた詳細については、Microsoft のドキュメントの「XA トランザクションについて」を参照してください。

トランザクションの追跡の使用

RDS では、トラブルシューティングの目的で MSDTC トランザクションの追跡を制御し、RDS DB インスタンスからこれらをダウンロードできます。トランザクションの追跡セッションは、次の RDS ストアドプロシージャを実行することで制御できます。

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'trace_action',
[@traceall='0|1'],
[@traceaborted='0|1'],
```

```
[@tracelong='0|1'];
```

以下のパラメータは必須です。

- `trace_action` - 追跡アクション。START、STOP または STATUS のいずれかを設定できます。

以下のパラメータはオプションです。

- `@traceall` - すべての分散トランザクションを追跡するには、1 に設定します。デフォルトは 0 です。
- `@traceaborted` - キャンセルされた分散トランザクションを追跡するには、1 に設定します。デフォルトは 0 です。
- `@tracelong` - 長時間実行される分散トランザクションを追跡するには、1 に設定します。デフォルトは 0 です。

Example START 追跡アクション例

新しいトランザクション追跡セッションをスタートするには、次のステートメント例を実行します。

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'START',
@traceall='0',
@traceaborted='1',
@tracelong='1';
```

> **Note**
>
> 一度にアクティブにできるトランザクション追跡セッションは 1 つだけです。追跡セッションがアクティブなときに新しい追跡セッションの START コマンドが発行されると、エラーが返され、アクティブな追跡セッションは変更されません。

Example STOP 追跡アクション例

トランザクション追跡セッションを停止するには、次のステートメントを実行します。

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'STOP'
```

このステートメントは、アクティブなトランザクション追跡セッションを停止し、トランザクション追跡データを RDS DB インスタンスのログディレクトリに保存します。出力の初期の行には全体的な結果が含まれ、後続の行はオペレーションの詳細を示します。

追跡セッションが正常に停止した例を次に示します。

```
OK: Trace session has been successfully stopped.
Setting log file to: D:\rdsdbdata\MSDTC\Trace\dtctrace.log
Examining D:\rdsdbdata\MSDTC\Trace\msdtctr.mof for message formats,  8 found.
Searching for TMF files on path: (null)
Logfile D:\rdsdbdata\MSDTC\Trace\dtctrace.log:
 OS version    10.0.14393  (Currently running on 6.2.9200)
 Start Time    <timestamp>
 End Time      <timestamp>
 Timezone is   @tzres.dll,-932 (Bias is 0mins)
 BufferSize            16384 B
 Maximum File Size     10 MB
 Buffers  Written      Not set (Logger may not have been stopped).
 Logger Mode Settings (11000002) ( circular paged
 ProcessorCount         1
Processing completed   Buffers: 1, Events: 3, EventsLost: 0 :: Format Errors: 0,
 Unknowns: 3
Event traces dumped to d:\rdsdbdata\Log\msdtc_<timestamp>.log
```

詳細情報を使用して、生成されたログファイル名のクエリを実行できます。RDS DB インスタンスからのログファイルのダウンロードの詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

追跡セッションログは、インスタンスに 35 日間残されます。古い追跡セッションのログは自動的に削除されます。

Example STATUS 追跡アクション例

トランザクション追跡セッションのステータスを追跡するには、次のステートメントを実行します。

```
exec msdb.dbo.rds_msdtc_transaction_tracing 'STATUS'
```

このステートメントは、結果セットの個別の行として以下を出力します。

```
OK
SessionStatus: <Started|Stopped>
```

```
TraceAll: <True|False>
TraceAborted: <True|False>
TraceLongLived: <True|False>
```

初期の行は、操作の全体的な結果を示します。該当する場合は OK または ERROR と詳細も表示されます。後続の行は、追跡セッションのステータスに関する詳細を示します。

- SessionStatus の値は次のいずれかを指定できます。
  - Started追跡セッションが実行中の場合は 。
  - Stopped実行中の追跡セッションがない場合は 。
- 追跡セッションフラグは、True コマンドでの設定方法によって False または START のどちらかになります。

MSDTC の無効化

MSDTC を無効にするには、MSDTC オプションをそのオプショングループから削除します。

コンソール

MSDTC オプションをそのオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. MSDTC オプションが含まれているオプショングループ (前の例では msdtc-se-2016) を選択します。
4. [オプションの削除] を選択します。
5. [オプションの削除] で、[削除するオプション] として [MSCTC] を選択します。
6. [すぐに適用] で、オプションをすぐに削除する場合は [はい] を選択し、次のメンテナンスウィンドウで削除する場合は [いいえ] を選択します。
7. [削除] を選択します。

CLI

MSDTC オプションをそのオプショングループから削除するには

- 以下のいずれかのコマンドを使用します。

Example

Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name msdtc-se-2016 \
    --options MSDTC \
    --apply-immediately
```

Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name msdtc-se-2016 ^
    --options MSDTC ^
    --apply-immediately
```

RDS for SQL Server の MSDTC のトラブルシューティング

場合によっては、クライアントコンピュータで実行されている MSDTC と RDS for SQL Server DB インスタンスで実行されている MSDTC サービスとの間の接続を確立できないことがあります。その場合は、以下のことを確認してください。

- DB インスタンスに関連付けられているセキュリティグループのインバウンドルールが正しく設定されている。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。
- クライアントコンピュータが正しく設定されている。
- クライアントコンピュータの MSDTC ファイアウォールルールが有効になっている。

クライアントコンピュータを設定するには

1. [コンポーネントサービス] を開きます。

   または、[サービスマネージャー] で、[ツール]、[コンポーネントサービス] の順に選択します。
2. [コンポーネントサービス]、[コンピュータ]、[マイコンピュータ]、[分散トランザクションコーディネーター] の順に展開します。
3. [ローカル DTC] のコンテキスト (右クリック) メニューを開き、[プロパティ] を選択します。

4. [セキュリティ] タブを選択します。
5. 次の項目をすべて選択します。
   - ネットワーク DTC アクセス
   - 受信を許可する
   - 送信を許可する
6. 正しい認証モードが選択されていることを確認します。
   - 相互認証を必要とする - クライアントマシンは、分散トランザクションに参加している他のノードと同じドメインに参加しているか、ドメイン間に信頼関係が設定されています。
   - 認証を必要としない - 他のすべてのケース。
7. [OK] を選択して変更を保存します。
8. サービスの再起動を求めるメッセージが表示されたら、[はい] を選択します。

MSDTC ファイアウォールルールを有効にするには

1. Windows ファイアウォールを開き、[詳細設定] を選択します。

   または、[サーバー マネージャー] で、[ツール] を選択し、[セキュリティが強化された Windows ファイアウォール] を選択します。

   > **Note**
   >
   > オペレーティングシステムによっては、Windows ファイアウォールを Windows Defender ファイアウォールと呼ぶ場合があります。

2. 左側のペインで [受信ルール] を選択します。
3. 次のファイアウォールルールが有効になっていない場合は、有効にします。
   - 分散トランザクションコーディネーター (RPC)
   - 分散トランザクションコーディネーター (RPC)-EPMAP
   - 分散トランザクションコーディネーター (TCP-In)
4. Windows ファイアウォールを閉じます。

RDS for SQL Server を使用した Microsoft SQL Server リソースガバナー

リソースガバナーは、インスタンスリソースを正確に制御できる SQL Server Enterprise Edition の機能です。これにより、ワークロードが CPU、メモリ、物理 I/O リソースを使用する方法に特定の制限を設定できます。リソースガバナーを使用すると、次のことができます。

- さまざまなワークロードがインスタンスリソースを共有する方法を管理することで、マルチテナント環境でのリソースの独占を防ぐ
- ユーザーやアプリケーションごとに特定のリソース制限と優先順位を設定することで、予測可能なパフォーマンスを実現する

既存の RDS for SQL Server DB インスタンスまたは新しい RDS for SQL Server DB インスタンスでリソースガバナーを有効にできます。

リソースガバナーは、次の 3 つの基本概念を使用します。

- リソースプール - インスタンスの物理リソース (CPU、メモリ、I/O) を管理するコンテナ。2 つの組み込みプール (内部プールとデフォルトプール) を取得し、追加のカスタムプールを作成できます。
- ワークロードグループ - 同様の特性を持つデータベースセッションのコンテナ。すべてのワークロードグループはリソースプールに属します。2 つの組み込みワークロードグループ (内部およびデフォルト) を取得し、追加のカスタムワークロードグループを作成できます。
- 分類 - ユーザー名、アプリケーション名、データベース名、またはホスト名に基づいて、受信セッションを処理するワークロードグループを決定するプロセス。

SQL Server のリソースガバナー機能の詳細については、Microsoft ドキュメントの「リソースガバナー」を参照してください。

目次

- サポート対象のバージョンとリージョン
- 制限と推奨事項
- RDS for SQL Server インスタンスの Microsoft SQL Server リソースガバナーを有効にする
  - RESOURCE_GOVERNOR のオプショングループの作成
  - RESOURCE_GOVERNOR オプションのオプショングループへの追加
  - オプショングループを DB インスタンスに関連付ける

- RDS for SQL Server インスタンスでの Microsoft SQL Server リソースガバナーの使用
  - リソースプールの管理
    - リソースプールの作成
    - リソースプールの変更
    - リソースプールの削除
  - ワークロードグループの管理
    - ワークロードグループを作成する
    - ワークロードグループの変更
    - ワークロードグループの削除
  - 分類子関数を作成して登録する
  - 分類子関数の削除
  - 分類子関数の登録解除
  - 統計をリセットする
  - リソースガバナーの設定変更
  - TempDB をリソースプールにバインドする
  - リソースプールから TempDB をバインド解除する
  - リソースガバナーのクリーンアップ
- マルチ AZ 配置に関する考慮事項
- リードレプリカに関する考慮事項
- RDS for SQL Server インスタンスのシステムビューを使用して Microsoft SQL Server リソースガバナーをモニタリングする
  - リソースプールランタイム統計
- RDS for SQL Server インスタンスの Microsoft SQL Server リソースガバナーを無効にする
- RDS for SQL Server でリソースガバナーを設定するためのベストプラクティス

サポート対象のバージョンとリージョン

Amazon RDS は、RDS for SQL Server が利用可能なすべての AWS リージョンで、次の SQL Server バージョンとエディションのリソースガバナーをサポートしています。

- SQL Server 2025 Standard、Developer、および Enterprise Edition
- SQL Server 2022: Developer および Enterprise Edition

- SQL Server 2019 Enterprise Edition
- SQL Server 2017 Enterprise Edition
- SQL Server 2016 Enterprise Edition

制限と推奨事項

リソースガバナーには、次の制限と推奨事項が適用されます。

- エディションとサービスの制限:
  - バージョン 2016 ~ 2025 の SQL Server Enterprise Edition および Developer Edition で使用できます。SQL Server 2025 以降、リソースガバナーは Standard Edition でも使用できます。
  - リソース管理は SQL Server データベースエンジンに限定されます。Analysis Services、Integration Services、Reporting Services のリソースガバナーはサポートされていません。
- 設定の制限:
  - すべての設定で Amazon RDS ストアドプロシージャを使用する必要があります。
  - ネイティブ DDL ステートメントと SQL Server Management Studio GUI 設定はサポートされていません。
- リソースプールパラメータ
  - `rds_` で始まるプール名はサポートされていません。
  - 内部リソースプールとデフォルトリソースプールの変更は許可されません。
  - ユーザー定義のリソースプールでは、次のリソースプールパラメータはサポートされていません。
    - `MIN_MEMORY_PERCENT`
    - `MIN_CPU_PERCENT`
    - `MIN_IOPS_PER_VOLUME`
    - `AFFINITY`
- ワークロードグループのパラメータ:
  - `rds_` で始まるワークロードグループ名はサポートされていません。
  - 内部ワークロードグループの変更は許可されません。
  - `@GROUP_MAX_TEMPDB_DATA_MB` および `@GROUP_MAX_TEMPDB_DATA_PERCENT` パラメータは、SQL Server 2025 以降でのみサポートされています。それより前のバージョンでこれらのパラメータを指定すると、エラーが返されます。

  - デフォルトのワークロードグループの場合:
    - REQUEST_MAX_MEMORY_GRANT_PERCENT パラメータは変更できません。
    - デフォルトのワークロードグループの場合、REQUEST_MAX_MEMORY_GRANT_PERCENT は 1 ～70 である必要があります。
    - 他のすべてのパラメータはロックされており、変更できません。
  - ユーザー定義のワークロードグループでは、すべてのパラメータを変更できます。
- 分類子関数の制限:
  - 分類子関数は、指定された条件 (ユーザー名、データベース、ホスト、またはアプリケーション名) に基づいて接続をカスタムワークロードグループにルーティングします。
  - それぞれのルーティング条件を持つユーザー定義のワークロードグループを最大 2 つサポートします。
  - 各グループ内で基準を AND 条件と組み合わせます。
  - ワークロードグループごとに少なくとも 1 つのルーティング基準が必要です。
  - 上記の分類方法のみがサポートされています。
  - 関数名の先頭には、rg_classifier_ を付ける必要があります。
  - 条件が一致しない場合のデフォルトのグループ割り当て。

RDS for SQL Server インスタンスの Microsoft SQL Server リソースガバナーを有効にする

RDS for SQL Server DB インスタンスに RESOURCE_GOVERNOR オプションを追加して、リソースガバナーを有効にします。以下のプロセスを使用します。

1. 新しいオプショングループを作成するか、既存のオプショングループを選択します。
2. オプショングループに [RESOURCE_GOVERNOR] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

> **Note**
>
> オプショングループを使用してリソースガバナーを有効にする場合、再起動は必要ありません。

RESOURCE_GOVERNOR のオプショングループの作成

リソースガバナーを有効にするには、使用する DB インスタンスの SQL Server のエディションとバージョンに対応するオプショングループを作成または変更します。この手順を完了するには、AWS マネジメントコンソール または AWS CLI を使用してください。

コンソール

次の手順を使用して、SQL Server Enterprise Edition 2022 のオプショングループを作成します。

オプショングループを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. [Create group] (グループの作成) を選択します。
4. [Create subnet group(オプショングループの作成)] ウィンドウで以下を行います。
   a. [名前] に、AWS アカウント内で一意のオプショングループ名 (**resource-governor-ee-2022** など) を入力します。名前には、英字、数字、ハイフンのみを使用できます。
   b. [説明] に、オプショングループの簡単な説明 (**RESOURCE_GOVERNOR option group for SQL Server EE 2022** など) を入力します。この説明は表示用に使用されます。
   c. [エンジン] で [sqlserver-ee] を選択します。
   d. [メジャーエンジンのバージョン] で、[16.00] を選択します。
5. [作成] を選択します。

CLI

次の手順では、SQL Server Enterprise Edition 2022 のオプショングループを作成します。

オプショングループを作成するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-option-group \
```

```
    --option-group-name resource-governor-ee-2022 \
    --engine-name sqlserver-ee \
    --major-engine-version 16.00 \
    --option-group-description "RESOURCE_GOVERNOR option group for SQL Server EE
 2022"
```

Windows の場合:

```
aws rds create-option-group ^
    --option-group-name resource-governor-ee-2022 ^
    --engine-name sqlserver-ee ^
    --major-engine-version 16.00 ^
    --option-group-description "RESOURCE_GOVERNOR option group for SQL Server EE
 2022"
```

RESOURCE_GOVERNOR オプションのオプショングループへの追加

次に、AWS マネジメントコンソール または AWS CLI を使用して RESOURCE_GOVERNOR オプションをオプショングループに追加します。

コンソール

RESOURCE_GOVERNOR オプションを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 作成したオプショングループ (この例では resource-governor-ee-2022) を選択します。
4. [オプションを追加] を選択します。
5. [オプションの詳細] で、[オプション名] として [RESOURCE_GOVERNOR] を選択します。
6. [スケジュール] で、オプションをすぐに追加するか、次のメンテナンスウィンドウで追加するかを選択します。
7. [オプションを追加] を選択します。

CLI

RESOURCE_GOVERNOR オプションを追加するには

- オプショングループに [RESOURCE_GOVERNOR] オプションを追加します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name resource-governor-ee-2022 \
    --options "OptionName=RESOURCE_GOVERNOR" \
    --apply-immediately
```

  Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name resource-governor-ee-2022 ^
    --options "OptionName=RESOURCE_GOVERNOR" ^
    --apply-immediately
```

オプショングループを DB インスタンスに関連付ける

RESOURCE_GOVERNOR オプショングループを DB インスタンスに関連付けるには、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

リソースガバナーの有効化を完了するには、RESOURCE_GOVERNOR オプショングループを新規または既存の DB インスタンスに関連付けます。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを変更することでそれらを関連付けます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

CLI

RESOURCE_GOVERNOR オプショングループを新規または既存の DB インスタンスに関連付けることができます。

RESOURCE_GOVERNOR オプショングループを使用してインスタンスを作成するには

- オプショングループの作成時に使用したのと同じ DB エンジンのタイプとメジャーバージョンを指定します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mytestsqlserverresourcegovernorinstance \
    --db-instance-class db.m5.2xlarge \
    --engine sqlserver-ee \
    --engine-version 16.00 \
    --license-model license-included \
    --allocated-storage 100 \
    --master-username admin \
    --master-user-password password \
    --storage-type gp2 \
    --option-group-name resource-governor-ee-2022
```

  Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mytestsqlserverresourcegovernorinstance ^
    --db-instance-class db.m5.2xlarge ^
    --engine sqlserver-ee ^
    --engine-version 16.00 ^
    --license-model license-included ^
    --allocated-storage 100 ^
    --master-username admin ^
    --master-user-password password ^
    --storage-type gp2 ^
    --option-group-name resource-governor-ee-2022
```

インスタンスを変更して **RESOURCE_GOVERNOR** オプショングループを関連付けるには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mytestinstance \
    --option-group-name resource-governor-ee-2022 \
    --apply-immediately
```

  Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mytestinstance ^
    --option-group-name resource-governor-ee-2022 ^
    --apply-immediately
```

RDS for SQL Server インスタンスでの Microsoft SQL Server リソースガバナーの使用

リソースガバナーオプションをオプショングループに追加した後も、リソースガバナーはデータベースエンジンレベルでまだアクティブではありません。リソースガバナーを完全に有効にするには、RDS for SQL Server ストアドプロシージャを使用してリソースガバナーオブジェクトを有効にし、必要なリソースガバナーオブジェクトを作成する必要があります。詳細については、「SQL Server DB インスタンスへの接続」を参照してください。

まず、SQL Server データベースに接続し、適切な RDS for SQL Server ストアドプロシージャを呼び出して設定を完了します。データベースに接続する手順については、「SQL Server DB インスタンスへの接続」を参照してください。

各ストアドプロシージャを呼び出す方法については、以下のトピックを参照してください。

トピック

- リソースプールの管理
- ワークロードグループの管理
- 分類子関数を作成して登録する
- 分類子関数の削除

- 分類子関数の登録解除
- 統計をリセットする
- リソースガバナーの設定変更
- TempDB をリソースプールにバインドする
- リソースプールから TempDB をバインド解除する
- リソースガバナーのクリーンアップ

リソースプールの管理

リソースプールの作成

オプショングループでリソースガバナーを有効にすると、rds_create_resource_pool を使用してカスタムリソースプールを作成できます。これらのプールを使用すると、CPU、メモリ、IOPS の特定の割合をさまざまなワークロードに割り当てることができます。

使用方法

```
USE [msdb]
EXEC dbo.rds_create_resource_pool
    @pool_name=value,
    @MAX_CPU_PERCENT=value,
    @CAP_CPU_PERCENT=value,
    @MAX_MEMORY_PERCENT=value,
    @MAX_IOPS_PER_VOLUME=value
```

以下のパラメータは必須です。

- @group_name - 既存のユーザー定義のワークロードグループの名前です。
- @pool_name - リソースプールのユーザー定義の名前です。*pool_name* は英数字で、最大 128 文字で、データベースエンジンインスタンス内で一意である必要があり、データベース識別子のルールに準拠する必要があります。

以下のパラメータはオプションです。

- @MAX_CPU_PERCENT - CPU 競合がある場合にリソースプール内のすべてのリクエストが受け取る最大平均 CPU 帯域幅を指定します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1 ~ 100 です。

- @CAP_CPU_PERCENT - リソースプール内のすべてのリクエストが受信する CPU 帯域幅のハードキャップを指定します。最大 CPU 帯域幅レベルを指定された値と同じに制限します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1～100 です。
- @MAX_MEMORY_PERCENT - このリソースプールのリクエストで使用できるクエリワークスペースメモリの最大量を指定します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1～100 です。
- @MAX_IOPS_PER_VOLUME - リソースプールを許可するディスクボリュームあたりの 1 秒あたりの最大 I/O オペレーション (IOPS) を指定します。#の許容範囲は 0～2^31-1 (2,147,483,647) です。プールの IOPS 制限を削除するには、0 を指定します。デフォルトは 0 です。

例

すべてのデフォルト値を使用してリソースプールを作成する例:

```
--This creates resource pool 'SalesPool' with all default values
USE [msdb]
EXEC rds_create_resource_pool @pool_name = 'SalesPool';

--Apply changes
USE [msdb]
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_resource_pools
```

異なるパラメータを指定してリソースプールを作成する例:

```
--creates resource pool
USE [msdb]
EXEC dbo.rds_create_resource_pool
@pool_name='analytics',
@MAX_CPU_PERCENT = 30,
@CAP_CPU_PERCENT = 40,
@MAX_MEMORY_PERCENT = 20;

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_resource_pools
```

リソースプールの変更

使用方法

```
USE [msdb]
EXEC dbo.rds_alter_resource_pool
    @pool_name=value,
    @MAX_CPU_PERCENT=value,
    @CAP_CPU_PERCENT=value,
    @MAX_MEMORY_PERCENT=value,
    @MAX_IOPS_PER_VOLUME=value;
```

以下のパラメータは必須です。

- @pool_name - 既存のユーザー定義リソースプールの名前です。Amazon RDS SQL Server では、デフォルトのリソースプールの変更は許可されていません。

オプションのパラメータを少なくとも 1 つ指定する必要があります。

- @MAX_CPU_PERCENT - CPU 競合がある場合にリソースプール内のすべてのリクエストが受け取る最大平均 CPU 帯域幅を指定します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1 ~ 100 です。
- @CAP_CPU_PERCENT - リソースプール内のすべてのリクエストが受信する CPU 帯域幅のハードキャップを指定します。最大 CPU 帯域幅レベルを指定された値と同じに制限します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1 ~ 100 です。
- @MAX_MEMORY_PERCENT - このリソースプールのリクエストで使用できるクエリワークスペースメモリの最大量を指定します。#は、デフォルト設定の 100 の整数です。#の許容範囲は 1 ~ 100 です。
- @MAX_IOPS_PER_VOLUME - リソースプールを許可するディスクボリュームあたりの 1 秒あたりの最大 I/O オペレーション (IOPS) を指定します。#の許容範囲は 0 ~ 2^31-1 (2,147,483,647) です。プールの IOPS 制限を削除するには、0 を指定します。デフォルトは 0 です。

例

```
--This alters resource pool
USE [msdb]
EXEC dbo.rds_alter_resource_pool
    @pool_name='analytics',
```

```
    @MAX_CPU_PERCENT = 10,
    @CAP_CPU_PERCENT = 20,
    @MAX_MEMORY_PERCENT = 50;

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration.
select * from sys.resource_governor_resource_pools
```

リソースプールの削除

使用方法

```
USE [msdb]
EXEC dbo.rds_drop_resource_pool
@pool_name=value;
```

以下のパラメータは必須です。

- `@pool_name` - 既存のユーザー定義リソースプールの名前です。

> **Note**
>
> SQL Server では、内部リソースプールまたはデフォルトリソースプールの削除は許可されていません。

例

```
--This drops resource pool
USE [msdb]
EXEC dbo.rds_drop_resource_pool
@pool_name='analytics'

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_resource_pools
```

ワークロードグループの管理

rds_create_workload_group および rds_alter_workload_group で作成および管理されるワークロードグループを使用すると、クエリグループの重要度レベル、メモリ許可、およびその他のパラメータを設定できます。

ワークロードグループを作成する

使用方法

```
USE [msdb]
EXEC dbo.rds_create_workload_group
@group_name = value,
@IMPORTANCE ={ LOW | MEDIUM | HIGH },
@REQUEST_MAX_MEMORY_GRANT_PERCENT =value,
@REQUEST_MAX_CPU_TIME_SEC = value ,
@REQUEST_MEMORY_GRANT_TIMEOUT_SEC = value,
@MAX_DOP = value,
@GROUP_MAX_REQUESTS = value,
@GROUP_MAX_TEMPDB_DATA_MB = value,
@GROUP_MAX_TEMPDB_DATA_PERCENT = value,
@pool_name = value
```

以下のパラメータは必須です。

- @pool_name - 既存のユーザー定義リソースプールの名前です。
- @group_name - 既存のユーザー定義のワークロードグループの名前です。

以下のパラメータはオプションです。

- @IMPORTANCE - ワークロードグループ内のリクエストの相対的な重要度を指定します。デフォルト値は MEDIUM です。
- @REQUEST_MAX_MEMORY_GRANT_PERCENT - 1 つのリクエストがプールから取得できるクエリワークスペースメモリの最大量を指定します。#は、MAX_MEMORY_PERCENT で定義されたリソースプールサイズのパーセンテージです。デフォルト値は 25 です。
- @REQUEST_MAX_CPU_TIME_SEC - バッチリクエストで使用できる CPU の最大時間を秒単位で指定します。#は 0 または正の整数である必要があります。#のデフォルト設定は 0 です。つまり、無制限です。

- @REQUEST_MEMORY_GRANT_TIMEOUT_SEC - クエリがクエリワークスペースメモリからのメモリ許可が利用可能になるまで待機できる最大時間を秒単位で指定します。#は 0 または正の整数である必要があります。#のデフォルト設定である 0 では、クエリコストに基づく内部計算を使用して最大時間を決定します。
- @MAX_DOP - 並列クエリ実行の最大並列度 (MAXDOP) を指定します。#の許容範囲は 0～64 です。#のデフォルト設定である 0 では、グローバル設定が使用されます。
- @GROUP_MAX_REQUESTS = ワークロードグループで実行できる同時リクエストの最大数を指定します。#は 0 または正の整数である必要があります。#のデフォルト設定は 0 で、無制限のリクエストを許可します。
- @GROUP_MAX_TEMPDB_DATA_MB – ワークロードグループが消費できる tempdb データスペースの最大量をメガバイト単位で指定します。#は 0、正の数値、または NULL である必要があります。小数値を使用できます。値が 0 の場合、ワークロードグループのセッションによる tempdb スペースの割り当ては許可されません。値が設定されていない場合、リソースガバナーはワークロードグループによる tempdb スペースの消費を制限しません。このパラメータは SQL Server 2025 以降で使用できます。
- @GROUP_MAX_TEMPDB_DATA_PERCENT – ワークロードグループが消費できる tempdb データスペースの最大量を、tempdb の最大サイズに対する割合で指定します。#は 0～100 の範囲、または NULL である必要があります。小数値を使用できます。値が 0 の場合、ワークロードグループのセッションによる tempdb スペースの割り当ては許可されません。値が設定されていない場合、リソースガバナーはワークロードグループによる tempdb スペースの消費を制限しません。GROUP_MAX_TEMPDB_DATA_MB が設定されている場合、または tempdb の最大サイズが定義されていない場合、GROUP_MAX_TEMPDB_DATA_PERCENT は効果がありません。このパラメータは SQL Server 2025 以降で使用できます。
- @pool_name = ワークロードグループを *pool_name* で識別されるユーザー定義のリソースプール、または `default` リソースプールに関連付けます。*pool_name* が指定されていない場合、ワークロードグループは組み込み `default` プールに関連付けられます。

例

```
--This creates workload group named 'analytics'
USE msdb;
EXEC dbo.rds_create_workload_group
    @group_name = 'analytics',
    @IMPORTANCE = 'HIGH',
    @REQUEST_MAX_MEMORY_GRANT_PERCENT = 25,
    @REQUEST_MAX_CPU_TIME_SEC = 0,
```

```
    @REQUEST_MEMORY_GRANT_TIMEOUT_SEC = 0,
    @MAX_DOP = 0,
    @GROUP_MAX_REQUESTS = 0,
    @pool_name = 'analytics';

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_workload_groups
```

tempdb ガバナンスを使用したワークロードグループの作成例 (SQL Server 2025 以降):

```
-- Create workload group with tempdb space limits
USE msdb;
EXEC dbo.rds_create_workload_group
    @group_name = 'etl_workload',
    @IMPORTANCE = 'MEDIUM',
    @REQUEST_MAX_MEMORY_GRANT_PERCENT = 25,
    @MAX_DOP = 4,
    @GROUP_MAX_TEMPDB_DATA_MB = 2048,
    @GROUP_MAX_TEMPDB_DATA_PERCENT = 30,
    @pool_name = 'etl_pool';

-- Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

-- Validate configuration
SELECT name, group_max_tempdb_data_mb, group_max_tempdb_data_percent
FROM sys.resource_governor_workload_groups
WHERE name = 'etl_workload';
```

ワークロードグループの変更

使用方法

```
EXEC msdb.dbo.rds_alter_workload_group
    @group_name = value,
    @IMPORTANCE = 'LOW|MEDIUM|HIGH',
    @REQUEST_MAX_MEMORY_GRANT_PERCENT = value,
    @REQUEST_MAX_CPU_TIME_SEC = value,
    @REQUEST_MEMORY_GRANT_TIMEOUT_SEC = value,
    @MAX_DOP = value,
```

```
    @GROUP_MAX_REQUESTS = value,
    @GROUP_MAX_TEMPDB_DATA_MB = value,
    @GROUP_MAX_TEMPDB_DATA_PERCENT = value,
    @pool_name = value
```

以下のパラメータは必須です。

- @group_name - デフォルトまたは既存のユーザー定義のワークロードグループの名前です。

> **Note**
>
> デフォルトのワークロードグループの REQUEST_MAX_MEMORY_GRANT_PERCENT パラメータのみの変更がサポートされています。デフォルトのワークロードグループの場合、REQUEST_MAX_MEMORY_GRANT_PERCENT は 1～70 である必要があります。デフォルトのワークロードグループでは、他のパラメータを変更することはできません。すべてのパラメータは、ユーザー定義のワークロードグループで変更できます。

以下のパラメータはオプションです。

- @IMPORTANCE - ワークロードグループ内のリクエストの相対的な重要度を指定します。デフォルト値は MEDIUM です。
- @REQUEST_MAX_MEMORY_GRANT_PERCENT - 1 つのリクエストがプールから取得できるクエリワークスペースメモリの最大量を指定します。#は、MAX_MEMORY_PERCENT で定義されたリソースプールサイズのパーセンテージです。デフォルト値は 25 です。Amazon RDS では、REQUEST_MAX_MEMORY_GRANT_PERCENT は 1～70 である必要があります。
- @REQUEST_MAX_CPU_TIME_SEC - バッチリクエストで使用できる CPU の最大時間を秒単位で指定します。#は 0 または正の整数である必要があります。#のデフォルト設定は 0 です。つまり、無制限です。
- @REQUEST_MEMORY_GRANT_TIMEOUT_SEC - クエリがクエリワークスペースメモリからのメモリ許可が利用可能になるまで待機できる最大時間を秒単位で指定します。#は 0 または正の整数である必要があります。#のデフォルト設定である 0 では、クエリコストに基づく内部計算を使用して最大時間を決定します。
- @MAX_DOP - 並列クエリ実行の最大並列度 (MAXDOP) を指定します。#の許容範囲は 0～64 です。#のデフォルト設定である 0 では、グローバル設定が使用されます。

- @GROUP_MAX_REQUESTS - ワークロードグループで実行できる同時リクエストの最大数を指定します。#は 0 または正の整数である必要があります。#のデフォルト設定は 0 で、無制限のリクエストを許可します。
- @GROUP_MAX_TEMPDB_DATA_MB – ワークロードグループが消費できる tempdb データスペースの最大量をメガバイト単位で指定します。#は 0、正の数値、または NULL である必要があります。小数値を使用できます。このパラメータは SQL Server 2025 以降で使用できます。
- @GROUP_MAX_TEMPDB_DATA_PERCENT – ワークロードグループが消費できる tempdb データスペースの最大量を、tempdb の最大サイズに対する割合で指定します。#は 0 ~ 100 の範囲、または NULL である必要があります。小数値を使用できます。GROUP_MAX_TEMPDB_DATA_MB が設定されている場合、または tempdb の最大サイズが定義されていない場合、GROUP_MAX_TEMPDB_DATA_PERCENT は効果がありません。このパラメータは SQL Server 2025 以降で使用できます。
- @pool_name - ワークロードグループを *pool_name* で識別されるユーザー定義のリソースプールに関連付けます。

例

デフォルトのワークロードグループ変更 REQUEST_MAX_MEMORY_GRANT_PERCENT を変更する例:

```
--Modify default workload group (set memory grant cap to 10%)
USE msdb
EXEC dbo.rds_alter_workload_group
    @group_name = 'default',
    @REQUEST_MAX_MEMORY_GRANT_PERCENT=10;

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
SELECT * FROM sys.resource_governor_workload_groups WHERE name='default';
```

デフォルト以外のワークロードグループを変更する例:

```
EXEC msdb.dbo.rds_alter_workload_group
    @group_name = 'analytics',
    @IMPORTANCE = 'HIGH',
    @REQUEST_MAX_MEMORY_GRANT_PERCENT = 30,
```

```
    @REQUEST_MAX_CPU_TIME_SEC = 3600,
    @REQUEST_MEMORY_GRANT_TIMEOUT_SEC = 60,
    @MAX_DOP = 4,
    @GROUP_MAX_REQUESTS = 100;

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;
```

デフォルト以外のワークロードグループを別のリソースプールに移動する例:

```
EXEC msdb.dbo.rds_alter_workload_group
@group_name = 'analytics',
@pool_name='abc'

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_workload_groups
```

tempdb ガバナンス (SQL Server 2025 以降) を使用したワークロードグループの変更例:

```
-- Add tempdb limits to an existing workload group
USE msdb;
EXEC msdb.dbo.rds_alter_workload_group
    @group_name = 'etl_workload',
    @GROUP_MAX_TEMPDB_DATA_MB = 4096,
    @GROUP_MAX_TEMPDB_DATA_PERCENT = 50;

-- Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

-- Validate configuration
SELECT name, group_max_tempdb_data_mb, group_max_tempdb_data_percent
FROM sys.resource_governor_workload_groups
WHERE name = 'etl_workload';
```

ワークロードグループの削除

使用方法

```
EXEC msdb.dbo.rds_drop_workload_group
```

```
@group_name = value
```

以下のパラメータは必須です。

- @group_name - 既存のユーザー定義のワークロードグループの名前です。

例

```
--Drops a Workload Group:
EXEC msdb.dbo.rds_drop_workload_group
@group_name = 'analytics';

--Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;

--Validate configuration
select * from sys.resource_governor_workload_groups
```

分類子関数を作成して登録する

この手順では、指定された条件 (ユーザー名、データベース、ホスト、またはアプリケーション名) に基づいてカスタムワークロードグループへの接続をルーティングするリソースガバナーの分類子関数をマスターデータベースに作成します。リソースガバナーが有効で、分類子関数がリソースガバナー設定で指定されている場合、関数の出力によって新しいセッションに使用されるワークロードグループが決まります。分類子関数がない場合、すべてのセッションが default グループに分類されます。

特徴:

- それぞれのルーティング条件で最大 2 つのワークロードグループをサポートします。
- 各グループ内で基準を AND 条件と組み合わせます。
- ワークロードグループごとに少なくとも 1 つのルーティング基準が必要です。
- 関数名の先頭には、rg_classifier_ を付ける必要があります。
- 条件が一致しない場合のデフォルトのグループ割り当て。

分類子関数には、次の特性と動作があります。

- 関数はサーバースコープ (マスターデータベース) で定義されます。

- 関数はスキーマバインディングを使用して定義されます。
- この関数は、接続プーリングが有効になっている場合でも、新しいセッションごとに評価されます。
- 関数は、セッションのワークロードグループコンテキストを返します。セッションは、セッションの存続期間中、分類子によって返されるワークロードグループに割り当てられます。
- 関数が NULL、デフォルト、または存在しないワークロードグループの名前を返す場合、セッションにはデフォルトのワークロードグループコンテキストが与えられます。何らかの理由で関数が失敗した場合も、セッションにはデフォルトのコンテキストが与えられます。
- 複数の分類子関数を作成できます。ただし、SQL Server では、一度に登録できる分類子関数は 1 つだけです。
- 分類子名を NULL に設定する登録解除手順 (EXEC dbo.msdb.rds_alter_resource_governor_configuration @deregister_function = 1;) を使用して分類子ステータスが削除されるか、(EXEC dbo.msdb.rds_alter_resource_governor_configuration @classifier_function = <function_name>;) を使用して別の分類子関数が登録されない限り、分類子関数を削除することはできません。
- 分類子関数がない場合、すべてのセッションがデフォルトグループに分類されます。
- 分類子関数は、リソースガバナー設定で参照されている間は変更できません。ただし、別の分類子関数を使用するように設定を変更することはできます。分類子を変更する場合は、分類子関数のペアの作成を検討してください。例えば、rg_classifier_a と rg_classifier_b を作成できます。

使用方法

```
EXEC msdb.dbo.rds_create_classifier_function
@function_name = value,
@workload_group1 = value,
@user_name1 = value,
@db_name1 = value,
@host_name1 = value,
@app_name1 = value,
@workload_group2 = value,
@user_name2 = value,
@db_name2 = value,
@host_name2 = value,
@app_name2 = value
```

以下のパラメータは必須です。

- @function_name - 分類子関数の名前。rg_classifier_ で始まる必要があります
- @workload_group1 - 最初のワークロードグループの名前

以下のパラメータはオプションです。

(グループ 1 には、これらの基準の少なくとも 1 つを指定する必要があります)

- @user_name1 - グループ 1 のログイン名
- @db_name1 - グループ 1 のデータベース名
- @host_name1 - グループ 1 のホスト名
- @app_name1 - グループ 1 のアプリケーション名

(グループ 2 が指定されている場合、少なくとも 1 つの条件を指定する必要があります)

- @workload_group2 - 2 番目のワークロードグループの名前
- @user_name2 - グループ 2 のログイン名
- @db_name2 - グループ 2 のデータベース名
- @host_name2 - グループ 2 のホスト名
- @app_name2 - グループ 2 のアプリケーション名

> **Note**
> システムアカウント、データベース、アプリケーション、ホストは制限されています。

例

1 つのワークロードグループの基本的な例:

```
/*Create a classifier to route all requests from 'PowerBI' app to workload group
'reporting_group'*/

EXEC msdb.dbo.rds_create_classifier_function
@function_name = 'rg_classifier_a',
```

```
@workload_group1 = 'reporting_group',
@app_name1 = 'PowerBI';

--Register the classifier
EXEC msdb.dbo.rds_alter_resource_governor_configuration
@classifier_function = 'rg_classifier_a';

-- Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration

/*Query sys.resource_governor_configuration to validate that resource governor is
 enabled and is using the classifier function we created and registered*/

use master
go
SELECT OBJECT_SCHEMA_NAME(classifier_function_id) AS classifier_schema_name,
       OBJECT_NAME(classifier_function_id) AS classifier_object_name,
       is_enabled
FROM sys.resource_governor_configuration;
```

分類子関数の削除

使用方法

```
USE [msdb]
EXEC dbo.rds_drop_classifier_function
@function_name = value;
```

以下のパラメータは必須です。

- @function_name - 既存のユーザー定義分類子関数の名前です。

例

```
EXEC msdb.dbo.rds_drop_classifier_function
@function_name = 'rg_classifier_b';
```

分類子関数の登録解除

分類子関数の登録を解除するには、この手順を使用します。関数の登録が解除されると、新しいセッションは自動的にデフォルトのワークロードグループに割り当てられます。

使用方法

```
USE [msdb]
EXEC dbo.rds_alter_resource_governor_configuration
@deregister_function = 1;
```

登録解除には、次のパラメータが必要です。

- `@deregister_function` は 1 でなければなりません。

例

```
EXEC msdb.dbo.rds_alter_resource_governor_configuration
    @deregister_function = 1;
GO

-- Apply changes
EXEC msdb.dbo.rds_alter_resource_governor_configuration;
```

統計をリセットする

リソースガバナー統計は、前回のサーバー再起動以降に累積されます。特定の時間から統計を収集する必要がある場合は、次の Amazon RDS ストアドプロシージャを使用して統計をリセットできます。

使用方法

```
USE [msdb]
EXEC dbo.rds_alter_resource_governor_configuration
@reset_statistics = 1;
```

統計をリセットするには、次のパラメータが必要です。

- `@reset_statistics` は 1 でなければなりません。

リソースガバナーの設定変更

リソースガバナーが有効になっていない場合、`rds_alter_resource_governor_configuration` はリソースガバナーを有効にします。リソースガバナーを有効にすると、次の結果が得られます。

- 分類子関数がある場合は、新しいセッションに対して実行され、ワークロードグループに割り当てられます。
- リソースガバナー設定で指定されたリソース制限が優先され、適用されます。
- リソースガバナー設定で指定されたリソース制限が優先され、適用されます。
- リソースガバナーを有効にする前に存在していたリクエストは、リソースガバナーが有効になっているときに行われた設定変更の影響を受ける可能性があります。
- リソースガバナーを有効にする前からの既存のリクエストは、リソースガバナーが有効になっているときに行われた設定変更の影響を受ける可能性があります。
- RDS for SQL Server で、リソースガバナー設定の変更を有効にするには、`EXEC msdb.dbo.rds_alter_resource_governor_configuration` を実行する必要があります。

使用方法

```
USE [msdb]
EXEC dbo.rds_alter_resource_governor_configuration
```

TempDB をリソースプールにバインドする

Amazon RDS SQL Server バージョン 2019 以降では、`rds_bind_tempdb_metadata_to_resource_pool` を使用して、tempdb メモリ最適化メタデータを特定のリソースプールにバインドできます。

> **Note**
>
> tempdb メタデータをリソースプールにバインドする前に、メモリ最適化 tempdb メタデータ機能を有効にする必要があります。Amazon RDS でこの機能を有効にするには、静的パラメータ `tempdb metadata memory-optimized` を使用します。

Amazon RDS で静的パラメータを有効にし、パラメータを有効にするためにフェイルオーバーなしで再起動を実行します。

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name test-sqlserver-ee-2022 \
    --parameters "ParameterName='tempdb metadata memory-
optimized',ParameterValue=True,ApplyMethod=pending-reboot"
```

使用方法

```
USE [msdb]
EXEC dbo.rds_bind_tempdb_metadata_to_resource_pool
@pool_name=value;
```

以下のパラメータは必須です。

- `@pool_name` - 既存のユーザー定義リソースプールの名前です。

> **Note**
>
> この変更では、メモリ最適化 TempDB メタデータ機能が既に有効になっている場合でも、フェイルオーバーなしで SQL サービスを再起動する必要があります。

リソースプールから TempDB をバインド解除する

tempdb メモリ最適化メタデータをリソースプールからバインド解除します。

> **Note**
>
> この変更を有効にするには、フェイルオーバーなしで SQL サービスを再起動する必要もあります

使用方法

```
USE [msdb]
EXEC dbo.rds_unbind_tempdb_metadata_from_resource_pool
```

リソースガバナーのクリーンアップ

この手順では、オプショングループからリソースガバナーオプションを削除した後、関連するすべてのオブジェクトをクリーンアップします。これにより、リソースガバナーを無効にし、デフォルトのワークロードグループをデフォルト設定に戻し、カスタムワークロードグループ、リソースプール、分類子関数を削除します。

主な特徴:

- デフォルトのワークロードグループをデフォルト設定に戻す
- リソースガバナーを無効にする
- カスタムワークロードグループを削除する
- カスタムリソースプールを削除する
- 分類子関数を削除する
- 有効になっている場合、tempdb リソースプールバインディングを削除する

> **Important**
>
> このクリーンアップは、ワークロードグループにアクティブなセッションがある場合にエラーになる可能性があります。ビジネス要件に従って、アクティブなセッションが終了するのを待つか、アクティブなセッションを終了します。これをメンテナンスウィンドウ中に実行することをお勧めします。
> このクリーンアップは、リソースプールが tempdb にバインドされ、フェイルオーバーなしで再起動されていない場合にエラーが発生する可能性があります。リソースプールを tempdb にバインドするか、以前に tempdb からリソースプールをバインド解除した場合は、フェイルオーバーなしで再起動を実行して変更を有効にします。これをメンテナンスウィンドウ中に実行することをお勧めします。

使用方法

```
USE [msdb]
EXEC dbo.rds_cleanup_resource_governor
```

マルチ AZ 配置に関する考慮事項

RDS for SQL Server は、マルチ AZ 配置のセカンダリインスタンスにリソースガバナーをレプリケートします。変更された新しいリソースガバナーがセカンダリインスタンスと最後に同期された日時を確認できます。

次のクエリを使用して、レプリケーションの `last_sync_time` を確認します。

```
SELECT * from msdb.dbo.rds_fn_server_object_last_sync_time();
```

クエリ結果で、同期時刻がリソースガバナーの更新時刻または作成時刻を過ぎている場合、リソースガバナーはセカンダリと同期します。

リソースガバナーのレプリケーションを確認するために手動で DB フェイルオーバーを実行するには、まず `last_sync_time` が更新されるのを待ちます。次に、マルチ AZ フェイルオーバーに進みます。

リードレプリカに関する考慮事項

- ソース DB インスタンスと同じリージョンの SQL Server レプリカの場合、ソースと同じオプショングループを使用します。オプショングループへの変更は、メンテナンスウィンドウに関係なく、レプリカにすぐに伝達されます。
- SQL Server クロスリージョンレプリカを作成する場合、RDS により、そのための専有オプショングループが作成されます。
- SQL Server クロスリージョンレプリカを、その専有オプショングループから削除することはできません。他の DB インスタンスで、SQL Server クロスリージョンレプリカの専有オプショングループを使用することはできません。
- リソースガバナーオプションはレプリケートされていないオプションです。専用オプショングループから、レプリケートされないオプションを追加または削除できます。
- SQL Server クロスリージョンリードレプリカを昇格するとき、昇格されたレプリカは、オプションの管理を含め、その他の SQL Server DB インスタンスと同じように動作します。

> **Note**
>
> リードレプリカでリソースガバナーを使用する場合は、オプションがオプショングループに追加された後に、Amazon RDS ストアドプロシージャを使用して、リードレプリカでリソースガバナーが手動で設定されていることを確認する必要があります。リソースガバナーの設定は、リードレプリカに自動的にレプリケートされません。また、リードレプリカのワークロードは通常、プライマリインスタンスとは異なります。したがって、ワークロードとインスタンスタイプに基づいてリソース設定をレプリカに適用することをお勧めします。リードレプリカでこれらの Amazon RDS ストアドプロシージャを個別に実行して、リードレプリカでリソースガバナーを設定できます。

RDS for SQL Server インスタンスのシステムビューを使用して Microsoft SQL Server リソースガバナーをモニタリングする

リソースガバナー統計は、前回のサーバー再起動以降に累積されます。特定の時間から統計を収集する必要がある場合は、次の Amazon RDS ストアドプロシージャを使用して統計をリセットできます。

```
EXEC msdb.dbo.rds_alter_resource_governor_configuration
@reset_statistics = 1;
```

リソースプールランタイム統計

リソースプールごとに、リソースガバナーは CPU とメモリの使用率、メモリ不足イベント、メモリ許可、I/O、その他の統計情報を追跡します。詳細については、「sys.dm_resource_governor_resource_pools」を参照してください。

次のクエリは、すべてのリソースプールで使用可能な統計のサブセットを返します。

```
SELECT rp.pool_id,
       rp.name AS resource_pool_name,
       wg.workload_group_count,
       rp.statistics_start_time,
       rp.total_cpu_usage_ms,
       rp.target_memory_kb,
       rp.used_memory_kb,
       rp.out_of_memory_count,
       rp.active_memgrant_count,
       rp.total_memgrant_count,
       rp.total_memgrant_timeout_count,
       rp.read_io_completed_total,
       rp.write_io_completed_total,
       rp.read_bytes_total,
       rp.write_bytes_total,
       rp.read_io_stall_total_ms,
       rp.write_io_stall_total_ms
FROM sys.dm_resource_governor_resource_pools AS rp
OUTER APPLY (
            SELECT COUNT(1) AS workload_group_count
            FROM sys.dm_resource_governor_workload_groups AS wg
            WHERE wg.pool_id = rp.pool_id
            ) AS wg;
```

RDS for SQL Server インスタンスの Microsoft SQL Server リソースガバナーを無効にする

RDS for SQL Server でリソースガバナーを無効にすると、サービスはワークロードリソースの管理を停止します。リソースガバナーを無効にする前に、これがデータベースの接続と設定にどのように影響するかを確認してください。

リソースガバナーを無効にすると、次の結果になります。

- 新しい接続が開かれたときに、分類子関数は実行されません。
- 新しい接続は自動的にデフォルトのワークロードグループに分類されます。
- 既存のワークロードグループとリソースプールの設定はすべてデフォルト値にリセットされます。
- 制限に達した場合、イベントは発生しません。
- リソースガバナーの設定は変更できますが、リソースガバナーが有効になるまで変更は適用されません。

リソースガバナーを無効にするには、オプショングループから `RESOURCE_GOVERNOR` オプションを削除します。

コンソール

以下の手順では、`RESOURCE_GOVERNOR` オプションを削除します。

RESOURCE_GOVERNOR オプションをオプショングループから削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. `RESOURCE_GOVERNOR` オプションが含まれているオプショングループ (前の例では `resource-governor-ee-2022`) を選択します。
4. [オプションを削除] を選択します。
5. [削除オプション] で、[削除するオプション] として [RESOURCE_GOVERNOR] を選択します。
6. [Apply immediately] (すぐに適用) で、オプションをすぐに削除する場合は [Yes] (はい) を選択し、次のメンテナンスウィンドウで削除する場合は [No] (いいえ) を選択します。
7. [削除] を選択します。

CLI

以下の手順では、RESOURCE_GOVERNOR オプションを削除します。

RESOURCE_GOVERNOR オプションをオプショングループから削除するには

- 以下のいずれかのコマンドを実行します。

  Example

  Linux、macOS、Unix の場合:

```
aws rds remove-option-from-option-group \
    --option-group-name resource-governor-ee-2022 \
    --options RESOURCE_GOVERNOR \
    --apply-immediately
```

  Windows の場合:

```
aws rds remove-option-from-option-group ^
    --option-group-name resource-governor-ee-2022 ^
    --options RESOURCE_GOVERNOR ^
    --apply-immediately
```

RDS for SQL Server でリソースガバナーを設定するためのベストプラクティス

リソースの消費を制御するために、RDS for SQL Server は Microsoft SQL Server のリソースガバナーをサポートしています。以下のベストプラクティスは、一般的な設定の問題を回避し、データベースのパフォーマンスを最適化するのに役立ちます。

1. リソースガバナーの設定は master データベースに保存されます。リソースガバナーの設定スクリプトのコピーは必ず個別に保存することをお勧めします。
2. 分類子関数はログイン処理時間を延長するため、分類子の複雑なロジックを避けることをお勧めします。関数が複雑すぎると、Amazon RDS オートメーションセッションなどのログイン遅延や接続タイムアウトが発生する可能性があります。これは、Amazon RDS オートメーションがインスタンスの状態をモニタリングする機能に影響を与える可能性があります。したがって、本番稼働前の環境で分類子関数をテストしてから、本番稼働環境に実装することをお勧めします。
3. ワークロードグループで REQUEST_MAX_MEMORY_GRANT_PERCENT に高い値 (70 以上) を設定しないでください。そうしないと、データベースインスタンスが他の同時クエリに十分なメモリを

割り当てることができなくなり、メモリ付与タイムアウトエラー (エラー 8645) が発生する可能性があります。逆に、この値を低く (1 未満) 設定したり 0 に設定すると、メモリワークスペースを必要とするクエリ (ソートまたはハッシュオペレーションを含むクエリなど) がユーザー定義のワークロードグループで適切に実行できなくなる可能性があります。RDS は、デフォルトのワークロードグループの値を 1 ~ 70 に制限することで、これらの制限を適用します。

4. tempdb をリソースプールにバインドする場合、メモリ最適化 tempdb メタデータをプールにバインドした後、プールが最大設定に達し、tempdb を使用するクエリがメモリ不足エラーで失敗する可能性があります。特定の状況では、メモリ不足エラーが発生した場合に SQL Server が停止する可能性があります。これが発生する可能性を減らすには、メモリプールの MAX_MEMORY_PERCENT を高い値に設定します。

Amazon RDS for Microsoft SQL Server の一般的な DBA タスク

このセクションでは、Microsoft SQL Server Amazon RDS データベースエンジンを実行する DB インスタンスに関連したいくつかの一般的な DBA タスクの Amazon RDS 固有の実装について説明します。マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスはできません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。

> **Note**
>
> SQL Server DB インスタンスを使用する場合、新しく作成したデータベースを変更するためのスクリプトを実行できますが、新しいデータベースのモデルとして使用される「モデル」データベースを変更することはできません。

トピック

- Amazon RDS で実行している Microsoft SQL Server DB インスタンスの tempdb データベースへのアクセス
- Database Engine Tuning Advisor を使用して Amazon RDS for SQL Server DB インスタンスのデータベースワークロードを分析する
- db_owner の Amazon RDS for SQL Server データベースの rdsa アカウントへの変更
- Amazon RDS for Microsoft SQL Server の照合と文字セットの管理
- Amazon RDS for SQL Server のデータベースユーザーの作成
- Amazon RDS for SQL Server データベースの復旧モデルの決定
- Amazon RDS for SQL Server の最終フェイルオーバー時間の決定
- ログシーケンス番号のギャップによるポイントインタイムリカバリ障害のトラブルシューティング
- Amazon RDS for SQL Server のデータベース名の表示を拒否または許可する
- Amazon RDS for SQL Server の一括ロード中の高速挿入の無効化
- Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースの削除
- マルチ AZ 配置の Amazon RDS for Microsoft SQL Server データベースの名前の変更
- Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット
- Amazon RDS for SQL Server のライセンス終了した DB インスタンスの復元
- Amazon RDS for SQL Server データベースのオフラインからオンラインへの切り替え
- Amazon RDS for SQL Server の変更データキャプチャの使用

- Amazon RDS 用 SQL Server エージェントの使用
- Amazon RDS for Microsoft SQL Server ログの使用方法
- Amazon RDS for SQL Server のトレースファイルとダンプファイルの使用

Amazon RDS で実行している Microsoft SQL Server DB インスタンスの tempdb データベースへのアクセス

Amazon RDS で実行している Microsoft SQL Server DB インスタンスの tempdb データベースにアクセスできます。tempdb に対してコードを実行するには、Microsoft SQL Server Management Studio (SSMS) 経由で Transact-SQL を使用するか、他の標準の SQL クライアントアプリケーションを使用します。DB インスタンスへの接続の詳細については、「SQL Server DB インスタンスへの接続」を参照してください。

DB インスタンスのマスターユーザーは、CONTROL への tempdb アクセスが付与されるため、tempdb データベースオプションを変更できます。マスターユーザーは、tempdb データベースの所有者ではありません。必要に応じて、マスターユーザーは他のユーザーに CONTROL アクセスを付与し、他のユーザーにも tempdb データベースオプションを変更することを許可できます。

> **Note**
>
> tempdb データベースに対して Database Console Command (DBCC) を実行することはできません。

tempdb データベースオプションの変更

Amazon RDS DB インスタンスで tempdb データベースのデータベースオプションを変更できます。変更可能なオプションの詳細については、Microsoft ドキュメントの「tempdb データベース」を参照してください。

ファイルの最大サイズオプションなどのデータベースオプションは、DB インスタンスの再起動後も保持されます。データベースオプションを変更して、データをインポートする際のパフォーマンスを最適化したり、ストレージ不足を防止したりすることができます。

データをインポートする際のパフォーマンスの最適化

大量のデータを DB インスタンス内にインポートする際のパフォーマンスを最適化するには、tempdb データベースの SIZE プロパティと FILEGROWTH プロパティに大きな数値を設定します。tempdb を最適化する方法の詳細については、Microsoft ドキュメントの「tempdb のパフォーマンスの最適化」を参照してください。

以下の例では、ファイルサイズを 100 GB に、ファイルの拡張単位を 10 パーセントに設定しています。

```
alter database[tempdb] modify file (NAME = N'templog', SIZE=100GB, FILEGROWTH = 10%)
```

ストレージの問題の防止

tempdb データベースによる使用可能なディスク容量の占有を防止するには、MAXSIZE プロパティを設定します。次の例では、プロパティを 2048 MB に設定しています。

```
alter database [tempdb] modify file (NAME = N'templog', MAXSIZE = 2048MB)
```

tempdb データベースの圧縮

Amazon RDS DB インスタンスの tempdb データベースを圧縮するには、2 つの方法があります。rds_shrink_tempdbfile プロシージャを使用するか、SIZE プロパティを設定できます。

rds_shrink_tempdbfile プロシージャの使用

Amazon RDS プロシージャ msdb.dbo.rds_shrink_tempdbfile を使用すると、tempdb データベースを圧縮できます。rds_shrink_tempdbfile が CONTROL にアクセスできる場合にのみ、tempdb を呼び出すことができます。rds_shrink_tempdbfile を呼び出すとき、DB インスタンスのダウンタイムはありません。

rds_shrink_tempdbfile プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
@temp_filename	SYSNAME	—	必須	圧縮するファイルの論理名。
@target_size	int	null	optional	ファイルの新しいサイズ (メガバイト単位)。

次の例では、tempdb データベース用にファイル名を取得します。

```
use tempdb;
GO

select name, * from sys.sysfiles;
GO
```

以下の例では、tempdb という名前の test_file データベースファイルを圧縮し、新しいサイズとして 10 メガバイトをリクエストしています。

```
exec msdb.dbo.rds_shrink_tempdbfile @temp_filename = N'test_file', @target_size = 10;
```

SIZE プロパティの設定

tempdb を設定して DB インスタンスを再起動することでも、SIZE データベースを圧縮できます。DB インスタンスの再起動の詳細については、「DB インスタンスの再起動」を参照してください。

次の例では、SIZE プロパティを 1024 MB に設定しています。

```
alter database [tempdb] modify file (NAME = N'templog', SIZE = 1024MB)
```

マルチ AZ 配置の TempDB 設定

データベースミラーリング (DBM) または Always On Availability Groups (AG) を使用して、RDS for SQL Server DB インスタンスがマルチ AZ 配置にある場合、tempdb データベースの使用については、以下の考慮事項に留意してください。

プライマリ DB インスタンスからセカンダリ DB インスタンスに tempdb データをレプリケートすることはできません。セカンダリ DB インスタンスにフェイルオーバーすると、そのセカンダリ DB インスタンスの tempdb は空になります。

プライマリ DB インスタンスからセカンダリ DB インスタンスに、ファイルのサイズ設定や自動拡張設定などの tempdb データベースオプションの設定を同期できます。tempDB 設定の同期は、すべての RDS for SQL Server バージョンでサポートされています。次のストアドプロシージャを使用して、tempdb 設定の自動同期を有効にできます。

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'TempDbFile';
```

> **Important**
>
> rds_set_system_database_sync_objects ストアドプロシージャを使用する前に、セカンダリ DB インスタンスではなく、プライマリ DB インスタンスで希望の tempdb 設定を行ってください。セカンダリ DB インスタンスで設定を変更した場合、自動同期を有効にすると、希望の tempdb 設定が削除される可能性があります。

次の関数を使用して、tempdb 設定の自動同期が有効になっているかどうかを確認できます。

```
SELECT * from msdb.dbo.rds_fn_get_system_database_sync_objects();
```

tempdb 設定の自動同期が有効な場合、object_class フィールドに戻り値があります。無効なときには、値は返されません。

次の関数を使用して、オブジェクトが UTC 時間で最後に同期された時刻を調べることができます。

```
SELECT * from msdb.dbo.rds_fn_server_object_last_sync_time();
```

例えば、tempdb 設定を 01:00 に変更してから rds_fn_server_object_last_sync_time 関数を実行した場合、last_sync_time として返される値は 01:00 より後であり、自動同期が発生したことを示します。

SQL Server Agent ジョブレプリケーションも使用している場合は、 @object_type パラメータで指定することで、SQL Agent ジョブと tempdb 設定の両方のレプリケーションを有効にできます。

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types =
 'SQLAgentJob,TempDbFile';
```

SQL Server Agent ジョブのレプリケーションの詳細については、「SQL Server エージェントジョブレプリケーションをオンにする」を参照してください。

rds_set_system_database_sync_objects ストアドプロシージャを使用して tempdb 設定変更が自動的に同期されるようにする代わりに、次のいずれかの手動方法を使用できます。

> **Note**
>
> rds_set_system_database_sync_objects ストアドプロシージャを使用して tempdb 設定の自動同期を有効にすることをお勧めます。自動同期を使用すると、tempdb 設定を変更するたびにこれらの手動タスクを実行する必要がなくなります。

- まず DB インスタンスを変更してマルチ AZ を無効にします。次に tempdb を変更し、最後にマルチ AZ を再度有効にします。この方法に伴うダウンタイムはありません。

  詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

- まず元のプライマリインスタンスで tempdb を変更します。次に手動でフェイルオーバーし、最後に新しいプライマリインスタンスで tempdb を変更します。このメソッドではダウンタイムが生じます。

  詳しくは、「DB インスタンスの再起動」を参照してください。

Database Engine Tuning Advisor を使用して Amazon RDS for SQL Server DB インスタンスのデータベースワークロードを分析する

Database Engine Tuning Advisor は、Microsoft によって提供されるクライアントアプリケーションで、データベースワークロードを分析し、実行するクエリのタイプに基づいて Microsoft SQL Server データベースの最適なインデックスセットを推奨します。SQL Server Management Studio と同様に、チューニングアドバイザーは SQL Server を実行している Amazon RDS DB インスタンスに接続するクライアントコンピュータから実行します。クライアントコンピュータは、独自のネットワーク内で、オンプレミスで実行するローカルコンピュータ、または Amazon RDS DB インスタンスと同じリージョンで実行している Amazon EC2 Windows インスタンスです。

このセクションでは、チューニングアドバイザーで分析するためにワークロードをキャプチャする方法を紹介します。Amazon RDS では SQL Server インスタンスへのホストアクセスが制限されるため、これがワークロードをキャプチャするための最適なプロセスです。詳細については、Microsoft のドキュメントの Database Engine Tuning Advisor を参照してください。

チューニングアドバイザーを使用するには、いわゆるワークロードをアドバイザーに提供する必要があります。ワークロードは、調整するデータベースに対して実行する一連の Transact-SQL ステートメントです。データベースエンジンチューニングアドバイザーは、データベースを調整する際のワークロード入力として、トレースファイル、トレーステーブル、Transact-SQL スクリプト、または XML ファイルを使用します。Amazon RDS を使用するときは、クライアントコンピュータ上のファイル、またはクライアントコンピュータにアクセス可能な Amazon RDS for SQL Server DB のデータベーステーブルがワークロードになります。ファイルまたはテーブルには、調整するデータベースに対するクエリが再生に適した形式で格納されている必要があります。

チューニングアドバイザーをもっとも効果的に機能させるには、ワークロードをできる限り実際的なものにする必要があります。DB インスタンスに対してトレースを実行することで、ワークロードのファイルまたはテーブルを生成できます。トレースの実行中に、DB インスタンスの負荷をシミュレートするか、正常な負荷でアプリケーションを実行できます。

トレースには、クライアント側とサーバー側の 2 種類があります。クライアント側トレースはセットアップが比較的容易で、SQL Server Profiler でキャプチャされたトレースイベントをリアルタイム

で監視することができます。サーバー側トレースは、セットアップが複雑で、複数の Transact-SQL スクリプトを作成する必要があります。さらに、トレースは Amazon RDS DB インスタンスのファイルに書き込まれるため、トレースによってストレージ領域が消費されます。この結果ストレージ領域が不足した場合、DB インスタンスは空き領域がない状態になり、使用不能になる可能性があるため、実行中のサーバー側トレースがどのくらいのストレージ領域を使用するかを追跡することが重要になります。

クライアント側トレースの場合、十分な量のトレースデータが SQL Server Profiler にキャプチャされると、ワークロードファイルを生成できます。そのためには、ローカルコンピュータのファイルにトレースを保存します。または、クライアントコンピュータから利用できる DB インスタンスのデータベーステーブルにトレースを保存します。クライアント側トレースを使用する主なデメリットは、大量の負荷がかかると、トレースですべてのクエリをキャプチャできない可能性があることです。この結果、データベースエンジンチューニングアドバイザーによって実行される分析の効果が低下します。大量の負荷の下でトレースを実行する必要があり、そのトレースセッション中にすべてのクエリを確実にキャプチャしたい場合は、サーバー側トレースを使用してください。

サーバー側トレースの場合、DB インスタンスのトレース ファイルを適切なワークロードファイルに入れるか、追跡完了後に DB インスタンスのテーブルにトレースを保存することができます。SQL Server Profiler を使用してトレースをローカルコンピュータのファイルに保存するか、チューニングアドバイザーで DB インスタンスのトレーステーブルから読み取ることができます。

SQL Server DB インスタンスでクライアント側トレースを実行する

SQL Server DB インスタンスでクライアント側トレースを実行するには

1. SQL Server Profiler を起動します。これは SQL Server インスタンスのフォルダの Performance Tools フォルダにインストールされます。クライアント側トレースを開始するには、トレース定義テンプレートをロードするか定義する必要があります。
2. SQL Server Profiler の [ファイル] メニューで、[新しいトレース] を選択します。[Connect to Server] ダイアログボックスで、トレースを実行するデータベースの DB インスタンスエンドポイント、ポート、マスターユーザー名、およびパスワードを入力します。
3. [Trace Properties] ダイアログボックスで、トレース名を入力し、トレース定義テンプレートを選択します。デフォルトテンプレート TSQL_Replay は、アプリケーションに標準で装備されています。このテンプレートを編集して、トレースを定義できます。[トレースのプロパティ] ダイアログボックスの [イベントの選択] タブで、イベントとイベント情報を編集します。

トレース定義テンプレートの詳細と SQL Server Profiler を使用してクライアント側トレースを指定する方法については、Microsoft ドキュメントの Database Engine Tuning Advisor を参照してください。

4. クライアント側トレースを開始し、DB インスタンスに対して実行される間、SQL クエリをリアルタイムで監視してください。
5. トレースが完了したら、[ファイル] メニューから [トレースの停止] を選択します。結果をファイルまたはトレーステーブルとして DB インスタンスに保存します。

SQL Server DB インスタンスでサーバー側トレースを実行する

サーバー側トレースを作成するスクリプトの作成は複雑になる可能性があるため、このドキュメントでは割愛します。このセクションでは、例として使用できるサンプルスクリプトを紹介します。クライアント側トレースと同様に、データベースエンジンチューニングアドバイザーを使用して開くことのできるワークロードファイルまたはトレーステーブルを作成することが目的です。

次に紹介する簡略化したサンプルスクリプトでは、サーバー側トレースを開始し、詳細をキャプチャしてワークロードファイルを作成します。トレースは、最初に D:\RDSDBDATA\Log ディレクトリの RDSTrace.trc ファイルに保存され、100 MB ごとにロールオーバーされて、それ以降のトレースファイルには RDSTrace_1.trc、RDSTrace_2.trc のように名前が付けられます。

```
DECLARE @file_name NVARCHAR(245) = 'D:\RDSDBDATA\Log\RDSTrace';
DECLARE @max_file_size BIGINT = 100;
DECLARE @on BIT = 1
DECLARE @rc INT
DECLARE @traceid INT

EXEC @rc = sp_trace_create @traceid OUTPUT, 2, @file_name, @max_file_size
IF (@rc = 0) BEGIN
   EXEC sp_trace_setevent @traceid, 10, 1, @on
   EXEC sp_trace_setevent @traceid, 10, 2, @on
   EXEC sp_trace_setevent @traceid, 10, 3, @on
 . . .
   EXEC sp_trace_setfilter @traceid, 10, 0, 7, N'SQL Profiler'
   EXEC sp_trace_setstatus @traceid, 1
   END
```

以下の例はトレースを停止するスクリプトです。前述のスクリプトで作成されるトレースは、明示的にトレースを停止するか、プロセスがディスク容量を使い果たすまで継続されます。

```
DECLARE @traceid INT
SELECT @traceid = traceid FROM ::fn_trace_getinfo(default)
WHERE property = 5 AND value = 1 AND traceid <> 1

IF @traceid IS NOT NULL BEGIN
   EXEC sp_trace_setstatus @traceid, 0
   EXEC sp_trace_setstatus @traceid, 2
END
```

サーバー側トレースの結果をデータベーステーブルに保存し、fn_trace_gettable 関数を使用することで、そのデータベーステーブルをチューニングアドバイザーのワークロードとして使用することができます。次のコマンドは、D:\rdsdbdata\Log ディレクトリにある RDSTrace.trc という名前の全ファイル (RDSTrace_1.trc などのすべてのロールオーバーファイルを含む) の結果を、現在のデータベースの RDSTrace という名前のテーブルにロードします。

```
SELECT * INTO RDSTrace
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace.trc', default);
```

特定のロールオーバーファイルをテーブル (例えば、RDSTrace_1.trc ファイル) に保存するには、fn_trace_gettable の最後のパラメータとしてロールオーバーファイルの名前を指定し、デフォルトの代わりに 1 を代入します。

```
SELECT * INTO RDSTrace_1
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace_1.trc', 1);
```

トレースを使用してチューニングアドバイザーを実行する

ローカルファイルまたはデータベーステーブルとしてトレースを作成したら、DB インスタンスに対してチューニングアドバイザーを実行できます。Amazon RDS でチューニングアドバイザーを使用するときのプロセスは、スタンドアロンのリモート SQL Server インスタンスを使用するときと同じです。クライアントマシンのチューニングアドバイザー UI を使用するか、コマンドラインから dta.exe ユーティリティを使用することができます。いずれの場合も、DB インスタンスのエンドポイントを使用して Amazon RDS DB インスタンスに接続し、チューニングアドバイザーを使用するときに、マスターユーザー名とマスターユーザーパスワードを指定する必要があります。

次のコード例では、エンドポイント **dta.cnazcmklsdei.us-east-1.rds.amazonaws.com** を持つ Amazon RDS DB インスタンスに対して dta.exe コマンドラインユーティリティを使用する方法をデモンストレーションします。この例には、マスターユーザー名 **admin** とマスターユーザーのパスワード **test** が含まれています。チューニングするサンプルデータベースは、**C:\RDSTrace.trc**

という名前が付いたマシンです。サンプルコマンドラインコードでは、**RDSTrace1** というトレースセッション名も指定し、ローカルマシンへの出力ファイルとして、SQL 出力スクリプトには **RDSTrace.sql**、結果ファイルには **RDSTrace.txt**、分析の XML ファイルには **RDSTrace.xml** という名前を指定します。また、RDSDTA データベースに **RDSTraceErrors** という名前のエラーテーブルが指定されます。

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -
if C:\RDSTrace.trc -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\
 RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

次は同じコマンドラインコードの例ですが、入力ワークロードがリモート Amazon RDS インスタンスにある **RDSTrace** データベースの **RDSDTA** というテーブルである点が異なります。

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -it
 RDSDTA.dbo.RDSTrace -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\
 RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

dta ユーティリティのコマンドラインパラメータの一覧については、Microsoft のドキュメントの dta Utility を参照してください。

db_owner の Amazon RDS for SQL Server データベースの **rdsa** アカウントへの変更

RDS for SQL Server DB インスタンスでデータベースを作成または復元すると、Amazon RDS はデータベースの所有者を `rdsa` に設定します。マルチ AZ 配置で SQL Server データベースミラーリング (DBM) または Always On 可用性グループ (AG) を使用している場合、Amazon RDS はセカンダリ DB インスタンスのデータベースの所有者を `NT AUTHORITY\SYSTEM` に設定します。セカンダリ DB インスタンスがプライマリロールに昇格されるまで、セカンダリデータベースの所有者を変更することはできません。ほとんどの場合、クエリの実行時にデータベースの所有者を `NT AUTHORITY\SYSTEM` に設定しても問題はありませんが、実行に昇格アクセス許可を必要とする `sys.sp_updatestats` などのシステムストアドプロシージャの実行時にエラーが発生する可能性があります。

次のクエリを使用して、`NT AUTHORITY\SYSTEM` が所有するデータベースの所有者を特定できます。

```
SELECT name FROM sys.databases WHERE SUSER_SNAME(owner_sid) = 'NT AUTHORITY\SYSTEM';
```

Amazon RDS ストアドプロシージャ rds_changedbowner_to_rdsa を使用して、データベースの所有者を rdsa に変更できます。rds_changedbowner_to_rdsa では、master, model, msdb, rdsadmin, rdsadmin_ReportServer, rdsadmin_ReportServerTempDB, SSISDB の各データベースは使用できません。

データベースの所有者を rdsa に変更するには、rds_changedbowner_to_rdsa ストアドプロシージャを呼び出してデータベース名を指定します。

Example使用例:

```
exec msdb.dbo.rds_changedbowner_to_rdsa 'TestDB1';
```

以下のパラメータは必須です。

- @db_name — データベースの所有者を rdsa に変更する対象のデータベースの名前。

> **Important**
>
> rds_changedbowner_to_rdsa を使用して、データベースの所有権を rdsa 以外のログインに変更することはできません。例えば、データベースを作成したログインに所有権を変更することはできません。他のデータベースユーザーを使用してメンバーシップを付与できない場合に、マスターユーザーの db_owner ロールの失われたメンバーシップを復元するには、マスターユーザーのパスワードをリセットして、db_owner ロールのメンバーシップを取得します。詳細については、「Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット」を参照してください。

Amazon RDS for Microsoft SQL Server の照合と文字セットの管理

このトピックでは、Amazon RDS で Microsoft SQL Server の照合順序と文字セットを管理する方法について説明します。データベースの作成中に照合順序を設定し、後で変更する方法を説明し、言語とロケールの要件に基づいてテキストデータを適切に処理できるようにします。さらに、Amazon RDS の SQL Server 環境で互換性とパフォーマンスを維持するためのベストプラクティスについても説明します。

SQL Server は、複数のレベルで照合をサポートします。DB インスタンスを作成するときに、デフォルトのサーバー照合を設定します。照合は、データベース、テーブル、または列レベルでオーバーライドできます。

トピック

- Microsoft SQL Server のサーバーレベルの照合
- Microsoft SQL Server のデータベースレベルの照合

Microsoft SQL Server のサーバーレベルの照合

Microsoft SQL Server DB インスタンスを作成するときに、使用するサーバーの照合順序を設定できます。別の照合を選択しない場合、サーバーレベルの照合はデフォルトで SQL_Latin1_General_CP1_CI_AS になります。サーバー照合は、デフォルトですべてのデータベースとデータベースオブジェクトに適用されます。

> **Note**
>
> DB スナップショットから復元する場合は、照合順序を変更できません。

Amazon RDS は現在、以下のサーバー照合をサポートしています。

照合	説明
Arabic_CI_AS	アラビア語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Chinese_PRC_BIN2	Chinese-PRC、バイナリコードポイントのソート順序
Chinese_PRC_CI_AS	中国語 - 中華人民共和国、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Chinese_Taiwan_Stroke_CI_AS	繁体字中国語 (台湾)、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
Danish_Norwegian_CI_AS	デンマーク-ノルウェー語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Danish_Norwegian_CI_AS_KS	デンマーク-ノルウェー語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Danish_Norwegian_CI_AS_KS_WS	デンマーク-ノルウェー語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり
Danish_Norwegian_CI_AS_WS	デンマーク-ノルウェー語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別あり
Danish_Norwegian_CS_AI	デンマーク-ノルウェー語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Danish_Norwegian_CS_AI_KS	デンマーク-ノルウェー語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別あり、全角半角の区別なし
Finnish_Swedish_100_BIN	Finnish-Swedish-100、バイナリソート
Finnish_Swedish_100_BIN2	Finnish-Swedish-100、バイナリコードポイント比較ソート

照合	説明
Finnish_Swedish_100_CI_AI	フィンランド-スウェーデン語-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Finnish_Swedish_100_CI_AS	フィンランド-スウェーデン語-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Finnish_Swedish_CI_AS	フィンランド語、スウェーデン語、およびスウェーデン語 (フィンランド)、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
French_CI_AS	フランス語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Greek_CI_AS	ギリシャ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Greek_CS_AS	ギリシャ語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Hebrew_BIN	ヘブライ語、バイナリソート
Hebrew_CI_AS	ヘブライ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_BIN	日本語、バイナリソート

照合	説明
Japanese_CI_AS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_CS_AS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Japanese_XJIS_140_CI_AS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別なし
Japanese_XJIS_140_CI_AS_KS_VSS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CI_AS_VSS	日本語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし、補足文字、バリエーションセレクタの区別あり
Japanese_XJIS_140_CS_AS_KS_WS	日本語、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別あり、補足文字、バリエーションセレクタの区別なし
Korean_Wansung_CI_AS	韓国語 (Wansung)、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_100_BIN	Latin1-General-100、バイナリソート
Latin1_General_100_BIN2	Latin1-General-100、バイナリコードポイントソート

照合	説明
Latin1_General_100_BIN2_UTF8	Latin1-General-100、バイナリコードポイントソート順序、UTF-8 エンコード
Latin1_General_100_CI_AS	Latin1-General-100、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_100_CI_AS_SC_UTF8	Latin1-General-100、大文字と小文字を区別しない、アクセントを区別する、補助文字、UTF-8 エンコード
Latin1_General_BIN	Latin1-General、バイナリソート
Latin1_General_BIN2	Latin1-General、バイナリ、コードポイントソート
Latin1_General_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Latin1_General_CI_AS_KS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別あり、全角半角の区別なし
Latin1_General_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Modern_Spanish_CI_AS	現代スペイン語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

照合	説明
Polish_CI_AS	ポーランド語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
SQL_1xCompat_CP850_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 49 (非 Unicode データの場合)
SQL_Latin1_General_CP1_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 54 (非 Unicode データの場合)
SQL_Latin1_General_CP1_CI_AS (デフォルト)	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 52 (非 Unicode データの場合)
SQL_Latin1_General_CP1_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1252 の SQL Server ソート順 51 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP437_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 437 の SQL Server ソート順 34 (非 Unicode データの場合)
SQL_Latin1_General_CP850_BIN	Latin1-General、バイナリソート順 (Unicode データの場合)、コードページ 850 の SQL Server ソート順 40 (非 Unicode データの場合)
SQL_Latin1_General_CP850_BIN2	Latin1-General、バイナリコードポイントソート (Unicode データの場合)、コードページ 850 の SQL Server ソート順 40 (非 Unicode データの場合)
SQL_Latin1_General_CP850_CI_AI	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別なし、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 44 (非 Unicode データの場合)
SQL_Latin1_General_CP850_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 42 (非 Unicode データの場合)
SQL_Latin1_General_Pref_CP850_CI_AS	Latin1-General-Pref、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 850 の SQL Server ソート順 183 (非 Unicode データの場合)

照合	説明
SQL_Latin1_General_CP1256_CI_AS	Latin1-General、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1256 の SQL Server ソート順 146 (非 Unicode データの場合)
SQL_Latin1_General_CP1255_CS_AS	Latin1-General、大文字と小文字の区別あり、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし (Unicode データの場合)、コードページ 1255 の SQL Server ソート順 137 (非 Unicode データの場合)
Thai_CI_AS	タイ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし
Turkish_CI_AS	トルコ語、大文字と小文字の区別なし、アクセント (濁音、破裂音) の区別あり、ひらがな片仮名の区別なし、全角半角の区別なし

AWS CLI を使用して、サポートされている照合順序のリストをプログラムで取得することもできます。

```
aws rds describe-db-engine-versions --engine sqlserver-ee --list-supported-character-
sets --query 'DBEngineVersions[].SupportedCharacterSets[].CharacterSetName' | sort -u
```

照合を選択

- Amazon RDS のコンソールを使用している場合、新しい DB インスタンスを作成するときには、[Additional configuration] (追加設定) を選択し、[Collation] (照合) フィールドに照合を入力します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- AWS CLI を使用している場合は、`--character-set-name` コマンドで `create-db-instance` オプションを使います。詳細については、create-db-instance を参照してください。

- Amazon RDS API を使用している場合は、CharacterSetName 操作で CreateDBInstance パラメータを使用します。詳細については、CreateDBInstance を参照してください。

Microsoft SQL Server のデータベースレベルの照合

デフォルト照合順序は、新しいデータベースまたはデータベースオブジェクトを作成する際に、照合順序を上書きすることにより、データベース、テーブル、または列レベルで変更できます。例えば、デフォルトのサーバー照合が SQL_Latin1_General_CP1_CI_AS の場合は、Mohawk 照合に対応できるように、これを Mohawk_100_CI_AS に変更することができます。クエリの引数も、必要に応じて他の照合順序を使用するために型変換できます。

例えば、次のクエリは、AccountName 列のデフォルト照合順序を Japanese_CI_AS に変更します。

```
CREATE TABLE [dbo].[Account]
 (
     [AccountID] [nvarchar](10) NOT NULL,
     [AccountName] [nvarchar](100) COLLATE Mohawk_100_CI_AS NOT NULL
 ) ON [PRIMARY];
```

Microsoft SQL Server DB エンジンは、組み込みの NCHAR、NVARCHAR、および NTEXT データ型で Unicode をサポートします。例えば、CJK サポートが必要な場合は、データベースとテーブルを作成するときに、文字列ストレージに対してこれらの Unicode データ型を使用して、デフォルトサーバー照合順序を上書きします。以下のリンクから、SQL Server に対する照合順序と Unicode のサポートに関連する Microsoft の情報を参照できます。

- 照合順序の使用
- 照合順序とインターナショナル対応に関する用語
- SQL Server 照合順序の使用
- データベースとデータベースエンジンアプリケーションの国際化に関する注意点

Amazon RDS for SQL Server のデータベースユーザーの作成

Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースユーザーを作成するには、次の例のように T-SQL スクリプトを実行します。SQL Server Management Suite (SSMS) などのアプリケーションを使用します。DB インスタンスの作成時に作成されたマスターユーザーとして DB インスタンスにログインします。

```
--Initially set context to master database
```

```
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
```

ロールにデータベースユーザーを追加する例については、SQLAgentUser ロールへのユーザーの追加を参照してください。

> **Note**
>
> ユーザーを追加する際に許可エラーが発生した場合は、DB インスタンスのマスターユーザーパスワードを変更することで許可を復元できます。詳細については、「Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット」を参照してください。
> アプリケーションでマスターユーザーのアクセス許可をクローンすることはベストプラクティスではありません。詳細については、「Amazon RDS for SQL Server でマスターユーザーの権限をクローンする方法」を参照してください。

Amazon RDS for SQL Server データベースの復旧モデルの決定

Amazon RDS では、復旧モデル、保持期間、およびデータベースステータスがリンクされています。

これらの設定のいずれかを変更する前に、変更の影響を理解しておくことが重要です。各設定が他の設定に影響を与える場合があります。次に例を示します。

- バックアップ保持を有効にしている状態でデータベースの復旧モデルを SIMPLE または BULK_LOGGED に変更すると、Amazon RDS で復旧モデルは 5 分以内に FULL にリセットされます。これに伴って、RDS で DB インスタンスのスナップショットも作成されます。
- バックアップ保持期間を 0 日に設定すると、RDS で復旧モードは SIMPLE に設定されます。

- バックアップ保持期間を 0 日に設定した状態で、データベースの復旧モデルを SIMPLE から他のオプションに変更すると、RDS で復旧モデルは SIMPLE にリセットされます。

> **Important**
>
> マルチ AZ インスタンスの復旧モデルは、絶対に変更しないでください。ALTER DATABASE などを使用して変更できそうに思える場合でも、変更してはなりません。バックアップ保持期間 (FULL 復旧モデル) は、マルチ AZ に必須です。この復旧モデルを変更すると、RDS によって即座に FULL に戻されます。
> この自動リセットにより、RDS でミラーが完全に再構築されます。この再構築中は、ミラーでフェイルオーバーの準備が整うまで、データベースの可用性が約 30～90 分間低下します。DB インスタンスも、シングル AZ からマルチ AZ に切り替える場合と同じように、パフォーマンスが低下します。パフォーマンスが低下する期間は、データベースのストレージサイズに応じて異なります。データベースのストレージサイズが大きいほど、低下する期間が長くなります。

SQL Server の復旧モデルの詳細については、Microsoft ドキュメントの復旧モデル (SQL Server) を参照してください。

Amazon RDS for SQL Server の最終フェイルオーバー時間の決定

最後のフェイルオーバー時間を確認するには、次のストアドプロシージャを使用します。

```
execute msdb.dbo.rds_failover_time;
```

このプロシージャは、次の情報を返します。

出力パラメータ	説明
errorlog_available_from	ログディレクトリでエラーログが使用可能になる時間が表示されます。
recent_failover_time	エラーログに最新のフェイルオーバー時間が含まれている場合は、その時間が表示されます。それ以外の場合は、null が表示されます。

> **Note**
>
> ストアドプロシージャは、最新のフェイルオーバー時間を取得するために、ログディレクトリ内のすべての使用可能な SQL Server エラーログを検索します。フェイルオーバーメッセージが SQL Server によって上書きされている場合、フェイルオーバー時間は取得されません。

Example最近のフェイルオーバーがない例

この例は、エラーログに最近のフェイルオーバーが存在しない場合の出力を示しています。2020-04-29 23:59:00.01 以降、フェイルオーバーは発生していません。

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	null

Example最近のフェイルオーバーの

この例は、エラーログにフェイルオーバーが存在する場合の出力を示しています。最新のフェイルオーバーは、2020-05-05 18:57:51.89 に発生しています。

errorlog_available_from	recent_failover_time
2020-04-29 23:59:00.0100000	2020-05-05 18:57:51.8900000

ログシーケンス番号のギャップによるポイントインタイムリカバリ障害のトラブルシューティング

RDS for SQL Server でポイントインタイムリカバリ (PITR) を試みると、ログシーケンス番号 (LSN) のギャップが原因で障害が発生する可能性があります。これらのギャップにより、RDS はデータベースを要求された時間に復元できなくなり、RDS は復元インスタンスを `incompatible-restore` 状態にします。

この問題の一般的な原因は次のとおりです。

- データベース復旧モデルへの手動変更。
- トランザクションログのバックアップを完了するためのリソースが不足しているため、RDS によって自動復旧モデルが変更される。

データベース内の LSN ギャップを特定するには、次のクエリを実行します。

```
SELECT * FROM msdb.dbo.rds_fn_list_tlog_backup_metadata(database_name)
ORDER BY backup_file_time_utc desc;
```

LSN ギャップを検出したら、次を実行できます。

- LSN ギャップの前に復元ポイントを選択します。
- 次のインスタンスバックアップが完了した後、ある時点まで待機して復元します。

この問題を回避するには、RDS for SQL Server データベースの復旧モデルを手動で変更しないことをお勧めします。これは、インスタンスの耐久性を妨げてしまうためです。また、定期的なトランザクションログのバックアップを確保するために、ワークロードに十分なリソースを持つインスタンスタイプを選択することをお勧めします。

トランザクションログ管理の詳細については、「Microsoft SQL Server ドキュメント」の「SQL Server トランザクションログのアーキテクチャと管理ガイド」を参照してください。

Amazon RDS for SQL Server のデータベース名の表示を拒否または許可する

マスターユーザーは DENY VIEW ANY DATABASE TO *LOGIN* を設定して、ユーザーにデータベースを非表示にすることはできません。このアクセス許可を変更するには、代わりに次のストアドプロシージャを使用します。

- *LOGIN* へのデータベースの表示アクセスを拒否する:

```
EXEC msdb.dbo.rds_manage_view_db_permission @permission='DENY',
 @server_principal='LOGIN'

go
```

- *LOGIN* へのデータベースの表示アクセスを許可する:

```
EXEC msdb.dbo.rds_manage_view_db_permission @permission='GRANT',
 @server_principal='LOGIN'

go
```

このストアドプロシージャを使用する場合は、次の点を考慮してください。

- データベース名は、SSMS および内部 DMV (動的管理ビュー) には非表示になります。ただし、データベース名は監査、ログ、メタデータテーブルでは引き続き表示可能です。これらは保護されている VIEW ANY DATABASE サーバーアクセス許可です。詳細については、「サーバーの権限の拒否」を参照してください。
- 権限が GRANT (許可) に戻されると、*LOGIN* はすべてのデータベースを表示できます。
- *LOGIN* を削除して再作成すると、LOGIN に関連する表示アクセス許可が ALLOW にリセットされます。
- マルチ AZ インスタンスの場合は、プライマリホストの *LOGIN* にのみ DENY または GRANT アクセス許可を設定します。変更はセカンダリホストに自動的に伝播されます。
- このアクセス許可は、ログインがデータベース名を表示できるかどうかのみを変更します。ただし、データベースとその内部のオブジェクトへのアクセスは個別に管理されます。

Amazon RDS for SQL Server の一括ロード中の高速挿入の無効化

SQL Server 2016 以降では、高速挿入はデフォルトで有効になっています。高速挿入では、データベースが単純または一括ログ復旧モデルにあるときに発生する最小限のログを活用して、挿入パフォーマンスを最適化します。高速挿入では、それぞれの一括ロードバッチが新しいエクステントを取得し、使用可能な空き領域がある既存のエクステントの配分ルックアップをバイパスして、挿入パフォーマンスを最適化します。

ただし、高速挿入では、バッチサイズが小さい一括ロードにより、オブジェクトにより消費される未使用の領域が増加する可能性があります。バッチサイズを増やすことができない場合は、トレースフラグ 692 を有効にすると、未使用の予約領域を減らすことができますが、パフォーマンスは低下します。このトレースフラグを有効にすると、ヒープまたはクラスター化インデックスにデータを一括でロードする際の高速挿入が無効になります。

DB パラメータグループを使用して、起動パラメータとしてトレースフラグ 692 を有効にします。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

トレースフラグ 692 は、SQL Server 2016 以降での Amazon RDS でサポートされています。トレースフラグの詳細については、Microsoft ドキュメントの DBCC TRACEON - Trace Flags を参照してください。

Amazon RDS for Microsoft SQL Server DB インスタンスのデータベースの削除

単一 AZ または マルチ AZ 配置で Microsoft SQL Server を実行している Amazon RDS DB インスタンスのデータベースを削除できます。データベースを削除するには、次のコマンドを使用します。

```
--replace your-database-name with the name of the database you want to drop
EXECUTE msdb.dbo.rds_drop_database  N'your-database-name'
```

> **Note**
>
> コマンドでストレートの単一の引用を使用します。スマート引用はエラーを発生させます。

この手順を使用してデータベースを削除すると、Amazon RDS は、このデータベースへのすべての既存の接続とデータベースのバックアップ履歴を削除します。

他のユーザーにバックアップと復元を許可するには、次の手順に従います。

```
USE master
GO
CREATE LOGIN user1 WITH PASSWORD=N'changeThis', DEFAULT_DATABASE=master,
 CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE msdb
GO
CREATE USER user1 FOR LOGIN user1
GO
use msdb
GO
GRANT EXECUTE ON msdb.dbo.rds_backup_database TO user1
GO
GRANT EXECUTE ON msdb.dbo.rds_restore_database TO user1
GO
```

マルチ AZ 配置の Amazon RDS for Microsoft SQL Server データベースの名前の変更

マルチ AZ を使用する Microsoft SQL Server データベースインスタンスの名前を変更するには、次の手順を使用します。

1. 最初に、DB インスタンスのマルチ AZ を無効にします。
2. `rdsadmin.dbo.rds_modify_db_name` を実行して、データベースの名前を変更します。
3. 次に、DB インスタンスのマルチ AZ ミラーリングまたは AlwaysOn 可用性グループをオンにして、元の状態に戻します。

詳細については、「Microsoft SQL Server DB インスタンスへのマルチ AZ の追加」を参照してください。

> **Note**
>
> インスタンスでマルチ AZ を使用していない場合は、`rdsadmin.dbo.rds_modify_db_name` の実行前または実行後に設定を変更する必要はありません。
> リードレプリカソースインスタンスのデータベースの名前を変更することはできません。

例: 次の例では、`rdsadmin.dbo.rds_modify_db_name` に保存された手順でデータベースの名前を **MOO** から **ZAR** に変更します。これはステートメント `DDL ALTER DATABASE [MOO] MODIFY NAME = [ZAR]` の実行に似ています。

```
EXEC rdsadmin.dbo.rds_modify_db_name N'MOO', N'ZAR'
GO
```

Amazon RDS for SQL Server のマスターユーザーの db_owner ロールメンバーシップのリセット

RDS for SQL Server データベースの db_owner ロールメンバーシップからマスターユーザーをロックし、他のデータベースユーザーがメンバーシップを付与できない場合、DB インスタンスのマスターユーザーパスワードを変更することで、失われたメンバーシップを復元できます。

DB インスタンスのマスターユーザーパスワードを変更することで、RDS は誤って取り消された可能性のある DB インスタンス内のデータベースに db_owner メンバーシップを付与します。DB インスタンスのパスワードは、Amazon RDS コンソール、AWS CLI コマンドの modify-db-instance、または ModifyDBInstance API オペレーションを使用して変更できます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS for SQL Server のライセンス終了した DB インスタンスの復元

Microsoft は、Microsoft ライセンスモビリティ情報を報告しない Amazon RDS の一部のお客様について、DB インスタンスを削除するよう要求しています。Amazon RDS では、これらの DB インスタンスのスナップショットを作成するので、そのスナップショットから、ライセンスを含んだモデルを使用する新しい DB インスタンスを復元できます。

Standard Edition のスナップショットから、Standard Edition または Enterprise Edition のいずれかに復元できます。

Enterprise Edition のスナップショットから、Standard Edition または Enterprise Edition のいずれかに復元できます。

Amazon RDS がインスタンスの最終スナップショットを作成した後で、SQL Server のスナップショットから復元するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[Snapshots] を選択します。
3. SQL Server DB インスタンスのスナップショットを選択します。Amazon RDS が、DB インスタンスの最終スナップショットを作成します。終了したインスタンスのスナップショットの名前は *instance_name*-final-snapshot の形式です。例えば、DB インスタンス名が **mytest.cdxgahslksma.us-east-1.rds.com** である場合、最終スナップショットは **mytest-final-snapshot** という名前で、元の DB インスタンスと同じ AWS リージョン内に配置されます。
4. [アクション]、[スナップショットの復元] の順に選択します。

   [Restore DB Instance] ウィンドウが表示されます。
5. [ライセンスモデル] で、[ライセンス込み] を選択します。
6. 使用する SQL Server DB エンジンを選択します。

7. [DB インスタンス識別子] に、復元された DB インスタンスの名前を入力します。
8. [DB インスタンスの復元] を選択します。

スナップショットから復元する方法の詳細については、「DB インスタンスへの復元」を参照してください。

Amazon RDS for SQL Server データベースのオフラインからオンラインへの切り替え

Amazon RDS DB インスタンスの Microsoft SQL Server データベースを `OFFLINE` から `ONLINE` に切り替えることができます。

SQL Server メソッド	Amazon RDS方法
ALTER DATABASE *db_name* SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online *db_name*

Amazon RDS for SQL Server の変更データキャプチャの使用

Amazon RDS は、Microsoft SQL Server で実行されている DB インスタンスの変更データキャプチャをサポートします。CDC は、テーブル内のデータに行われる変更をキャプチャします。また、各変更に関するメタデータを保存します。これにより、後にアクセスできるようになります。CDC の動作の詳細については、Microsoft ドキュメントの「変更データキャプチャ」を参照してください。Amazon RDS DB インスタンスで CDC を使用するには、`msdb.dbo.rds_cdc_enable_db` を実行して、データベース上で有効にします。CDC が有効になると、該当データベースの db_owner であるユーザーは誰でも、そのデータベースのテーブルの CDC を有効または無効にすることができます。

> **Important**
>
> 復元中、CDC は無効になります。関連するメタデータはすべて、データベースより自動的に削除されます。これは、スナップショット復元とポイントインタイムリストアに適用されます。これらの復元のいずれかを実行すると、CDC を再度有効にして、追跡するテーブルを再度指定できます。

DB インスタンスの CDC を有効にするには、msdb.dbo.rds_cdc_enable_db ストアドプロシージャを実行します。

```
exec msdb.dbo.rds_cdc_enable_db 'database_name'
```

DB インスタンスの CDC を無効にするには、msdb.dbo.rds_cdc_disable_db ストアドプロシージャを実行します。

```
exec msdb.dbo.rds_cdc_disable_db 'database_name'
```

以下の手順を使用して、ユーザーに CDC アクセス許可を付与します。

```
go
  GRANT EXECUTE ON msdb.dbo.rds_cdc_enable_db TO User1
  GRANT EXECUTE ON msdb.dbo.rds_cdc_disable_db TO User1
```

トピック

- 変更データキャプチャを使用したテーブルの追跡
- 変更データキャプチャのジョブ
- マルチ AZ インスタンスの変更データキャプチャ

変更データキャプチャを使用したテーブルの追跡

CDC がデータベースで有効になったら、特定のテーブルの追跡を開始できます。追跡するテーブルを選択するには、sys.sp_cdc_enable_table を実行します。

```
--Begin tracking a table
exec sys.sp_cdc_enable_table
   @source_schema           = N'source_schema'
,  @source_name             = N'source_name'
,  @role_name               = N'role_name'

--The following parameters are optional:

--, @capture_instance       = 'capture_instance'
--, @supports_net_changes   = supports_net_changes
--, @index_name             = 'index_name'
--, @captured_column_list   = 'captured_column_list'
```

```
--, @filegroup_name         = 'filegroup_name'
--, @allow_partition_switch = 'allow_partition_switch'
;
```

テーブルの CDC 設定を表示するには、sys.sp_cdc_help_change_data_capture を実行します。

```
--View CDC configuration
exec sys.sp_cdc_help_change_data_capture

--The following parameters are optional and must be used together.
--  'schema_name', 'table_name'
;
```

CDC テーブル、関数、ストアドプロシージャの詳細については、SQL Server のドキュメントの以下を参照してください。

- 変更データキャプチャのストアドプロシージャ (Transact-SQL)
- 変更データキャプチャの関数 (Transact-SQL)
- 変更データキャプチャのテーブル (Transact-SQL)

変更データキャプチャのジョブ

CDC が有効になったら、SQL Server により CDC ジョブが作成されます。データベースの所有者 (db_owner) は CDC ジョブを表示、作成、変更、および削除することができます。ただし、所有者は RDS システムアカウントです。そのため、これらのジョブをネイティブのビュー、プロシージャ、または SQL Server Management Studio で表示することはできません。

データベースの CDC の動作をコントロールするには、SQL Server のネイティブプロシージャ (sp_cdc_enable_table、sp_cdc_start_job など) を使用します。CDC ジョブパラメータ (maxtrans、maxscans など) を変更するには、sp_cdc_change_jobs を使用できます。

CDC ジョブに関する詳細情報を取得するには、次の動的管理ビューに対してクエリを実行します。

- sys.dm_cdc_errors
- sys.dm_cdc_log_scan_sessions
- sysjobs
- sysjobhistory

マルチ AZ インスタンスの変更データキャプチャ

マルチ AZ インスタンスで CDC を使用する場合は、ミラーの CDC ジョブ設定がプリンシパルのいずれかと一致していることを確認します。CDC ジョブは、database_id にマッピングされています。セカンダリのデータベース ID がプリンシパルと異なる場合、そのジョブは正しいデータベースと関連付けられません。フェイルオーバー後のエラーを防ぐために、RDS によってジョブは削除され、新しいプリンシパルに再作成されます。再作成されたジョブでは、フェイルオーバー前に記録されたパラメータがプリンシパルによって使用されます。

このプロセスはすぐに実行されますが、RDS によって変更される前に CDC ジョブを実行できる場合があります。プリンシパルとセカンダリレプリカの間で一貫したパラメータを維持するには、次の 3 つの方法があります。

- CDC を有効にしたすべてのデータベースに対して、同じジョブパラメータを使用する。
- CDC ジョブ設定を変更する前に、マルチ AZ インスタンスを単一 AZ に変換する。
- プリンシパルでパラメータを変更する場合は、必ず手動で転送する。

フェイルオーバー後に CDC ジョブを再作成するために使用される CDC パラメータを表示し、定義するには、rds_show_configuration および rds_set_configuration を使用します。

次の例では、cdc_capture_maxtrans のために設定された値を返します。RDS_DEFAULT に設定されているパラメータの場合は、RDS によって自動的に値が設定されます。

```
-- Show configuration for each parameter on either primary and secondary replicas.
exec rdsadmin.dbo.rds_show_configuration 'cdc_capture_maxtrans';
```

セカンダリの構成を設定するには、rdsadmin.dbo.rds_set_configuration を実行します。この手順では、セカンダリサーバーのすべてのデータベース向けにパラメータ値を設定します。これらの設定は、フェイルオーバー後にのみ使用されます。次の例では、CDC キャプチャのジョブの maxtrans を *1000* に設定します。

```
--To set values on secondary. These are used after failover.
exec rdsadmin.dbo.rds_set_configuration 'cdc_capture_maxtrans', 1000;
```

プリンシパルの CDC ジョブパラメータを設定するには、代わりに sys.sp_cdc_change_jobs を使用します。

Amazon RDS 用 SQL Server エージェントの使用

Amazon RDS では、Microsoft SQL Server の Enterprise、Standard、または Web エディションを実行している DB インスタンスの SQL Server エージェントを使用できます。SQL Server エージェントは、ジョブと呼ばれるスケジュールされた管理タスクを実行する Microsoft Windows サービスです。SQL Server エージェントを使用して T-SQL ジョブを実行し、SQL Server DB インスタンスでインデックスの再構築、データ破損の確認、およびデータの集計を行うことができます。

SQL Server DB インスタンスを作成すると、マスターユーザーが `SQLAgentUserRole` ロールに登録されます。

SQL Server エージェントは、指定したイベントに対応し、またはオンデマンドで、スケジュールに従ってジョブを実行できます。詳細については、Microsoft ドキュメントの SQL Server エージェントを参照してください。

> **Note**
>
> DB インスタンスのメンテナンスおよびバックアップウィンドウ中に実行されるジョブをスケジュールすることは避けてください。AWS によって起動されたメンテナンスおよびバックアッププロセスによって、ジョブが中断されたり、ジョブがキャンセルされたりする可能性があります。
>
> マルチ AZ 配置では、ジョブのレプリケーション機能がオンになっているとき、SQL Server エージェントジョブは、プライマリホストからセカンダリホストにレプリケートされます。詳細については、「SQL Server エージェントジョブレプリケーションをオンにする」を参照してください。
>
> マルチ AZ 配置は、SQL Server エージェントジョブの数が 10,000 に制限されています。制限の引き上げが必要な場合は、サポート に連絡して緩和をリクエストしてください。AWS サポート センターのページを開き、必要に応じてサインインし、[ケースの作成] を選択します。[Service Limit increase] (サービス制限の緩和) を選択します。フォームに入力して送信します。

SQL Server Management Studio (SSMS) で個々の SQL Server エージェントジョブの履歴を表示するには、オブジェクトエクスプローラーを開き、ジョブを右クリックして、[View History] (履歴を表示) を選択します。

SQL Server エージェントは DB インスタンスのマネージドホストで実行されているため、一部のアクションはサポートされていません。

- ActiveX、Windows コマンドシェル、または Windows PowerShell を使用した、レプリケーションジョブの実行やコマンドラインスクリプトの実行はサポートされません。
- SQL Server エージェントを手動で起動、停止、または再起動することはできません。
- SQL Server エージェントを使用した E メール通知は、DB インスタンスからは使用できません。
- SQL Server エージェントのアラートとオペレータはサポートされていません。
- SQL Server エージェントを使用したバックアップの作成はサポートされていません。Amazon RDS を使用して DB インスタンスをバックアップします。
- RDS for SQL Server は、現在、SQL Server エージェントトークンの使用をサポートしていません。

SQL Server エージェントジョブレプリケーションをオンにする

SQL Server エージェントジョブレプリケーションは、次のストアドプロシージャを使用して有効にできます。

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types = 'SQLAgentJob';
```

ストアドプロシージャは、Amazon RDS for SQL Server でサポートされているすべての SQL Server バージョンで実行できます。以下のカテゴリのジョブがレプリケートされます。

- [未分類 (ローカル)]
- [未分類 (マルチサーバー)]
- [未分類]
- データコレクター
- データベースエンジンチューニングアドバイザー
- データベースメンテナンス
- フルテキスト

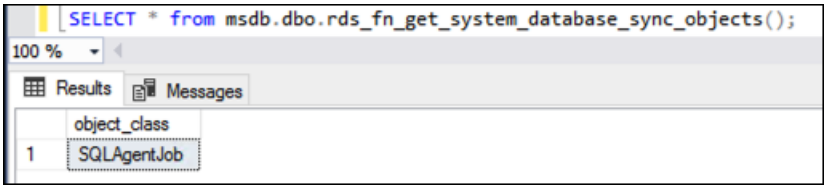
T-SQL ジョブステップを使用するジョブのみがレプリケートされます。SQL Server Integration Services (SSIS)、SQL Server Reporting Services (SSRS)、レプリケーション、PowerShell などのステップタイプのジョブはレプリケートされません。データベースメールとサーバーレベルのオブジェクトを使用するジョブはレプリケートされません。

> **Important**
>
> プライマリホストは、レプリケーションの信頼できるソースです。ジョブのレプリケーションをオンにする前に、SQL Server エージェントのジョブがプライマリであることを確認してください。これを行わない場合、新しいジョブがセカンダリホスト上にあるときにこの機能をオンにすると、SQL Server エージェントのジョブが削除される可能性があります。

次の関数を使用して、レプリケーションがオンになっているかどうかを確認できます。

```
SELECT * from msdb.dbo.rds_fn_get_system_database_sync_objects();
```

SQL Server エージェントジョブがレプリケートされている場合、T-SQL クエリは以下を返します。レプリケートしていない場合は、object_class に対して何も返しません。

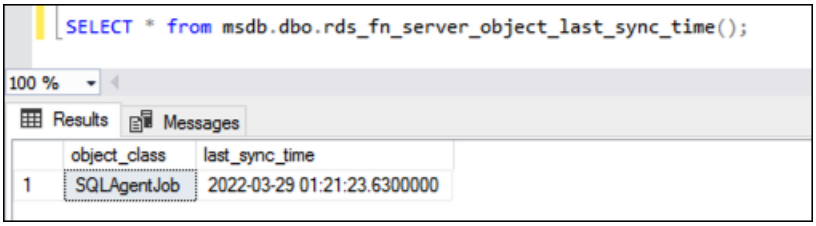


次の関数を使用して、オブジェクトが UTC 時間で最後に同期された時刻を調べることができます。

```
SELECT * from msdb.dbo.rds_fn_server_object_last_sync_time();
```

例えば、01:00 に SQL Server エージェントジョブを変更するとします。直近の同期時刻は 01:00 以降になると予想されます。これは、同期が行われたことを示します。

同期後、セカンダリノードで date_created と date_modified に返される値は一致することが期待されます。



tempdb レプリケーションも使用している場合は、@object_type パラメータで指定することで、SQL Agent ジョブと tempdb 設定の両方のレプリケーションを有効にできます。

```
EXECUTE msdb.dbo.rds_set_system_database_sync_objects @object_types =
 'SQLAgentJob,TempDbFile';
```

tempdb レプリケーションの詳細については、「マルチ AZ 配置の TempDB 設定」を参照してください。

SQL Server エージェントのロール

RDS for SQL Server は、ジョブを管理するためのさまざまなレベルのアクセス許可を持つ、以下の SQL Server エージェントのロールをサポートしています。

- SQLAgentUserRole

  アクセス許可
  - 独自のジョブ、スケジュール、オペレーターを作成および管理する
  - 独自のジョブとスケジュールのプロパティを表示する
  - 他のユーザーが作成したジョブは表示または管理できない

  このロールは、独自のジョブを作成および管理する必要があるが、他のユーザーが作成したジョブへのアクセスを必要としないユーザーに適しています。
- SQLAgentReaderRole

  アクセス許可
  - SQLAgentUserRole のすべてのアクセス許可
  - すべてのジョブとスケジュール (他のユーザーが作成したものを含む) を一覧表示する
  - すべてのジョブのプロパティを表示する
  - ジョブ履歴を確認する

  このロールは、すべてのジョブのステータスをモニタリングする必要があるが、ジョブを管理する必要がないユーザーに適しています。
- SQLAgentOperatorRole

  アクセス許可
  - SQLAgentUserRole と SQLAgentReaderRole のすべてのアクセス許可
  - ジョブを実行、停止、または開始する
  - ジョブ履歴を管理する
  - ジョブとスケジュールを有効/無効にする

- オペレーターとプロキシを表示する

このロールは、最も包括的なアクセス許可を提供するものであり、すべてのジョブを完全に制御する必要があるユーザーに適しています。

SQL Server ログインにロールを割り当てるには、次のコマンドを使用します。

```
USE msdb;
EXEC sp_addrolemember 'SQLAgentOperatorRole', 'username';
```

RDS for SQL Server での SQLAgentOperatorRole の管理

現在のジョブを表示するには、SQLAgentOperatorRole を SQL Server ログインに追加し、データベースから切断する前にそれを削除する必要があります。

SQL Server Management Studio で SQL Server エージェントツリーを視覚化するには、次の手順に従います。

SQL Server Management Studio (SSMS) で SQL Server エージェントを表示する

1. RDS マスター認証情報を使用して RDS SQL Server インスタンスにログインし、目的のユーザーに SQLAgentUserRole を付与します。

   ```
   USE msdb
   GO
   IF NOT EXISTS(SELECT name FROM sys.database_principals WHERE name = 'UserName')
   BEGIN
   CREATE USER UserName FROM LOGIN UserName
   END
   GO
   ALTER ROLE SQLAgentUserRole ADD MEMBER UserName
   GO
   GRANT ALTER ON ROLE::[SQLAgentOperatorRole] to UserName
   GO
   ```

   これらのコマンドは、msdb データベースにユーザーが存在しない場合に作成します。また、SQLAgentUserRole にユーザーを追加し、SQL Server エージェントツリーを SSMS で表示できるようにします。最後に、SQLAgentOperatorRole に対する変更アクセス許可をユーザーに付与します。これにより、ユーザーはロールに対して自身を追加または削除できます。

2. 上記のロールに自分自身を追加するには、ジョブを表示する必要があるユーザーを使用して RDS SQL Server インスタンスに接続し、次のスクリプトを実行します。

```
use msdb
go
ALTER ROLE SQLAgentOperatorRole ADD MEMBER UserName
GO
```

次に、[ジョブ] フォルダを右クリックし、[更新] を選択します。

3. このアクションを実行すると、[ジョブ] タブに [+] (プラス) ボタンが表示されます。クリックして SQL Server エージェントジョブのリストを展開します。

4. **Important**

RDS SQL Server インスタンスから切断する前に、SQLAgentOperatorRole から自分自身を削除する必要があります。

SQLAgentOperatorRole からログインを削除するには、Management Studio を切断または終了する前に次のクエリを実行します。

```
USE msdb
GO
ALTER ROLE SQLAgentOperatorRole DROP MEMBER UserName
GO
```

詳細については、「RDS SQL Server での SQLAgentOperatorRole の活用」を参照してください。

SQLAgentUser ロールへのユーザーの追加

SQL Server エージェントにログインまたはユーザーを追加できるようにするには、マスターユーザーとしてログインし、次の操作を行ってください。

1. `CREATE LOGIN` コマンドを使用して、別のサーバーレベルログインを作成します。
2. `msdb` コマンドを使用して `CREATE USER` にユーザーを作成し、このユーザーを前の手順で作成したログインにリンクします。
3. `SQLAgentUserRole` システムストアドプロシージャを使用して、`sp_addrolemember` にユーザーを追加します。

例えば、マスターユーザー名が **admin** で、ユーザー名が **theirname**、パスワードが **theirpassword** のユーザーに SQL Server エージェントへのアクセスを許可するとします。その場合は、以下の手順を使用できます。

SQLAgentUser ロールにユーザーを追加するには

1. マスターユーザーとしてログインします。
2. 以下のコマンドを実行します。

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login
 theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
--Added database user theirname in msdb to SQLAgentUserRole in msdb
EXEC sp_addrolemember [SQLAgentUserRole], [theirname];
```

SQL Server エージェントジョブの削除

`sp_delete_job` ストアドプロシージャを使用して、Amazon RDS for Microsoft SQL Server の SQL Server エージェントジョブを削除します。

SSMS を使用して SQL Server エージェントジョブを削除することはできません。これを試みると、次のようなエラーメッセージが表示されます。

```
The EXECUTE permission was denied on the object 'xp_regread', database
 'mssqlsystemresource', schema 'sys'.
```

マネージド型サービスとしての RDS では、Windows レジストリにアクセスする手順の実行が制限されています。SSMS を使用すると、RDS が承認されていないプロセス (`xp_regread`) の実行を試みます。

> **Note**
>
> RDS for SQL Server では、sysadmin ロールのメンバーのみが、別のログインが所有するジョブを更新または削除できます。詳細については、「RDS SQL Server での SQLAgentOperatorRole の活用」を参照してください。

SQL Server エージェントジョブを削除するには

- 次の T-SQL ステートメントを実行します。

```
EXEC msdb..sp_delete_job @job_name = 'job_name';
```

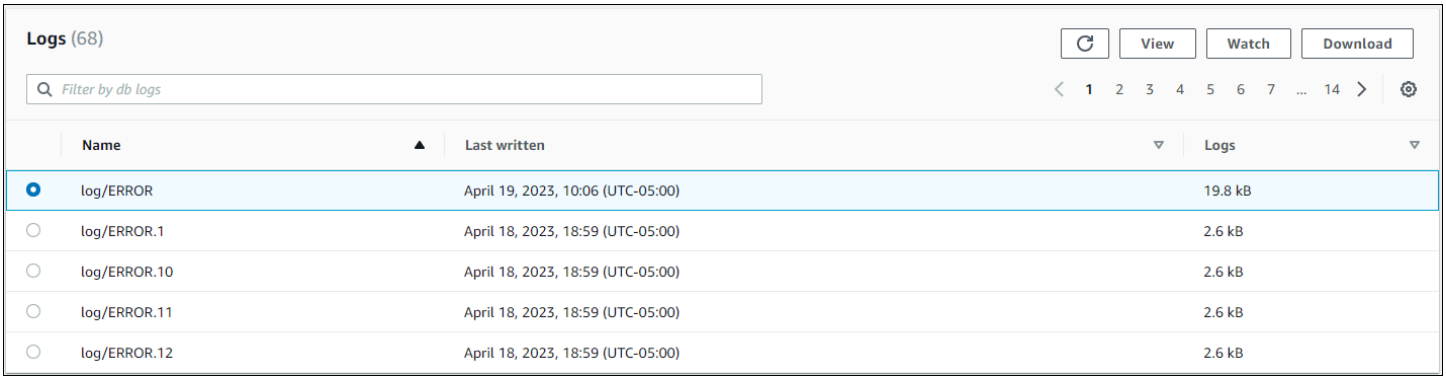
Amazon RDS for Microsoft SQL Server ログの使用方法

Amazon RDS コンソールを使用して、SQL Server エージェントのログ、Microsoft SQL Server のエラーログ、SQL Server Reporting Services (SSRS) のログを表示、監視、ダウンロードすることができます。

ログファイルの監視

Amazon RDS コンソールでログを表示すると、その時点で存在しているログの内容を表示できます。コンソールでログを監視すると、動的な状態でログが表示され、ほぼリアルタイムでログの更新を確認できます。

監視の対象としてアクティブになるのは最新のログのみです。例えば、以下に示すようなログがあるとします。



最新のログである log/ERROR のみがアクティブに更新されています。他のログを表示することもできますが、静的であり、更新されません。

ログファイルのアーカイブ

Amazon RDS コンソールには、過去 1 週間と当日のログが表示されます。ログをダウンロードしてアーカイブし、過去のリファレンスとして保持できます。ログをアーカイブする方法の 1 つとして、Amazon S3 バケットにログをロードする方法があります。Amazon S3 バケットをセットアップしてファイルをアップロードする方法については、Amazon Simple Storage Service 入門ガイドの「Amazon S3 の基礎」を参照し、[開始方法] をクリックしてください。

エラーログとエージェントログの表示

Microsoft SQL Server のエラーログおよびエージェントログを表示するには、以下のパラメータで Amazon RDS ストアドプロシージャ `rds_read_error_log` を使用します。

- **`@index`** – 取得するログのバージョン。デフォルト値は 0 で、現在のエラーログを取得します。以前のログを取得するには 1、さらにもう一つ前のログを取得するには 2、という方法で指定します。
- **`@type`** – 取得するログのタイプ。エラーログを取得するには、1 を指定します。エージェントログを取得するには、2 を指定します。

Example

以下の例では現在のエラーログをリクエストします。

```
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 1;
```

SQL Server エラーの詳細については、Microsoft ドキュメントの Database engine errors を参照してください。

Amazon RDS for SQL Server のトレースファイルとダンプファイルの使用

このセクションでは、Microsoft SQL Server を実行する Amazon RDS DB インスタンスのトレースファイルとダンプファイルの操作について説明します。

トレース SQL クエリを生成する

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint

set @maxfilesize = 5
```

```
exec @rc = sp_trace_create @TraceID output,  0, N'D:\rdsdbdata\log\rdstest',
 @maxfilesize, NULL
```

オープントレースを表示する

```
select * from ::fn_trace_getinfo(default)
```

トレースの内容を表示する

```
select * from ::fn_trace_gettable('D:\rdsdbdata\log\rdstest.trc', default)
```

トレースファイルおよびダンプファイルの保持期間を設定する

トレースファイルとダンプファイルが蓄積されて、ディスク領域を消費することがあります。デフォルトでは、Amazon RDS では、7 日を経過したトレースファイルとダンプファイルは消去されます。

現在のトレースファイルとダンプファイルの保持期間を表示するには、以下の例に示すように、rds_show_configuration の手順を使用します。

```
exec rdsadmin..rds_show_configuration;
```

トレースファイルの保持期間を変更するには、rds_set_configuration プロシージャを使用し、tracefile retention を分単位で設定します。以下の例では、トレースファイルの保持期間を 24 時間に設定しています。

```
exec rdsadmin..rds_set_configuration 'tracefile retention', 1440;
```

ダンプファイルの保持期間を変更するには、rds_set_configuration プロシージャを使用し、dumpfile retention を分単位で設定します。以下の例では、ダンプファイルの保持期間を 3 日に設定しています。

```
exec rdsadmin..rds_set_configuration 'dumpfile retention', 4320;
```

セキュリティ上の理由から、SQL Server DB インスタンスの特定のトレースファイルまたはダンプファイルを削除することはできません。未使用のトレースファイルまたはダンプファイルをすべて削除するには、ファイルの保持期間を 0 に設定します。

Amazon RDS for MySQL

Amazon RDS では、DB インスタンス向けに MySQL の複数のバージョンがサポートされています。サポートされているバージョンの詳細については、「Amazon RDS での MySQL のバージョン」を参照してください。

Amazon RDS for MySQL DB インスタンスを作成するには、Amazon RDS の管理ツールまたはインターフェイスを使用します。続いて、次の操作を行います。

- インスタンスのサイズの変更
- DB インスタンスへの接続の許可
- バックアップまたはスナップショットからの作成と復元
- マルチ AZ セカンダリの作成
- リードレプリカの作成
- DB インスタンスのアクティビティとパフォーマンスのモニタリング

DB インスタンスでデータを保存したり、またはアクセスするには、標準の MySQL のユーティリティとアプリケーションを使用します。

Amazon RDS for MySQL は、多くの業界スタンダードに準拠しています。例えば、RDS for MySQL データベースを使用して、HIPAA 準拠のアプリケーションを構築できます。RDS for MySQL データベースを使用して、AWS との事業提携契約 (BAA) に基づき、保護されるべき医療情報 (PHI) など、医療関連の情報を保存できます。また、Amazon RDS for MySQL は、Federal Risk and Authorization Management Program (FedRAMP) のセキュリティ要件を満たしています。さらに、Amazon RDS for MySQL は、AWS GovCloud (US) リージョン内において FedRAMP HIGH ベースラインで FedRAMP Joint Authorization Board (JAB) の Provisional Authority to Operate (P-ATO) も取得しています。サポートされるコンプライアンススタンダードの詳細については、AWS クラウドコンプライアンスを参照してください。

MySQL のバージョン別の機能の詳細については、MySQL ドキュメントの「MySQL の主な機能」を参照してください。

DB インスタンスを作成する前に、「Amazon RDS 環境のセットアップ」の手順を完了してください。DB インスタンスを作成すると、RDS マスターユーザーは DBA 権限を取得します。ただし、いくつかの制限があります。このアカウントは、追加のデータベースアカウントの作成などの管理タスクに使用します。

以下を作成することができます。

- DB インスタンス
- DB スナップショット
- ポイントインタイムの復元
- 自動バックアップ
- 手動バックアップ

Amazon VPC に基づいて Virtual Private Cloud (VPC) 内で MySQLを実行する DB インスタンスを使用できます。また、さまざまなオプションをオンにして、MySQL DB インスタンスに機能を追加することもできます。Amazon RDS は、可用性の高いフェイルオーバーソリューションとして MySQL のマルチ AZ 配置をサポートしています。

> **Important**
>
> マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする、特定のシステムプロシージャやテーブルへのアクセスも制限しています。データベースへのアクセスには、mysql クライアントなどの標準の SQL クライアントアプリケーションを使用します。ただし、Telnet またはセキュアシェル (SSH) を使用してホストに直接アクセスすることはできません。

トピック

- Amazon RDS での MySQL 機能のサポート
- Amazon RDS での MySQL のバージョン
- MySQL DB インスタンスへの接続
- MySQL DB インスタンス接続の保護
- Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上
- RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上
- RDS for MySQL DB エンジンのアップグレード
- MySQL DB スナップショットエンジンバージョンのアップグレード
- Amazon RDS for MySQL DB インスタンスへのデータのインポート
- Amazon RDS での MySQL のレプリケーションの使用

- RDS for MySQL のアクティブ/アクティブクラスターの設定
- レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート
- MySQL DB インスタンスのオプション
- MySQL のパラメータ
- MySQL DB インスタンスの一般的な DBA タスク
- MySQL DB インスタンスのローカルタイムゾーン
- Amazon RDS for MySQL の既知の問題と制限
- RDS for MySQL ストアドプロシージャリファレンス

Amazon RDS での MySQL 機能のサポート

RDS for MySQL は MySQL のほとんどの特徴と機能をサポートしています。一部の機能には、制限付きのサポートまたは制限された特権があります。

[What's New with Database?] (データベースの新機能) ページで新しい Amazon RDS 機能をフィルタリングできます。[製品] で [Amazon RDS] を選択します。その後、**MySQL 2022** などのキーワードを使用して検索します。

> **Note**
>
> 以下のリストは完全なものではありません。

トピック

- Amazon RDS for MySQL メジャーバージョンでの MySQL 機能のサポート
- RDS for MySQL のサポートされているストレージエンジン
- Amazon RDS の MySQL での memcached およびその他のオプションの使用
- Amazon RDS の MySQL に対する InnoDB キャッシュウォーミング
- RDS for MySQL 8.4 の包括的な言語変更
- Amazon RDS でサポートされていない MySQL の機能

Amazon RDS for MySQL メジャーバージョンでの MySQL 機能のサポート

次のセクションでは、Amazon RDS for MySQL のメジャーバージョンでの MySQL 機能のサポートに関する情報について説明しています。

トピック

- Amazon RDS は MySQL 8.4 をサポート

Amazon RDS for MySQL のサポートされているマイナーバージョンについては、「Amazon RDS でサポートされている MySQL のマイナーバージョン」を参照してください。

Amazon RDS は MySQL 8.4 をサポート

Amazon RDS は、MySQL バージョン 8.4 以降を実行する DB インスタンスで以下の新しい機能をサポートしています。

- 暗号化ライブラリ – RDS for MySQL は OpenSSL を FIPS 140-3 認定の AWS Libcrypto (AWS-LC) に置き換えました。詳細については、https://github.com/aws/aws-lc の AWS-LC GitHub リポジトリを参照してください。
- TLS の変更 – RDS for MySQL は TLS 1.2 および TLS 1.3 のみをサポートしています。詳細については、「the section called “MySQL での SSL/TLS サポート”」を参照してください。
- memcached のサポート – memcached インターフェイスは MySQL 8.4 での使用が廃止されました。詳細については、「MySQL の memcached サポート」を参照してください。
- デフォルトの認証プラグイン – デフォルトの認証プラグインは `caching_sha2_password` です。詳細については、「the section called “MySQL デフォルト認証プラグイン”」を参照してください。
- **`mysqlpump`** クライアントユーティリティ – `mysqlpump` クライアントユーティリティは MySQL 8.4 では使用不可になりました。詳細については、「AWS 規範ガイダンス」の「RDS for MySQL のロールベースの権限モデル」および「mysqldump and mysqlpump」を参照してください。
- マネージドレプリケーションストアドプロシージャ – `caching_sha2_password` で設定されたレプリケーションユーザーによるレプリケーションを、ストアドプロシージャを使用して管理する場合は、`SOURCE_SSL=1` を指定して TLS を設定する必要があります。`caching_sha2_password` は RDS for MySQL 8.4 のデフォルトの認証プラグインです。
- パラメータの動作の変更 – MySQL 8.4 で、次のパラメータが変更されました。
  - `innodb_dedicated_server` – このパラメータはデフォルトで有効になります。詳細については、「MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定」を参照してください。
  - `innodb_buffer_pool` - 現状でこのパラメータはデータベースエンジンにより計算されますが、この設定は上書きできます。詳細については、「MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定」を参照してください。
  - `innodb_redo_log_capacity` – 今後は、このパラメータで REDO ログファイルのサイズを制御します。現状でこのパラメータはデータベースエンジンにより計算されますが、この設定は上書きできます。詳細については、「MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定」を参照してください。
- 非推奨または削除されたパラメータ – RDS for MySQL では、MySQL 8.4 DB インスタンスのパラメータグループから次のパラメータが削除されました。今後は、`innodb_redo_log_capacity` パラメータで REDO ログファイルのサイズを制御します。
  - `innodb_log_file_size`
  - `innodb_log_files_in_group`
- パラメータの新しいデフォルト値 – 次のパラメータには、MySQL 8.4 DB インスタンスの新しいデフォルト値があります。

- パフォーマンスに関連するさまざまな MySQL コミュニティパラメータが変更されました。詳細については、MySQL 8.0 以降の MySQL 8.4 の新機能」を参照してください。

  本番稼働用インスタンスを移行する前に、RDS for MySQL 8.4 でアプリケーションのパフォーマンスをテストすることをお勧めします。

- `innodb_purge_threads` – InnoDB 履歴リストの長さが大きくなりすぎないように、デフォルト値は式 `LEAST({DBInstanceVCPU/2},4)` に設定されています。
- `group_replication_exit_state_action` – デフォルト値は `OFFLINE_MODE` で、MySQL コミュニティのデフォルトと一致します。詳細については、「RDS for MySQL アクティブ/アクティブクラスターに関する考慮事項とベストプラクティス」を参照してください。
- `binlog_format` – デフォルト値は `ROW` で、MySQL コミュニティのデフォルトと一致します。シングル AZ DB インスタンスまたはマルチ AZ DB インスタンスのパラメータは変更できますが、マルチ AZ DB クラスターのパラメータは変更できません。マルチ AZ DB クラスターは半同期レプリケーションを使用し、`binlog_format` を `MIXED` または `STATEMENT` に設定すると、レプリケーションは失敗します。

- 包括的な言語の変更 – RDS for MySQL 8.4 には、包括的な言語のキーワードとシステムスキーマに関連する RDS for MySQL 8.0 からの変更が含まれています。詳細については、「RDS for MySQL 8.4 の包括的な言語変更」を参照してください。

MySQL 8.4 のすべての機能と変更のリストについては、MySQL ドキュメントの「MySQL 8.0 以降の MySQL 8.4 の新機能」を参照してください。

サポートされない機能の一覧については、「the section called “サポートされていない機能”」を参照してください。

RDS for MySQL のサポートされているストレージエンジン

MySQL では、さまざまな機能を備えた複数のストレージエンジンがサポートされていますが、それらのすべてのエンジンが、回復性やデータ耐久性の高いのために最適化されているわけではありません。Amazon RDS は、MySQL DB インスタンス用の、InnoDB ストレージエンジンを完全にサポートしています。Amazon RDS でのポイントインタイムの復元とスナップショット復元機能には、回復可能なストレージエンジンが必要であり、InnoDB ストレージエンジンのみがサポートされています。詳細については、「MySQL の memcached サポート」を参照してください。

Federated ストレージエンジンは、現在、Amazon RDS for MySQL ではサポートされていません。

ユーザーが作成したスキーマの場合、MyISAM ストレージエンジンでは、信頼性の高い回復がサポートされておらず、エンジンの回復後に MySQL が再起動したとき、ポイントインタイム復元またはスナップショット復元が意図したとおりに機能せず、データが紛失または破損する場合があります。それでも Amazon RDS で MyISAM を使用する場合は、スナップショットがいくつかの条件下で役立つことがあります。

> **Note**
>
> `mysql` スキーマのシステムテーブルを MyISAM ストレージにできます。

既存の MyISAM テーブルを InnoDB テーブルに変換する場合、`ALTER TABLE` コマンド (例: `alter table TABLE_NAME engine=innodb;`) を使用できます。MyISAM と InnoDB の長所と短所は異なっているため、アプリケーションで切り替えた際の影響を切り替え前に十分に評価しておく必要があることを念頭に置いてください。

MySQL 5.1、5.5 および 5.6 は、Amazon RDS でサポートされなくなりました。ただし、既存の MySQL 5.1、5.5 および 5.6 スナップショットは復元できます。MySQL 5.1、5.5 および 5.6 スナップショットを復元すると、DB インスタンスが MySQL 5.7 に自動的にアップグレードされます。

Amazon RDS の MySQL での memcached およびその他のオプションの使用

ほとんどの Amazon RDS DB エンジンでは、DB インスタンスの追加機能を選択できる、オプショングループをサポートしています。RDS for MySQL DB インスタンスは、シンプルなキーベースのキャッシュである `memcached` オプションをサポートします。`memcached` およびその他のオプションの詳細については、「MySQL DB インスタンスのオプション」を参照してください。オプショングループの操作方法の詳細については、「オプショングループを使用する」を参照してください。

Amazon RDS の MySQL に対する InnoDB キャッシュウォーミング

InnoDB キャッシュウォームアップでは、DB インスタンスがシャットダウンされたときのバッファープールの最新の状態を保存し、DB インスタンスが起動されたときに保存された情報からバッファープールを再ロードすることによって、MySQL DB インスタンスのパフォーマンスを向上させることができます。これにより、通常のデータベースの使用からバッファープールを「ウォームアップする」必要がなくなり、既知の一般的なクエリのページを使用してバッファープールを事前にロードします。保存されたバッファープール情報を格納するファイルには、バッファープールのページ自体ではなく、バッファープール内のページのメタデータのみが格納されます。そのため、こ

のファイルは多くのストレージ領域を必要としません。ファイルサイズは、キャッシュサイズの約 0.2% となります。例えば、64 GiB のキャッシュでは、キャッシュのウォームアップファイルのサイズは 128 MiB です。InnoDB キャッシュウォームアップの詳細については、MySQL ドキュメントの「バッファプールの状態の保存とリストア」を参照してください。

RDS for MySQL DB インスタンスは、InnoDB キャッシュウォーミングをサポートしています。InnoDB キャッシュウォームアップを有効にするには、DB インスタンスのパラメータグループで `innodb_buffer_pool_dump_at_shutdown` および `innodb_buffer_pool_load_at_startup` パラメータを 1 に設定します。パラメータグループのこれらのパラメータ値を変更すると、パラメータグループを使用するすべての MySQL DB インスタンスに影響します。特定の MySQL DB インスタンスの InnoDB キャッシュウォームアップを有効にするには、それらのインスタンスの新しいパラメータグループを作成することが必要になる場合があります。パラメータグループについては、「Amazon RDS のパラメータグループ」を参照してください。

InnoDB キャッシュウォームアップは、主に、スタンダードストレージを使用している DB インスタンスにパフォーマンス上のメリットをもたらします。PIOPS ストレージを使用する場合、一般的に大きなパフォーマンス上のメリットは見られません。

Important

フェイルオーバー時など、MySQL DB インスタンスが正常にシャットダウンしなかった場合、バッファープールの状態はディスクに保存されません。この場合、DB インスタンスが再開されるときに、MySQL は利用可能なバッファープールファイルをロードします。特に害はありませんが、復元されたバッファープールは、再開前のバッファープールの最新の状態を反映していない可能性があります。起動時に InnoDB キャッシュをウォームアップするために、バッファープールの最新の状態が利用できるようにするには、定期的に「オンデマンド」でバッファープールをダンプすることをお勧めします。
自動で定期的にバッファープールをダンプするイベントを作成できます。例えば、次のステートメントは、1 時間ごとにバッファープールをダンプする `periodic_buffer_pool_dump` という名前のイベントを作成します。

```
CREATE EVENT periodic_buffer_pool_dump
ON SCHEDULE EVERY 1 HOUR
DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

MySQL のイベントの詳細については、MySQL ドキュメントの「イベントの構文」を参照してください。

オンデマンドでのバッファープールのダンプとロード

InnoDB キャッシュを「オンデマンド」で保存およびロードできます。

- バッファプールの現在の状態をディスクにダンプするには、mysql.rds_innodb_buffer_pool_dump_now ストアドプロシージャを呼び出します。
- バッファプールの保存された状態をディスクからロードするには、mysql.rds_innodb_buffer_pool_load_now ストアドプロシージャを呼び出します。
- 進行中のロードオペレーションをキャンセルするには、mysql.rds_innodb_buffer_pool_load_abort ストアドプロシージャを呼び出します。

RDS for MySQL 8.4 の包括的な言語変更

RDS for MySQL 8.4 には、包括的な言語のキーワードやシステムスキーマに関連する MySQL 8.4 コミュニティエディションからの変更が含まれています。たとえば、`SHOW REPLICA STATUS` コマンドは `SHOW SLAVE STATUS` に置き換えられています。

トピック

- 設定パラメータ名の変更
- ストアドプロシージャ名の変更

設定パラメータ名の変更

RDS for MySQL 8.4 では、次の設定パラメータに新しい名前が付けられました。

互換性のために、`mysql` クライアントでどちらの名前を使用しても、パラメータ名をチェックできます。カスタム MySQL 8.4 パラメータグループ内の値を変更する場合、新しい名前のみ使用できます。詳細については、「デフォルトおよびカスタムパラメータグループ」を参照してください。

削除される名前	新しい名前または優先される名	
`init_slave`	`init_replica`	
`log_slave_updates`	`log_replica_updates`	

削除される名前	新しい名前または優先される名	
log_slow_slave_statements	log_slow_replica_statements	
rpl_stop_slave_timeout	rpl_stop_replica_timeout	
skip_slave_start	skip_replica_start	
slave_checkpoint_group	replica_checkpoint_group	
slave_checkpoint_period	replica_checkpoint_period	
slave_compressed_protocol	replica_compressed_protocol	
slave_exec_mode	replica_exec_mode	
slave_load_tmpdir	replica_load_tmpdir	
slave_max_allowed_packet	replica_max_allowed_packet	
slave_net_timeout	replica_net_timeout	
slave_parallel_type	replica_parallel_type	
slave_parallel_workers	replica_parallel_workers	
slave_pending_jobs_size_max	replica_pending_jobs_size_max	
slave_preserve_commit_order	replica_preserve_commit_order	

削除される名前	新しい名前または優先される名
slave_skip_errors	replica_skip_errors
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_transaction_retries	replica_transaction_retries
slave_type_conversions	replica_type_conversions
sql_slave_skip_counter	sql_replica_skip_counter

> **Note**
>
> Amazon RDS は NDB クラスターをサポートしていないため、replica_allow_batching パラメータは使用できません。

ストアドプロシージャ名の変更

RDS for MySQL 8.4 では、次のストアドプロシージャは名前が変更されています。

RDS for MySQL 8.4 の初期のリリースでは、互換性のためにどちらの名前でも使用できます。以前のプロシージャ名は、将来のリリースで削除される予定です。詳細については、「バイナリログレプリケーションの設定、開始、停止」を参照してください。

削除される名前	新しい名前または優先される名
mysql.rds_next_master_log	mysql.rds_next_source_log

削除される名前	新しい名前または優先される名	
mysql.rds_reset_external_master	mysql.rds_reset_external_source	
mysql.rds_set_external_master	mysql.rds_set_external_source	
mysql.rds_set_external_master_with_auto_position	mysql.rds_set_external_source_with_auto_position	
mysql.rds_set_external_master_with_delay	mysql.rds_set_external_source_with_delay	
mysql.rds_set_master_auto_position	mysql.rds_set_source_auto_position	

Amazon RDS でサポートされていない MySQL の機能

現在、Amazon RDS では MySQL の以下の機能はサポートされていません。

- 認証用プラグイン
- システムログへのエラーログ記録
- InnoDB テーブル領域の暗号化
- NDB クラスター
- パスワード強度用プラグイン
- 永続的システム可変
- リライタクエリ書き換えプラグイン
- マルチ AZ DB クラスターを除く半同期レプリケーション
- トランスポータブルテーブルスペース
- X プラグイン

マネージドサービスエクスペリエンスを提供するために、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする特定のシステムプロシージャやテーブルへのアクセスを制限しています。Amazon RDS は、任意のスタンダード SQL クライアントアプリケーションによる、DB インスタンス上のデータベースへのアクセスがサポートされています。Amazon RDS では、Telnet、Secure Shell (SSH)、または Windows のリモートデスクトップ接続を使用しての、DB インスタンスに対するダイレクトホストアクセスは行えません。DB インスタンスを作成するユーザーには、DB インスタンスのすべてのデータベースに対する db_owner が割り当てられ、すべてのデータベースレベルのアクセス許可 (バックアップ用を除く) が付与されます。バックアップは、Amazon RDS により自動的に実行されます。

Amazon RDS での MySQL のバージョン

MySQL のバージョン番号は、バージョン = X.Y.Z の形式で記述されています。Amazon RDS の用法では、X.Y はメジャーバージョン番号を、Z はマイナーバージョン番号を示します。Amazon RDS 実装では、メジャーバージョン番号が変更された場合 (—5.7 から 8.0 へなど)、そのバージョン変更はメジャーと考えます。マイナーバージョン番号のみが変更された場合 (8.0.32 から 8.0.34 へなど)、そのバージョン変更はマイナーと考えます。

トピック

- バージョン最新性タイムライン
- Amazon RDS でサポートされている MySQL のマイナーバージョン
- Amazon RDS でサポートされている MySQL のメジャーバージョン
- RDS for MySQL のバージョンの Amazon RDS 延長サポート
- データベースプレビュー環境の使用
- データベースプレビュー環境の MySQL バージョン 9.7
- データベースプレビュー環境の MySQL バージョン 9.6
- データベースプレビュー環境の MySQL バージョン 9.5
- Amazon RDS for MySQL の非推奨バージョン

バージョン最新性タイムライン

Amazon RDS for MySQL は、コミュニティデータベースエンジンのリリースを定義された頻度で追跡します。これらのバージョン最新性タイムラインは、その頻度を明確にするために公開されます。これらのタイムラインを使用して、次のことができます。

- メジャーバージョンのアップグレードを計画し、新しい RDS for MySQL メジャーバージョンが利用可能になる時期を見積もります。
- メンテナンス期間中にマイナーバージョンアップグレードをスケジュールします。

次の表に、RDS for MySQL のバージョン最新性タイムラインを示します。

リリースタイプ	タイムライン
メジャーバージョン	新しいメジャーバージョン <major>.1 の最初のマイナーバージョン (Oracle MySQL LTS メジャー) について、コミュニティリリースから 6 か月以内
マイナーバージョン	コミュニティリリースから 30 日以内

Amazon RDS でサポートされている MySQL のマイナーバージョン

Amazon RDS は、現在、以下の MySQL マイナーバージョンをサポートしています。

> **Note**
>
> Amazon RDS 延長サポートは、マイナーバージョンでは利用できません。

次の表は、Amazon RDS が現在サポートしている MySQL 8.4 のマイナーバージョンを示しています。

MySQL エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
8.4.10	2026 年 6 月 16 日	2026 年 7 月 7 日	2027 年 7 月 7 日
8.4.9	2026 年 4 月 21 日	2026 年 5 月 8 日	2027 年 5 月 8 日
8.4.8	2026 年 1 月 20 日	2026 年 2 月 3 日	2027 年 2 月 3 日
8.4.7	2025 年 10 月 21 日	2025 年 11 月 13 日	2026 年 11 月 30 日
8.4.6	2025 年 7 月 22 日	2025 年 8 月 1 日	2026 年 10 月 31 日
8.4.5	2025 年 4 月 15 日	2025 年 4 月 29 日	2026 年 10 月 31 日

次の表は、Amazon RDS が現在サポートしている MySQL 8.0 のマイナーバージョンを示しています。

> **Note**
>
> マイナーバージョンの標準サポートは、メジャーバージョンの標準サポートより前に終了します。例えば、マイナーバージョン 8.0.28 の標準サポートは 2024 年 3 月 28 日に終了し、メジャーバージョン 8.0 の標準サポートは 2026 年 7 月 31 日に終了します。RDS は、これらの日付の間に MySQL コミュニティがリリースする追加の 8.0.* マイナーバージョンをサポートします。できるだけ頻繁に、すべてのメジャーバージョンで利用可能な最新マイナーバージョンにアップグレードすることをお勧めします。

MySQL エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 標準サポート終了日
8.0.46*	2026 年 4 月 21 日	2026 年 5 月 8 日	2026 年 7 月 31 日

* Amazon RDS 延長サポートの対象となるマイナーエンジンバージョン。詳細については、「Amazon RDS 延長サポートの使用」を参照してください。

次の表は、Amazon RDS が現在サポートしている MySQL 8.0 のマイナーバージョンを示しています。

> **Note**
>
> マイナーバージョンの延長サポートは、メジャーバージョンの延長サポートより前に終了します。RDS は、これらの日付の間に追加の 8.0.xx-RDS.xxyyzz マイナーバージョンを生成およびリリースします。できるだけ頻繁に、すべてのメジャーバージョンで利用可能な最新マイナーバージョンにアップグレードすることをお勧めします。

MySQL エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 延長サポートの終了日
8.0.46-RDS.20260624*	該当しない	2026 年 7 月 15 日	2027 年 7 月 31 日

* MySQL Community ではメジャーバージョン 8.0 を廃止し、新しいマイナーバージョンはリリースされません。これは、Amazon RDS が RDS 延長サポートの対象となる MySQL 8.0 データベースの重要なセキュリティパッチとバグ修正を使用してリリースしたマイナーバージョンです。マイナーバージョンの詳細については、「the section called “RDS 延長サポートバージョン”」を参照してください。RDS 延長サポートの詳細については、「Amazon RDS の Amazon RDS 延長サポート」を参照してください。

次の表は、Amazon RDS が現在サポートしている MySQL 5.7 のマイナーバージョンを示しています。

> **Note**
>
> マイナーバージョンの延長サポートは、メジャーバージョンの延長サポートより前に終了します。例えば、マイナーバージョン 5.7.44-RDS.20240529 は 2025 年 9 月に延長サポート終了日に達し、メジャーバージョン 5.7 は 2027 年 2 月 28 日に終了日に達します。RDS は、これらの日付の間に追加の 5.7.44-RDS.xxyyzz マイナーバージョンを生成およびリリースします。できるだけ頻繁に、すべてのメジャーバージョンで利用可能な最新マイナーバージョンにアップグレードすることをお勧めします。

MySQL エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 延長サポートの終了日
5.7.44-RDS.20260624*	該当しない	2026 年 7 月 15 日	2027 年 7 月 31 日
5.7.44-RDS.20260521*	該当しない	2026 年 6 月 4 日	2027 年 6 月 30 日
5.7.44-RDS.20260212*	該当しない	2026 年 2 月 26 日	2027 年 2 月 28 日
5.7.44-RDS.20251212*	該当しない	2025 年 12 月 12 日	2026 年 12 月 30 日
5.7.44-RDS.20250818*	該当しない	2025 年 9 月 15 日	2026 年 10 月 31 日

MySQL エンジンバージョン	コミュニティリリース日	RDS リリース日	RDS 延長サポートの終了日
5.7.44-RDS.20250508*	該当しない	2025 年 5 月 20 日	2026 年 10 月 31 日
5.7.44-RDS.20250213*	該当しない	2025 年 3 月 12 日	2026 年 10 月 31 日

* MySQL Community ではメジャーバージョン 5.7 を廃止し、新しいマイナーバージョンはリリースされません。これは、Amazon RDS が RDS 延長サポートの対象となる MySQL 5.7 データベースの重要なセキュリティパッチとバグ修正を使用してリリースしたマイナーバージョンです。マイナーバージョンの詳細については、「the section called “RDS 延長サポートバージョン”」を参照してください。RDS 延長サポートの詳細については、「Amazon RDS の Amazon RDS 延長サポート」を参照してください。

新しい DB インスタンスを作成するときは、現在サポートされているいずれかの MySQL バージョンを指定できます。メジャーバージョン (MySQL 5.7 など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいずれかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合は、Amazon RDS では、お客様が指定したメジャーバージョンの最新リリースにデフォルトで設定されます。サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示するには、AWS CLI の `describe-db-engine-versions` コマンドを実行します。

例えば RDS for MySQL のサポートされているエンジンバージョンを一覧表示するには、次の CLI コマンドを実行します。

```
aws rds describe-db-engine-versions --engine mysql --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

デフォルトの MySQL バージョンは、AWS リージョン によって異なる場合があります。特定のマイナーバージョンで DB インスタンスを作成するには、DB インスタンスの作成中にマイナーバージョンを指定します。次の AWS リージョン コマンドを実行して、AWS CLI のデフォルトのマイナーバージョンを確認できます。

```
aws rds describe-db-engine-versions --default-only --engine mysql
 --engine-version major_engine_version --region region --query "*[].
{Engine:Engine,EngineVersion:EngineVersion}" --output text
```

major-engine-version をメジャーエンジンバージョンに置き換え、*#####*を AWS リージョン に置き換えます。例えば、次の AWS CLI コマンドは、5.7 メジャーバージョンの デフォルト MySQL マイナーエンジンバージョンと、米国西部 (オレゴン) AWS リージョン (us-west-2) を返します。

```
aws rds describe-db-engine-versions --default-only --engine mysql --engine-version 5.7
 --region us-west-2 --query "*[].{Engine:Engine,EngineVersion:EngineVersion}" --output
 text
```

Amazon RDS を使用して、Amazon RDS でサポートされている新しいメジャーバージョンに MySQL インスタンスをアップグレードするタイミングを制御します。特定の MySQL バージョンとの互換性を維持したり、本番稼動用環境にデプロイする新しいバージョンを事前にアプリケーションでテストしたり、自分のスケジュールに最適なタイミングでメジャーバージョンのアップグレードを実行したりできます。

マイナーバージョンの自動アップグレードを有効にすると、DB インスタンスは Amazon RDS でサポートされる新しい MySQL マイナーバージョンに自動的にアップグレードされます。このパッチ適用は、予定されたメンテナンスウィンドウ内で行われます。DB インスタンスを変更して、マイナーバージョンの自動アップグレードを有効または無効にすることができます。

スケジュールされた自動アップグレードを解除している場合は、サポートされているマイナーバージョンリリースに手動でアップグレードできます。その手順はメジャーバージョンの更新の場合と同じです。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

Amazon RDS では現在、MySQL データベースエンジンで以下のメジャーバージョンのアップグレードがサポートされています。

- MySQL 5.7 から MySQL 8.0 へ
- MySQL 8.0 から MySQL 8.4 へ

メジャーバージョンアップグレードは、何らかの互換性のリスクを伴うため、自動的には行われません。DB インスタンスを変更するリクエストを行う必要があります。本稼働インスタンスのアップグレード前に、どのアップグレードも完全にテストする必要があります。MySQL DB インスタンスの

アップグレードの詳細については、「RDS for MySQL DB エンジンのアップグレード」を参照してください。

DB インスタンスを新しいバージョンに対してテストできます。そのためには、既存の DB インスタンスの DB スナップショットを作成し、DB スナップショットから復元して新しい DB インスタンスを作成した後、その新しい DB インスタンスのバージョンアップグレードをスタートします。その後で、オリジナルの DB インスタンスをアップグレードするかどうかを決める前に、コピーした DB インスタンスで安全性を確かめることができます。

Amazon RDS での MySQL マイナーバージョン

MySQL コミュニティがマイナーバージョンに加えた変更については、Oracle ウェブサイトの「Critical Patch Updates, Security Alerts and Bulletins」を参照してください。Critical Patch Update で、Oracle がマイナーバージョンをリリースした月を選択します。次に、影響を受ける製品とバージョンで MySQL マイナーバージョンを選択します。

マイナーバージョン

- MySQL バージョン 8.4.10
- MySQL バージョン 8.4.9
- MySQL バージョン 8.4.8
- MySQL バージョン 8.4.7
- MySQL バージョン 8.4.6
- MySQL バージョン 8.4.5
- MySQL バージョン 8.4.4
- MySQL バージョン 8.0.46
- MySQL バージョン 8.0.45
- MySQL バージョン 8.0.44
- MySQL バージョン 8.0.43
- MySQL バージョン 8.0.42
- MySQL バージョン 8.0.41
- MySQL バージョン 8.0.40
- MySQL バージョン 8.0.39
- MySQL バージョン 8.0.37

MySQL バージョン 8.4.10

Amazon RDS で MySQL バージョン 8.4.10 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MySQL バージョン 8.4.9

Amazon RDS で MySQL バージョン 8.4.9 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2026b` に基づいてタイムゾーン情報を更新しました。

- InnoDB DYNAMIC 行形式を使用するテーブルで「行サイズが大きすぎる」エラーが発生する可能性があった問題を修正しました。
- 範囲比較と LIMIT を含む ORDER BY DESC を使用するクエリで、ソート順序が正しくない可能性がある問題を修正しました。

MySQL バージョン 8.4.8

Amazon RDS で MySQL バージョン 8.4.8 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025c` に基づいてタイムゾーン情報を更新しました。

- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MySQL バージョン 8.4.7

Amazon RDS で MySQL バージョン 8.4.7 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MySQL バージョン 8.4.6

Amazon RDS で MySQL バージョン 8.4.6 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MySQL バージョン 8.4.5

Amazon RDS で MySQL バージョン 8.4.5 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025b` に基づいてタイムゾーン情報を更新しました。

MySQL バージョン 8.4.4

Amazon RDS で MySQL バージョン 8.4.4 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `tzdata2025a` に基づいてタイムゾーン情報を更新しました。
- Amazon RDS ストアドプロシージャ `mysql.rds_set_configuration` および `mysql.rds_kill` の実行中に照合エラーが発生するバグを修正しました。

MySQL バージョン 8.0.46

Amazon RDS で MySQL バージョン 8.0.46 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- InnoDB DYNAMIC 行形式を使用するテーブルで「行サイズが大きすぎる」エラーが発生する可能性があった問題を修正しました。
- 範囲比較と LIMIT を含む ORDER BY DESC を使用するクエリで、ソート順序が正しくない可能性がある問題を修正しました。

MySQL バージョン 8.0.45

Amazon RDS で MySQL バージョン 8.0.45 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025c に基づいてタイムゾーン情報を更新しました。

- 一部の SQL ステートメントが監査ログに記録されない問題を修正しました。

MySQL バージョン 8.0.44

Amazon RDS で MySQL バージョン 8.0.44 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MySQL バージョン 8.0.43

Amazon RDS で MySQL バージョン 8.0.43 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

MySQL バージョン 8.0.42

Amazon RDS で MySQL バージョン 8.0.42 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025b に基づいてタイムゾーン情報を更新しました。

MySQL バージョン 8.0.41

Amazon RDS で MySQL バージョン 8.0.41 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- tzdata2025a に基づいてタイムゾーン情報を更新しました。
- Amazon RDS ストアドプロシージャ mysql.rds_set_configuration および mysql.rds_kill の実行中に照合エラーが発生するバグを修正しました。

MySQL バージョン 8.0.40

Amazon RDS で MySQL バージョン 8.0.40 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- データベースのアップグレード中に文字セットの不一致エラーが発生するバグを修正しました。

MySQL バージョン 8.0.39

Amazon RDS で MySQL バージョン 8.0.39 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- `sql_log_off` が `SESSION_VARIABLES_ADMIN` 権限で正しく動作しないバグを修正しました。
- マスターユーザーが他のデータベースユーザーに `SESSION_VARIABLE_ADMIN` 権限を付与できないバグを修正しました。
- RDS が提供するストアドプロシージャの実行中に照合が不正に混在するバグを修正しました。

MySQL バージョン 8.0.37

Amazon RDS で MySQL バージョン 8.0.37 が使用可能になりました。このリリースには、MySQL コミュニティと Amazon RDS によって追加された修正と改善が含まれています。

新しい特徴と拡張機能

- インスタント DDL ステートメントの後に UPDATE を実行するとアサーションの失敗につながるバグを修正しました。

Amazon RDS でサポートされている MySQL のメジャーバージョン

RDS for MySQL メジャーバージョンは、少なくとも対応するコミュニティバージョンのコミュニティが終了するまでの標準サポートの対象となります。RDS の標準サポート終了日を過ぎてもメジャーバージョンは有料で引き続き使用できます。詳細については、「Amazon RDS の Amazon RDS 延長サポート」および「Amazon RDS for MySQL 価格設定」を参照してください。

次の日付を参考にすると、テストおよびアップグレードのサイクルを計画することができます。

> **Note**
>
> AWS CLI または RDS API を使用して、メジャーエンジンバージョンのサポート日に関する情報を表示することもできます。詳細については、「Amazon RDS 延長サポートでのエンジンバージョンのサポート日を表示」を参照してください。

MySQL メジャーバージョン	コミュニティリリース日	RDS リリース日	コミュニティサポート終了日	RDS 標準サポート終了日	RDS 延長サポート開始 1 年目の価格設定日	RDS 延長サポート開始 3 年目の価格設定日	RDS 延長サポートの終了日
MySQL 8.4	2024 年 4 月 30 日	2024 年 11 月 21 日	2029 年 4 月 30 日	2029 年 7 月 31 日	2029 年 8 月 1 日	2031 年 8 月 1 日	2032 年 7 月 31 日
MySQL 8.0*	2018年4月19日	2018 年 10 月 23 日	2026 年 4 月 30 日	2026 年 7 月 31 日	2026 年 8 月 1 日	2028 年 8 月 1 日	2029 年 7 月 31 日
MySQL 5.7*	2015 年 10 月 21 日	2016年2月22日	2023 年 10 月 31 日	2024 年 2 月 29 日	2024 年 3 月 1 日	2026 年 3 月 1 日	2029 年 6 月 30 日

* MySQL 5.7 および 8.0 は RDS 延長サポートでのみ利用可能になりました。詳細については、「Amazon RDS の Amazon RDS 延長サポート」を参照してください。

RDS for MySQL のバージョンの Amazon RDS 延長サポート

以下に、RDS for MySQL のバージョンの RDS 延長サポートのすべてのリリースを一覧表示します。

リリース

- RDS for MySQL バージョン 8.0.46-RDS.20260624 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20260624 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20260521 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20260212 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20251212 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20250818 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20250508 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20250213 の RDS 延長サポート

- RDS for MySQL バージョン 5.7.44-RDS.20250103 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20240808 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20240529 の RDS 延長サポート
- RDS for MySQL バージョン 5.7.44-RDS.20240408 の RDS 延長サポート

RDS for MySQL バージョン 8.0.46-RDS.20260624 の RDS 延長サポート

RDS for MySQL バージョン 8.0.46-RDS.20260624 の RDS 延長サポートが利用できます。

修正 CVE:

- CVE-2026-46863

RDS for MySQL バージョン 5.7.44-RDS.20260624 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20260624 の RDS 延長サポートが利用できます。

修正 CVE:

- CVE-2026-46863

RDS for MySQL バージョン 5.7.44-RDS.20260521 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20260521 の RDS 延長サポートが利用できます。

修正されたバグ:

- シャットダウン中のメモリリークを修正しました。

修正 CVE:

- CVE-2026-34270
- CVE-2026-34271
- CVE-2026-34272
- CVE-2026-22004
- CVE-2026-34304
- CVE-2026-35236

- CVE-2026-35237
- CVE-2026-35238
- CVE-2026-21998
- CVE-2026-22005
- CVE-2026-22002
- CVE-2026-34267
- CVE-2026-34278
- CVE-2026-35240

RDS for MySQL バージョン 5.7.44-RDS.20260212 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20260212 の RDS 延長サポートが利用できます。

修正されたバグ:

- バグ 22295186 の修正をテストするために使用されるテスト証明書を更新します。
- BLOB 列のプレフィックスインデックスを含むメモリリークを修正しました。

修正 CVE:

- CVE-2026-21936
- CVE-2026-21968
- CVE-2026-21941
- CVE-2026-21948

RDS for MySQL バージョン 5.7.44-RDS.20251212 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20251212 の RDS 延長サポートが利用できます。

修正されたバグ:

- バッファプールのサイズが上限を超えた場合にサーバー起動で発生する問題を修正しました。
- サーバーが異常終了する原因となった `INFORMATION_SCHEMA.INNODB_LOCKS` からの読み取りを修正しました。
- MySQL テストラン (MTR) での JUnit レポートのサポートに関する問題を修正しました。

- `-DWITH_INNODB_MEMCACHED=ON` オプションを使用してビルドする際のコンパイルの問題を修正しました。
- バグ 25182306 の MySQL テストラン (MTR) の実行に関する問題を修正しました。

修正 CVE:

- CVE-2025-53054
- CVE-2025-53044
- CVE-2025-53045
- CVE-2025-53062
- CVE-2025-53040
- CVE-2025-53042
- CVE-2025-53067

RDS for MySQL バージョン 5.7.44-RDS.20250818 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20250818 の RDS 延長サポートが利用できます。

修正されたバグ:

- サーバーが `autocommit=OFF` で動作したときにクエリ書き換えプラグインが失敗する問題を修正しました。
- Debian および Ubuntu ビルドがルートレスモードで実行できないアクセス許可の問題を修正しました。
- バグ 30875669 の更新の欠落を修正しました。

修正 CVE:

- CVE-2025-50082
- CVE-2025-50083
- CVE-2025-50079
- CVE-2025-50084
- CVE-2025-50087
- CVE-2025-50091

- CVE-2025-50101
- CVE-2025-50102
- CVE-2025-50098
- CVE-2025-53023
- CVE-2025-50081
- CVE-2025-50085
- CVE-2025-50077
- CVE-2025-50088
- CVE-2025-50092
- CVE-2025-50099
- CVE-2025-50096

RDS for MySQL バージョン 5.7.44-RDS.20250508 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20250508 の RDS 延長サポートが利用できます。

修正されたバグ:

- index_id が最大 uint32 より大きい場合、ロールバック後に仮想インデックスが不安定になる問題を修正しました。
- メモリの問題でテストが失敗する問題を修正しました。
- <NAME>Execute</NAME> に対して <COMMAND_CLASS> が空になる問題を修正しました。
- GCC 14 で MySQL をコンパイルする問題を修正しました [noclose 5.7]。

修正 CVE:

- CVE-2025-30682
- CVE-2025-30687
- CVE-2025-30688
- CVE-2025-21581
- CVE-2025-21585
- CVE-2025-30689
- CVE-2025-30722

- CVE-2025-21577
- CVE-2025-30693
- CVE-2025-30695
- CVE-2025-30703

RDS for MySQL バージョン 5.7.44-RDS.20250213 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20250213 の RDS 延長サポートが利用できます。

修正されたバグ:

- InnoDB のアサーション失敗 `result != FTS_INVALID` を修正しました。
- `ALTER TABLE` オペレーションで `INPLACE` アルゴリズムを使用して InnoDB テーブルを再構築した後、クラッシュし、空間インデックスが広範囲に破損する問題を修正しました。
- `ON DELETE CASCADE` の実行時に生成された列が `innobase_get_computed_value` でクラッシュする問題を修正しました。
- `row_MySQL_pad_col` でのアサーション失敗を修正しました。
- オンライン DDL で次のエラーが発生する問題を修正しました: `ERROR 1712 (HY000): Index PRIMARY is corrupted`。
- `Item_rollup_sum_switcher::current_arg` でのクラッシュを修正しました。
- データベースキャッシュが `DROP USER` でフラッシュされない問題を修正しました。
- `my_print_help` でのバッファオーバーランを修正しました。
- `FULLTEXT` インデックスによって `FTS_DOC_ID` が最大の符号なし 32 ビット値に制限される InnoDB の問題を修正しました。

修正 CVE:

- CVE-2025-21497
- CVE-2025-21555
- CVE-2025-21559
- CVE-2025-21490
- CVE-2025-21491
- CVE-2025-21500
- CVE-2025-21501

- CVE-2025-21540
- CVE-2025-21543
- CVE-2025-21520

RDS for MySQL バージョン 5.7.44-RDS.20250103 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20250103 の RDS 延長サポートが利用できます。

修正されたバグ:

- 同じトランザクションで FULLTEXT インデックスを削除して追加する際の FTS クリーンアップの問題を修正しました。
- MySQL クライアントのメモリ割り当てタイミングを最適化して、リークの可能性を防止しました。
- UNION 演算子を使用する場合の結果の切り捨てが 34 バイトに修正されました。
- 認可コードの ulong bitmask による範囲外のアクセスを修正しました。

修正 CVE:

- CVE-2024-21230
- CVE-2024-21201
- CVE-2024-21241
- CVE-2024-21203

RDS for MySQL バージョン 5.7.44-RDS.20240808 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20240808 の RDS 延長サポートが利用できます。

修正されたバグ:

- ディクショナリ列インデックスに関連するアサーションの失敗を修正しました。
- is_binlog_cache_empty() 関数の問題を修正しました。
- sql/item.cc ファイル内の heap-use-after-free エラーを修正しました。
- index-only 読み取りで空間インデックスを無効にすることで、複数の空間インデックスの問題を修正しました。

- LOCK_ORDER: CONNECTION_CONTROL プラグインのインストルメンテーションの問題を修正しました。
- CONNECTION_CONTROL プラグインでスレッドが停止する問題を修正しました。
- PREPARED STATEMENTS で PSI_THREAD_INFO の更新が行われない問題を修正しました。
- innodb_optimize_fulltext_only を使用した FTS インデックス単語の二重処理を修正しました。

修正 CVE:

- CVE-2024-21177

RDS for MySQL バージョン 5.7.44-RDS.20240529 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20240529 の RDS 延長サポートが利用できます。

修正されたバグ:

- fix_after_pullout を実装することによって、field.cc アサーションの失敗を修正しました。
- 特定の SQL クエリのメタデータをクライアントに返すときの null ポインタの障害を修正しました。これらのクエリには、SELECT 句に動的パラメータとサブクエリが含まれていました。
- ルーズインデックススキャン、またはインデックスの連続していない範囲のスキャンに GROUP BY を使用すると、誤った結果になる問題を修正しました。
- 永続化中の MySQL クラッシュに関する GTID 情報の損失を修正しました。
- InnoDB トランザクションが無期限にハングする可能性がある競合状態を修正しました。
- グループレプリケーションの認定情報クリーンアップの競合状態を修正しました。
- 同時ページ操作に伴うバックワードインデックススキャンの問題を修正しました。
- 同時シナリオでの整合性のない全文検索 (FTS) 状態の問題を修正しました。
- テーブル削除時の変更バッファに関するアサーションの問題を修正しました。
- すべてのプラグインタイプで deinit 関数を呼び出すための統合動作。

修正 CVE:

- CVE-2024-20963

- CVE-2024-20993
- CVE-2024-20998
- CVE-2024-21009
- CVE-2024-21054
- CVE-2024-21055
- CVE-2024-21057
- CVE-2024-21062
- CVE-2024-21008
- CVE-2024-21013
- CVE-2024-21047
- CVE-2024-21087
- CVE-2024-21096

RDS for MySQL バージョン 5.7.44-RDS.20240408 の RDS 延長サポート

RDS for MySQL バージョン 5.7.44-RDS.20240408 の RDS 延長サポートが利用できます。

このリリースには、以下の CVE のパッチが含まれています。

- CVE-2024-20963

データベースプレビュー環境の使用

2023 年 7 月、Oracle は MySQL の新しいリリースモデルを発表しました。このモデルには、イノベーションリリースと LTS リリースの 2 種類のリリースが含まれます。Amazon RDS では、MySQL イノベーションリリースを RDS プレビュー環境で利用できます。MySQL イノベーションリリースの詳細については、「MySQL イノベーションと長期Support (LTS) バージョンの紹介」を参照してください。

データベースプレビュー環境の RDS for MySQL DB インスタンスは、機能的に他の RDS for MySQL DB インスタンスに似ています。ただし、データベースプレビュー環境は本稼働に使用できません。

プレビュー環境には以下の制限があります。

- Amazon RDS は、すべての DB インスタンスを作成から 60 日後にバックアップおよびスナップショットとともに削除します。

- 汎用 SSD およびプロビジョンド IOPS SSD ストレージのみを使用できます。
- DB インスタンスに関して サポート からヘルプを受けることはできません。代わりに、AWS マネージド Q&A コミュニティ、AWS re:Post に質問を投稿できます。
- DB インスタンスのスナップショットを本稼働環境にコピーすることはできません。

プレビューでは、以下のオプションがサポートされています。

- db.m6i、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5、db.m5 DB インスタンスクラスを使って DB インスタンスを作成できます。RDS インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。
- シングル AZ 配置とマルチ AZ 配置の両方を使用できます。
- スタンダードの MySQL ダンプおよびロード機能を使用して、データベースをデータベースプレビュー環境にエクスポートしたり、データベースプレビュー環境にインポートしたりできます。

データベースプレビュー環境でサポートされない機能

以下の機能は、データベースプレビュー環境で使用できません。

- クロスリージョンスナップショットのコピー
- クロスリージョンリードレプリカ
- RDS Proxy

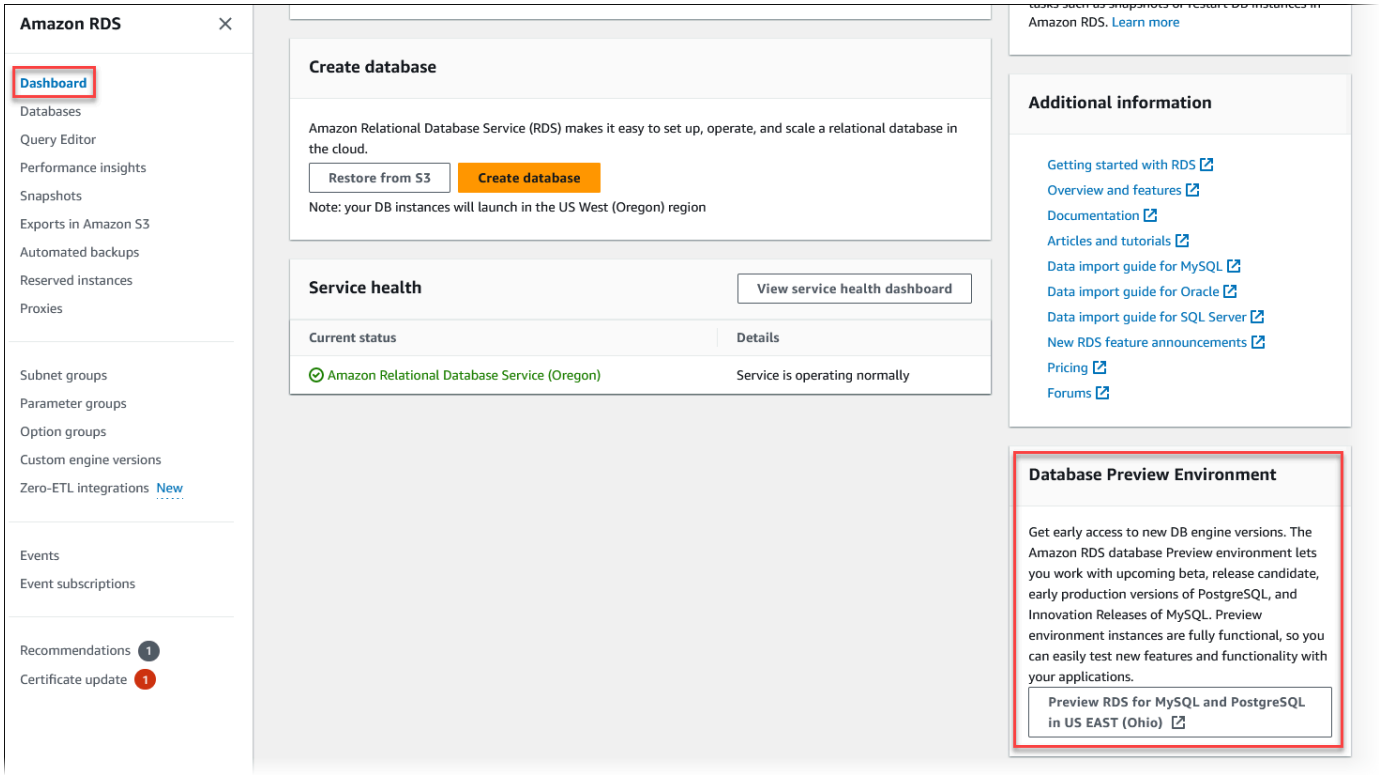
データベースプレビュー環境での新しい DB インスタンスの作成

AWS マネジメントコンソール、AWS CLI または RDS API を使用して、データベースプレビュー環境で DB インスタンスを作成できます。

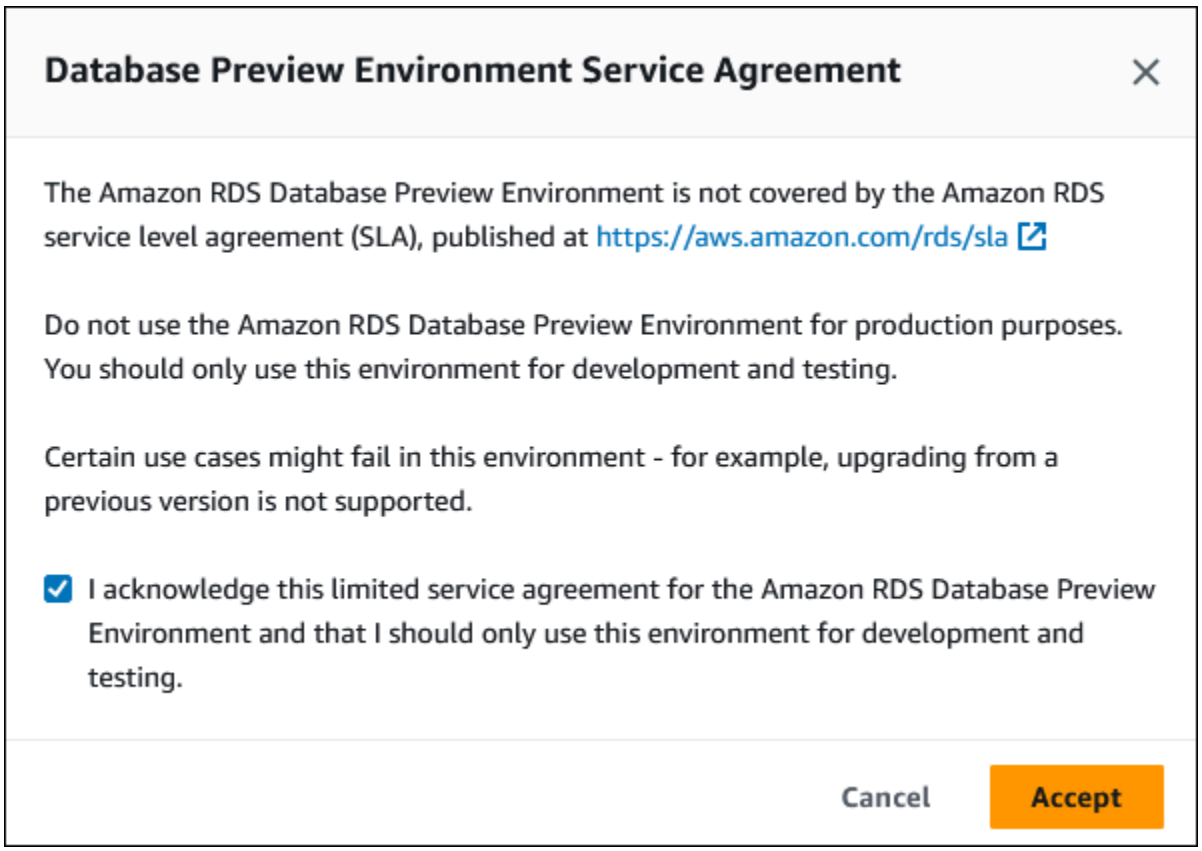
コンソール

データベースプレビュー環境で新しい DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[ダッシュボード] を選択します。
3. [ダッシュボード] ページで、次の図に示すように、[データベースプレビュー環境] セクションを見つけます。



また、[データベースプレビュー環境] に直接移動することもできます。先に進む前に、制限事項を確認して同意する必要があります。



4. RDS for MySQL DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の コンソール 手順を参照してください。

AWS CLI

AWS CLI を使用してデータベースプレビュー環境でインスタンスを作成するには、次のエンドポイントを使用します。

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MySQL DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の AWS CLI 手順を参照してください。

RDS API

RDS API を使用してデータベースプレビュー環境でインスタンスを作成するには、次のエンドポイントを使用します。

```
rds-preview.us-east-2.amazonaws.com
```

RDS for MySQL DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、「DB インスタンスの作成」の RDS API 手順を参照してください。

データベースプレビュー環境の MySQL バージョン 9.7

MySQL バージョン 9.7 が Amazon RDS データベースプレビュー環境で利用可能になりました。MySQL バージョン 9.7 には、「Changes in MySQL 9.7.0」に記載されているいくつかの改善点が含まれています。

データベースプレビュー環境の詳細については、「the section called “データベースプレビュー環境”」を参照してください。コンソールからプレビュー環境にアクセスするには、https://console.aws.amazon.com/rds-preview/ を選択します。

データベースプレビュー環境の MySQL バージョン 9.6

MySQL バージョン 9.6 が Amazon RDS データベースプレビュー環境で利用可能になりました。MySQL バージョン 9.6 には、「Changes in MySQL 9.6.0」に記載されているいくつかの改善点が含まれています。

データベースプレビュー環境の詳細については、「the section called “データベースプレビュー環境”」を参照してください。コンソールからプレビュー環境にアクセスするには、https://console.aws.amazon.com/rds-preview/ を選択します。

> **Note**
>
> `utf8` 文字セット名と `utf8_*` 照合順序名は、MySQL 9.6 のサーバー起動オプションとして受け入れられなくなりました。パラメータグループでは、すべての `character_set_*` および `collation_*` パラメータにおいて、`utf8` は `utf8mb3` に置き換えられ、`utf8_*` 照合順序は対応する `utf8mb3_*` に置き換えられました。MySQL 9.6 にアップグレードすると、パラメータグループが自動的に新しいバージョンに更新されます。ただし、以前にカスタムパラメータグループ設定を通じてこれらのパラメータのいずれかを `utf8` または `utf8_*` 照合順序に設定していた場合は、`utf8mb3/utf8mb3_*` を使用するように更新するか、完全な Unicode サポートのために `utf8mb4` への移行を検討してください。

データベースプレビュー環境の MySQL バージョン 9.5

MySQL バージョン 9.5 が Amazon RDS データベースプレビュー環境で利用可能になりました。MySQL バージョン 9.5 には、「Changes in MySQL 9.5.0」に記載されているいくつかの改善点が含まれています。

データベースプレビュー環境の詳細については、「the section called “データベースプレビュー環境”」を参照してください。コンソールからプレビュー環境にアクセスするには、https://console.aws.amazon.com/rds-preview/ を選択します。

Amazon RDS for MySQL の非推奨バージョン

Amazon RDS for MySQL バージョン 5.1、5.5 および 5.6 は非推奨です。

Amazon RDS for MySQL バージョン 9.1、9.2、9.3、および 9.4 は、データベースプレビュー環境で廃止されました。

MySQL の Amazon RDS 廃止ポリシーについては、「Amazon RDS についてのよくある質問」を参照してください。

MySQL DB インスタンスへの接続

MySQL データベースエンジンを実行している DB インスタンスに接続するには、DB インスタンスを作成する必要があります。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。Amazon RDS によって DB インスタンスがプロビジョニングされたら、スタンダードの MySQL クライアントアプリケーションまたはユーティリティを使用してインスタンスに接続できます。接続文字列では、DB インスタンスのエンドポイントの DNS アドレスをホストパラメータとして指定し、DB インスタンスのエンドポイントのポート番号をポートパラメータとして指定します。

RDS DB インスタンスを認証するには、MySQL の認証メソッドのいずれかと AWS Identity and Access Management (IAM) データベース認証を使用できます。

- MySQL の認証方法のいずれかを使用して MySQL に対して認証する方法については、MySQL ドキュメントの「認証方法」を参照してください。
- IAM データベース認証を使用して MySQL を認証する方法については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。

MySQL コマンドラインクライアントなどのツールを使用して、MySQL DB インスタンスに接続できます。MySQL コマンドラインクライアントを使用する方法については、MySQL ドキュメントの「mysql - MySQL コマンドラインクライアント」を参照してください。接続に使用できる GUI ベースのアプリケーションは、MySQL Workbench です。詳細については、「MySQL Workbench のダウンロード」ページを参照してください。MySQL のインストール (MySQL コマンドラインクライアントを含む) については、「MySQL のインストールと更新」を参照してください。

Amazon VPC の外部から DB インスタンスに接続するには、DB インスタンスがパブリックにアクセス可能であり、アクセスが DB インスタンスのセキュリティグループのインバウンドルールで許可されているなど、いくつかの要件を満たす必要があります。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

MySQL DB インスタンスへの接続で Secure Sockets Layer (SSL) または Transport Layer Security (TLS) 暗号化を使用できます。詳細については、Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート を参照してください。AWS Identity and Access Management (IAM) データベース認証を使用している場合は、必ず SSL/TLS 接続を使用してください。詳細については、MariaDB、MySQL、および PostgreSQL の IAM データベース認証 を参照してください。

また、ウェブサーバーから DB インスタンスに接続することもできます。詳細については、「チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する」を参照してください。

> **Note**
>
> MariaDB DB インスタンスとの接続方法の詳細については、「MariaDB DB インスタンスへの接続」を参照してください。

RDS for MySQL DB インスタンスを検索して接続するには、次のトピックを参照してください。

トピック

- RDS for MySQL DB インスタンスの接続情報の検索
- MySQL コマンドラインクライアントのインストール
- MySQL コマンドラインクライアントからの接続 (非暗号化)
- MySQL Workbench からの接続
- MySQL 用 AWS JDBC ドライバー、AWS Python ドライバー、AWS ODBC ドライバーを使用して RDS for MySQL に接続する
- MySQL DB インスタンスへの接続のトラブルシューティング

RDS for MySQL DB インスタンスの接続情報の検索

DB インスタンスの接続情報には、エンドポイント、ポート、およびマスターユーザーなどの有効なデータベースユーザーが含まれます。例えば、エンドポイントの値が `mydb.123456789012.us-east-1.rds.amazonaws.com` であるとします。この場合、ポート値は `3306` であり、データベースユーザーは `admin` です。この情報を考慮して、接続文字列に次の値を指定します。

- ホスト、ホスト名または DNS 名には、`mydb.123456789012.us-east-1.rds.amazonaws.com` を指定します。
- ポートで、`3306` を指定します。
- ユーザーには、`admin` を指定します。

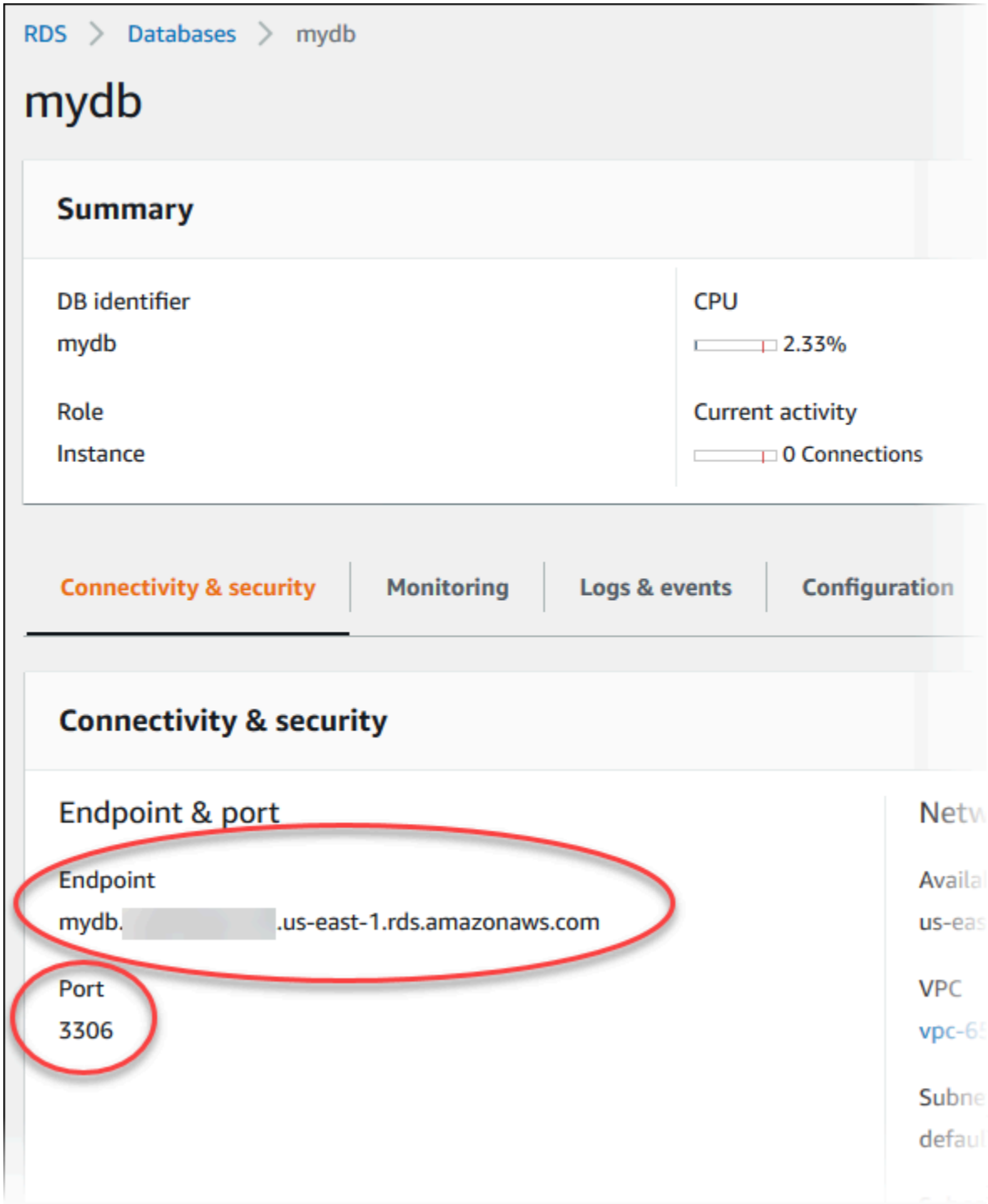
DB インスタンスに接続するには、MySQL DB エンジンの任意のクライアントを使用します。例えば、MySQL コマンドラインクライアントまたは MySQL ワークベンチを使用できます。

DB インスタンスの接続情報を検索するには、AWS マネジメントコンソール、AWS CLI describe-db-instances コマンド、または Amazon RDS API DescribeDBInstances オペレーションを使用して、詳細を一覧表示できます。

コンソール

AWS マネジメントコンソール で DB インスタンスの接続情報を探すには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択して DB インスタンスのリストを表示します。
3. MySQL DB インスタンスの名前を選択して、その詳細を表示します。
4. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



5. マスターユーザーネームを見つける必要がある場合は、[設定] タブを選択し、[マスターユーザーネーム] の値を表示します。

AWS CLI

AWS CLI を使用して MySQL DB インスタンスの接続情報を検索するには、describe-db-instances コマンドを実行します。呼び出しで、DB インスタンス ID、エンドポイント、ポート、マスターユーザー名をクエリします。

Linux、macOS、Unix の場合:

```
aws rds describe-db-instances \
  --filters "Name=engine,Values=mysql" \
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

Windows の場合:

```
aws rds describe-db-instances ^
  --filters "Name=engine,Values=mysql" ^
  --query "*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"
```

出力は次のようになります。

```
[
    [
        "mydb1",
        "mydb1.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ],
    [
        "mydb2",
        "mydb2.123456789012.us-east-1.rds.amazonaws.com",
        3306,
        "admin"
    ]
]
```

RDS API

Amazon RDS API を使用して DB インスタンスの接続情報を検索するには、DescribeDBInstances オペレーションを呼び出します。出力で、エンドポイントアドレス、エンドポイントポート、およびマスターユーザー名の値を検索します。

MySQL コマンドラインクライアントのインストール

ほとんどの Linux ディストリビューションには、Oracle MySQL クライアントではなく MariaDB クライアントが含まれています。MariaDB の MySQL コマンドラインクライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install mariadb105
```

MariaDB の MySQL コマンドラインクライアントを Amazon Linux 2 にインストールするには、次のコマンドを実行します。

```
sudo yum install mariadb
```

ほとんどの DEB ベースの Linux ディストリビューションに MySQL コマンドラインクライアントをインストールするには、次のコマンドを実行します。

```
apt-get install mariadb-client
```

MySQL コマンドラインクライアントのバージョンを確認するには、次のコマンドを実行します。

```
mysql --version
```

現在のクライアントバージョン用の MySQL ドキュメントを表示するには、次のコマンドを実行します。

```
man mysql
```

MySQL コマンドラインクライアントからの接続 (非暗号化)

> **Important**
>
> クライアントとサーバーが同じ VPC にあり、ネットワークが信頼されている場合に限り、暗号化されていない MySQL 接続を使用します。暗号化された接続の使用については、「MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MySQL DB インスタンスへの接続 (暗号化)」を参照してください。

MySQL コマンドラインクライアントを使用して DB インスタンスに接続するには、コマンドプロンプトで次のコマンドを入力します。-h パラメータは、DB インスタンスの DNS 名 (エンドポイント) に置き換えます。- P パラメータは、使用中の DB インスタンスのポートに置き換えます。-u パラメータでは、マスターユーザーなどの有効なデータベースユーザーのユーザー名に置き換えます。プロンプトが表示されたら、マスターユーザーパスワードを入力します。

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com -P 3306 -
u mymasteruser -p
```

ユーザーのパスワードを入力すると、次のような出力が表示されます。

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9738
Server version: 8.0.28 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

MySQL Workbench からの接続

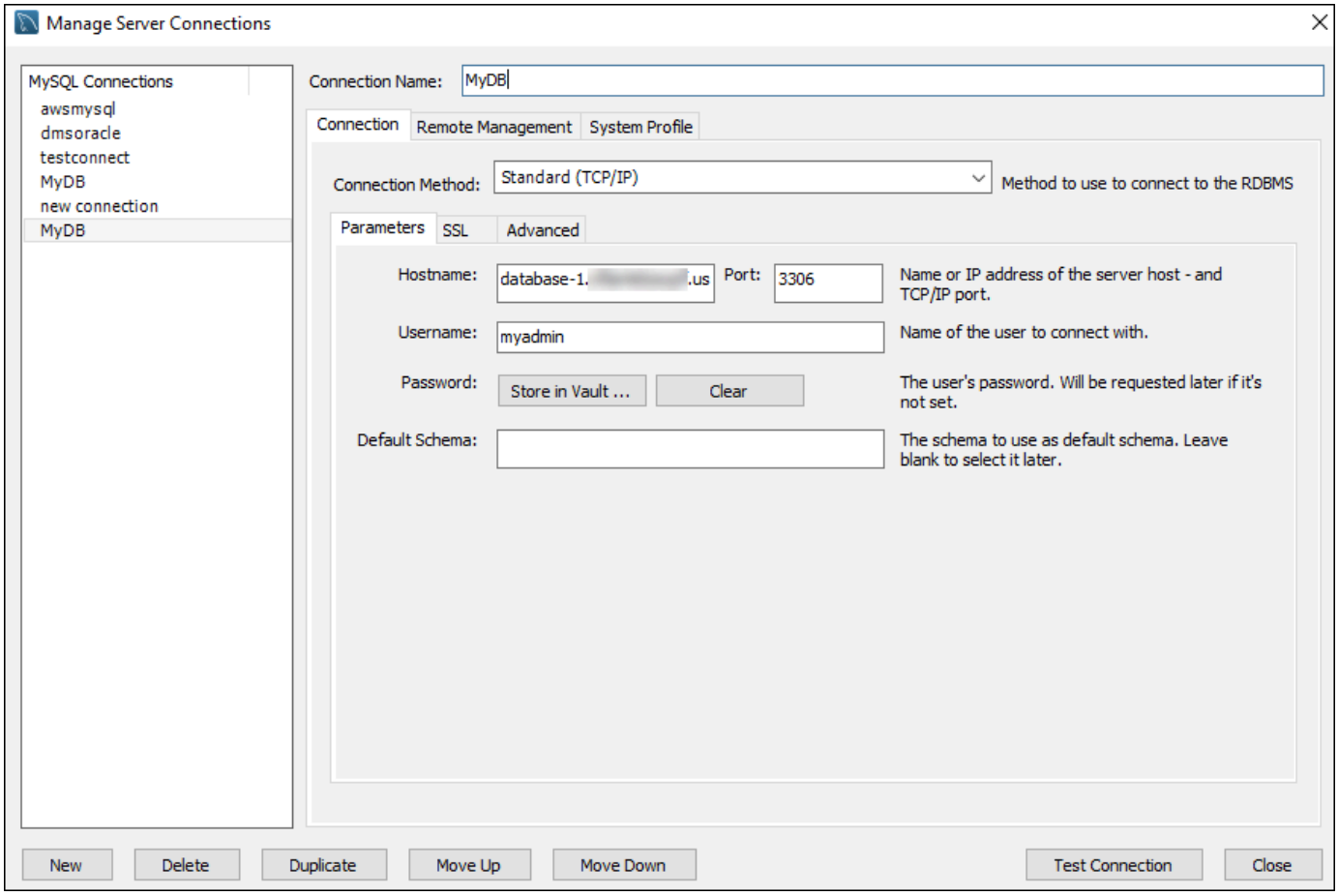
MySQL Workbench から接続するには

1. MySQL Workbench (MySQL Workbench のダウンロード) をダウンロードしてインストールします。
2. MySQL Workbench を開きます。

3. [データベース] から、[Manage Connections (接続の管理)] を選択します。
4. [Manage Server Connections (サーバー接続の管理)] ウィンドウで、 [新規] を選択します。
5. [Connect to Database (データベースに接続)] ウィンドウに、次の情報を入力します。

   - [Stored Connection] - 接続の名前 (**MyDB** など) を入力します。
   - [ホスト名] - DB インスタンスのエンドポイントを入力します。
   - [ポート] - DB インスタンスで使用するポートを入力します。
   - [ユーザーネーム] - マスターユーザーなど、有効なデータベースユーザーのユーザーネームを入力します。
   - [パスワード] - 必要に応じて、[Store in Vault (ボールトに保存)] を選択し、ユーザーのパスワードを入力して保存します。

   ウィンドウは次のようになります。



MySQL Workbench の機能を使用して、接続をカスタマイズできます。例えば、[SSL] タブを使用して SSL/TLS 接続を設定できます。MySQL Workbench の使用方法については、「MySQL

Workbench のドキュメント」を参照してください。SSL/TLS を使用した MySQL DB インスタンスへのクライアント接続の暗号化については、SSL/TLS を使用した Amazon RDS 上の MySQL DB インスタンスへのクライアント接続の暗号化 を参照してください。

6. 必要に応じて [Test Connection] を選択して、DB インスタンスへの接続が成功したことを確認します。
7. [Close] を選択します。
8. [データベース] から、[Connect to Database (データベースに接続)] を選択します。
9. [Stored Connection] から、接続を選択します。
10. [OK] を選択します。

MySQL 用 AWS JDBC ドライバー、AWS Python ドライバー、AWS ODBC ドライバーを使用して RDS for MySQL に接続する

MySQL 用 AWS JDBC ドライバー、AWS Python ドライバー、AWS ODBC ドライバーを使用して、RDS for MySQL DB インスタンスに接続します。詳細については、以下のトピックを参照してください。

トピック

- Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for MySQL への接続
- Amazon Web Services (AWS) Python ドライバーを使用した RDS for MySQL への接続
- MySQL 用 Amazon Web Services (AWS) ODBC ドライバーを使用した RDS for MySQL への接続

Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for MySQL への接続

Amazon Web Services (AWS) JDBC ドライバーは、高度な JDBC ラッパーとして設計されています。このラッパーは、既存の JDBC ドライバーの機能を補完し、拡張します。ドライバーには、コミュニティ MySQL Connector/J ドライバーおよびコミュニティ MariaDB Connector/J ドライバーとドロップイン互換性があります。

AWS JDBC ドライバーをインストールするには、AWS JDBC ドライバーの .jar ファイル (CLASSPATH アプリケーション内) を追加して、それぞれのコミュニティドライバーへの参照を保持します。対応する接続 URL プレフィックスを次のように更新します。

- `jdbc:mysql://` ~ `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` ~ `jdbc:aws-wrapper:mariadb://`

AWS JDBC ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) JDBC ドライバー GitHub リポジトリ」を参照してください。

Amazon Web Services (AWS) Python ドライバーを使用した RDS for MySQL への接続

Amazon Web Services (AWS) Python ドライバーは、高度な Python ラッパーとして設計されています。このラッパーは、オープンソースの Psycopg ドライバーの機能を補完し、拡張します。AWS Python ドライバーは Python バージョン 3.8 以降をサポートしています。`aws-advanced-python-wrapper` パッケージは、`pip` コマンドと `psycopg` オープンソースパッケージを使用してインストールできます。

AWS Python ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) Python Driver GitHub repository」を参照してください。

MySQL 用 Amazon Web Services (AWS) ODBC ドライバーを使用した RDS for MySQL への接続

MySQL 用 AWS ODBC ドライバーは、RDS for MySQL の高可用性を実現するために設計されたクライアントドライバーです。このドライバーは MySQL Connector/ODBC ドライバーと共存でき、同じワークフローと互換性があります。

MMySQL 用 AWS ODBC ドライバーおよびそのインストールと使用方法の詳細については、「Amazon Web Services (AWS) MySQL 用 ODBC ドライバー」GitHub リポジトリを参照してください。

MySQL DB インスタンスへの接続のトラブルシューティング

新しい DB インスタンスへの接続に失敗する一般的な原因には、次の 2 つがあります。

- MySQL アプリケーションまたはユーティリティが実行されているデバイスまたは Amazon EC2 インスタンスからの接続を許可しないセキュリティグループを使用して DB インスタンスが作成されました。DB インスタンスには、接続を許可する VPC セキュリティグループが必要です。詳しくは、「Amazon VPC と Amazon RDS」を参照してください。

  セキュリティグループでインバウンドのルールを追加または編集できます。[ソース] には [マイ IP] を選択します。これにより、ブラウザで検出された IP アドレスから DB インスタンスへのアクセスが許可されます。

- DB インスタンスが、デフォルトポートの 3306 を使用して作成されたが、会社のファイアウォールルールで、社内ネットワークのデバイスからそのポートへの接続がブロックされています。この問題を解決するには、別のポートでインスタンスを再起動します。

接続の問題の詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

MySQL DB インスタンス接続の保護

MySQL DB インスタンスを不正アクセスや潜在的な脅威から保護するための堅牢なセキュリティ対策を実装できます。セキュリティグループ、SSL/TLS 暗号化、IAM データベース認証は連携して、MySQL DB インスタンスの接続セキュリティの複数のレイヤーを作成します。これらのセキュリティコントロールは、コンプライアンス要件を満たし、データ侵害を防ぎ、アプリケーションとデータベース間の安全な通信チャネルを維持するのに役立ちます。転送中のデータを暗号化し、特定の IP 範囲へのアクセスを制限し、データベースパスワードではなく IAM ロールを使用してユーザー認証を管理することで、MySQL DB インスタンスを保護できます。

MySQL DB インスタンスのセキュリティは以下の 3 つのレベルで管理されます。

- AWS Identity and Access Management では、どのユーザーが DB インスタンスに対して Amazon RDS の管理アクションを実行できるかを制御します。IAM 認証情報を使用して AWS に接続している場合、IAM アカウントには、Amazon RDS の管理オペレーションを実行するためのアクセス許可を付与する IAM ポリシーが必要です。(詳しくは、「Amazon RDS での Identity and Access Management」を参照してください。)
- DB インスタンスを作成するときは、VPC セキュリティグループを使用して、どのデバイスまたは Amazon EC2 インスタンスが DB インスタンスのエンドポイントとポートへの接続を開くことができるかを制御します。これらのエンドポイントおよびポートの接続は、Secure Socket Layer (SSL) と Transport Layer Security (TLS) を使用して行います。さらに、会社のファイアウォールルールでも、社内のどのデバイスが DB インスタンスへの接続を開くことができるかを制御できます。
- MySQL DB インスタンスに対するログインとアクセス許可を認証するには、次の方法のいずれか、または組み合わせて行います。
  - MySQL のスタンドアロンインスタンスと同じ方法を使用することもできます。`CREATE USER`、`RENAME USER`、`GRANT`、`REVOKE`、`SET PASSWORD` などのコマンドは、オンプレミスデータベースでの方法と同様に、データベーススキーマテーブルを直接変更します。ただし、データベーススキーマテーブルを直接変更することはベストプラクティスではなく、RDS for MySQL バージョン 8.0.36 以降ではサポートされていません。詳細については、MySQL ドキュメントの「Access Control and Account Management」を参照してください。
  - IAM データベース認証を使用することもできます。IAM データベース認証を使用する場合は、IAM ユーザーまたは IAM ロールおよび認証トークンを使用して、DB インスタンスを認証します。認証トークン は、署名バージョン 4 の署名プロセスを使用して生成されている一意の値です。IAM データベース認証では、同一の認証情報を使用して AWS リソースおよびデータ

ベースへのアクセスを制御できます。(詳しくは、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。)

- もう 1 つのオプションは、RDS for MySQL の Kerberos 認証です。DB インスタンスは AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) を使用して Kerberos 認証を有効にします。信頼するドメインに参加している MySQL DB インスタンスに対してユーザーが認証を行う場合、認証リクエストが転送されます。転送されたリクエストは、Directory Service で作成したドメインディレクトリに移動します。(詳しくは、「Amazon RDS for MySQL での Kerberos 認証の使用」を参照してください。)

Amazon RDS DB インスタンスを作成すると、マスターユーザーには以下のデフォルト権限が付与されます。

エンジンバージョン	システム権限	データベースロール
RDS for MySQL バージョン 8.4.3 以上	`GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES ,INDEX, ALTER, SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE , REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE ROLE, DROP ROLE, APPLICATION_PASSWORD_ADMIN , FLUSH_OPTIMIZER_COSTS , FLUSH_PRIVILEGES , FLUSH_STATUS , FLUSH_TABLES , FLUSH_USER_RESOURCES , ROLE_ADMIN , SENSITIVE_VARIABLES_OBSERVER , SESSION_VARIABLES_ADMIN , SET_ANY_DEFINER , SHOW_ROUTINE , XA_RECOVER_ADMIN`	`rds_superuser_role` `rds_superuser_role` の詳細については、「RDS for MySQL のロールベースの権限モデル」を参照してください。
RDS for MySQL バージョン	`SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES , INDEX, ALTER, SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION`	`rds_superuser_role` `rds_superuser_role` の詳細については、「RDS for MySQL のロールベースの権

エンジンバージョン	システム権限	データベースロール
8.0.36 以上	SLAVE, REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE ROLE, DROP ROLE, APPLICATION_PASSWORD_ADMIN , ROLE_ADMIN , SET_USER_ID , XA_RECOVER_ADMIN	限モデル」を参照してください。
RDS for MySQL バージョン 8.0.36 未満	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES , INDEX, ALTER, SHOW DATABASES , CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION CLIENT , CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, REPLICATION SLAVE	なし

> **Note**
>
> DB インスタンスのマスターユーザーを削除できますが、お勧めしません。マスターユーザーを再作成するには、ModifyDBInstance RDS API オペレーションを使用するか、または modify-db-instance AWS CLI コマンドを実行して、新しいマスターユーザーのパスワードを該当するパラメータで指定します。インスタンスに既存のマスターユーザーがない場合、指定したパスワードを使用してマスターユーザーが作成されます。

各 DB インスタンスに管理サービスを提供するために、DB インスタンスの作成時に `rdsadmin` ユーザーが作成されます。`rdsadmin` アカウントの権限をドロップ、名前変更、パスワード変更、または変更しようとするとエラーになります。

DB インスタンスの管理を可能にするために、スタンダード的なコマンド `kill` と `kill_query` の使用は制限されています。代わりに、Amazon RDS のコマンド `rds_kill` と `rds_kill_query` が用意されており、DB インスタンスでのエンドユーザーセッションやクエリを終了させるために使用できます。

RDS for MySQL に対するパスワードの検証

MySQL では、セキュリティを高めるための validate_password プラグインが提供されています。このプラグインでは、MySQL DB インスタンスの DB パラメータグループでパラメータを使用してパスワードポリシーを適用します。プラグインは、MySQL バージョン 5.7、8.0、および 8.4 を実行している DB インスタンスでサポートされています。validate_password プラグインの詳細については、MySQL ドキュメントの「パスワード検証プラグイン」を参照してください。

MySQL DB インスタンスの **validate_password** プラグインを有効にするには

1. MySQL DB インスタンスに接続し、次のコマンドを実行します。

```
INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

2. DB インスタンスで使用される DB パラメータグループでプラグインのパラメータを設定します。

   パラメータに関する詳細については、MySQL ドキュメントの「パスワード検証プラグインオプションおよび変数」を参照してください。

   DB インスタンスのパラメータの変更の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。
3. DB インスタンスを再起動します。

validate_password プラグインを有効にしたら、新しい検証ポリシーに準拠するように、既存のパスワードをリセットします。

MySQL DB インスタンスは、Amazon RDS のパスワード検証を処理します。パスワードを変更するには、まず AWS マネジメントコンソール、modify-db-instance CLI コマンド、または ModifyDBInstance API オペレーションを使用してパスワード更新リクエストを送信します。パスワードがポリシーを満たしていなくても、RDS は最初にリクエストを受け入れます。その後、RDS はリクエストを非同期的に処理します。MySQL DB インスタンスのパスワードは、パスワードが定義されたポリシーを満たしている場合にのみ更新されます。パスワードがこれらのポリシーを満たさない場合、RDS は既存のパスワードを保持し、エラーイベントをログに記録します。

```
    Unable to reset your password. Error information: Password failed to meet
validation rules.
```

Amazon RDS イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。

SSL/TLS を使用した Amazon RDS 上の MySQL DB インスタンスへのクライアント接続の暗号化

Secure Sockets Layer (SSL) は、クライアントとサーバー間のネットワーク接続を安全に保つための業界標準のプロトコルです。SSL バージョン 3.0 以降では、名前が Transport Layer Security (TLS) に変更されています。Amazon RDS は、MySQL DB インスタンス向けに SSL/TLS 暗号化をサポートしています。SSL/TLS を使用して、アプリケーションクライアントと MySQL DB インスタンス間の接続を暗号化できます。SSL/TLS サポートは、MySQL のすべての AWS リージョン で提供されています。

Amazon RDS を使用すると、SSL/TLS を使用して MySQL DB インスタンスへのクライアント接続を暗号化し、MySQL DB インスタンスへのすべての接続に SSL/TLS を要求し、MySQL コマンドラインクライアントから SSL/TLS (暗号化) を使用して接続することで、転送中のデータを保護できます。以下のセクションでは、Amazon RDS 上の MySQL DB インスタンスの SSL/TLS 暗号化の設定と使用に関するガイダンスを提供します。

トピック

- Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート
- Amazon RDS の MySQL DB インスタンスへの特定のユーザーアカウントに SSL/TLS を要求する
- Amazon RDS 上の MySQL DB インスタンスへのすべての接続に SSL/TLS を要求する
- MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MySQL DB インスタンスへの接続 (暗号化)

Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート

Amazon RDS によって、Amazon RDS によるインスタンスのプロビジョニング時、SSL/TLS 証明書が作成され、DB インスタンスにインストールされます。これらの証明書は認証局によって署名されます。SSL/TLS 証明書には、なりすまし攻撃から保護するために、SSL/TLS 証明書の共通名 (CN) として DB インスタンスのエンドポイントが含まれています。

Amazon RDS によって作成された SSL/TLS 証明書は信頼されたルートエンティティであり、ほとんどの場合は使用できますが、アプリケーションが証明書チェーンを受け入れていない場合は使用できない可能性があります。アプリケーションが証明書チェーンを受け入れていない場合は、中間証明書を使用して AWS リージョンに接続してみてください。例えば、SSL/TLS を使用して AWS GovCloud (US) リージョンに接続するには、中間証明書を使用する必要があります。

証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。MySQL での SSL/TLS の使用の詳細については、「新しい SSL/TLS 証明書を使用して MySQL DB インスタンスに接続するようにアプリケーションを更新する」を参照してください。

MySQL バージョン 8.0 以前では、Amazon RDS for MySQL は OpenSSL を使用して接続のセキュリティを保護しています。MySQL バージョン 8.4 以降では、Amazon RDS for MySQL は AWS-LC を使用します。TLS のサポートは MySQL のバージョンによって異なります。次の表は、MySQL バージョンの TLS サポートを示しています。

MySQL のバージョン	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
MySQL 8.4	サポートされません	サポート外	サポート対象	サポート
MySQL 8.0	サポートされません	サポート外	サポート対象	サポート対象
MySQL 5.7	サポート対象	サポート対象	サポート	サポートされていません

Amazon RDS の MySQL DB インスタンスへの特定のユーザーアカウントに SSL/TLS を要求する

Amazon RDS 上の MySQL DB インスタンスへの特定のユーザーアカウント接続に SSL/TLS 暗号化を要求できます。機密情報を不正アクセスや傍受から保護することは、データの機密性が懸念される場合にセキュリティポリシーを適用する上で非常に重要です。

特定のユーザーのアカウントに SSL/TLS 接続を要求するには、MySQL のバージョンに応じて、次のいずれかのステートメントを使用して、ユーザーアカウントの `encrypted_user` に SSL/TLS 接続を要求します。

これを行うには、以下のステートメントを実行します。

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

MySQL での SSL/TLS 接続の詳細については、MySQL ドキュメントで「暗号化された接続の使用」を参照してください。

Amazon RDS 上の MySQL DB インスタンスへのすべての接続に SSL/TLS を要求する

MySQL DB インスタンスへのすべてのユーザー接続で SSL/TLS を使用するように、require_secure_transport パラメータを使用して要求します。デフォルトでは、require_secure_transport パラメータが OFF に設定されています。require_secure_transport パラメータを ON に設定すれば、DB インスタンスへの接続で SSL/TLS を必須にすることができます。

require_secure_transport パラメータの値は、DB インスタンスの DB パラメータグループを更新することで設定できます。変更を有効にするために、DB インスタンスを再起動する必要はありません。

DB インスタンスに対して require_secure_transport パラメータが ON に設定されている場合、データベースクライアントが暗号化された接続を確立できれば、データベースクライアントはそのクラスターに接続できます。それ以外の場合は、次のようなエラーメッセージがクライアントに返されます。

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

require_secure_transport パラメータの詳細については、MySQL のドキュメントを参照してください。

MySQL コマンドラインクライアントからの SSL/TLS による Amazon RDS 上の MySQL DB インスタンスへの接続 (暗号化)

mysql クライアントプログラムのパラメータは、使用している MySQL または MariaDB のバージョンによって若干異なります。

使用しているバージョンを確認するには、--version オプションを指定しながら mysql コマンドを実行します。次の例の出力では、MariaDB のクライアントプログラムが使用されていることが確認できます。

```
$ mysql --version
mysql  Ver 15.1 Distrib 10.5.15-MariaDB, for osx10.15 (x86_64) using readline 5.1
```

Amazon Linux、CentOS、SUSE、Debianなど、ほとんどのLinux ディストリビューションでは、MySQL をMariaDB に置き換えており、mysql バージョンは MariaDB と同じものを使用しています。

SSL/TLS を使用して DB インスタンスに接続するには、以下のステップを実行します。

MySQL コマンドラインクライアントを使用して SSL/TLS で DB インスタンスに接続するには

1. すべての AWS リージョン で使用できるルート証明書をダウンロードします。

   証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

2. MySQL コマンドラインクライアントを使用して、SSL/TLS 暗号化を使用して DB インスタンスに接続します。-h パラメータは、DB インスタンスの DNS 名 (エンドポイント) に置き換えます。--ssl-ca パラメータは、必要に応じて SSL/TLS 証明書のファイル名に置き換えます。-P パラメータは、使用中の DB インスタンスのポートに置き換えます。-u パラメータでは、マスターユーザーなどの有効なデータベースユーザーのユーザー名に置き換えます。プロンプトが表示されたら、マスターユーザーパスワードを入力します。

   次の例は、MySQL 5.7 以降で --ssl-ca パラメータを使用しながら、クライアントを起動する方法を示しています。

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=REQUIRED -P 3306 -u myadmin -p
   ```

   SSL/TLS 接続で DB インスタンスのエンドポイントを SSL/TLS 証明書のエンドポイントと照合することを義務付けるには、次のコマンドを入力します。

   ```
   mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl-mode=VERIFY_IDENTITY -P 3306 -u myadmin -p
   ```

次の例は、MariaDB で --ssl-ca パラメータを使用しながら、クライアントを起動する方法を示しています。

```
mysql -h mysql-instance1.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=global-bundle.pem --ssl -P 3306 -u myadmin -p
```

3. プロンプトが表示されたら、マスターユーザーパスワードを入力します。

以下のような出力結果が表示されるはずです。

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9738
Server version: 8.0.28 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

新しい SSL/TLS 証明書を使用して MySQL DB インスタンスに接続するようにアプリケーションを更新する

2023 年 1 月 13 日に Amazon RDS は、Secure Socket Layer または Transport Layer Security (SSL/TLS) を使用して RDS DB インスタンスに接続するための新しい認証局 (CA) 証明書を公開しました。ここでは、新しい証明書を使用するためのアプリケーションの更新について説明します。

このトピックでは、クライアントアプリケーションが SSL/TLS を使用して DB インスタンスに接続されているかどうかを判断できます。該当する場合はさらに、それらのアプリケーションの接続に証明書の検証が必要かどうかを確認できます。

> **Note**
>
> 一部のアプリケーションは、サーバー上の証明書を正常に検証できる場合にのみ、MySQL DB インスタンスに接続されるように設定されています。そのようなアプリケーションの場合は、新しい CA 証明書を含むように、クライアントアプリケーションの信頼ストアを更新する必要があります。
>
> disabled、preferred、および required の SSL モードを指定できます。preferred SSL モードを使用し、CA 証明書が存在しないか、最新でない場合、接続は SSL を使用しない状態にフォールバックし、暗号化なしで接続が行われます。

preferred モードの使用を避けることをお勧めします。preferred モードでは、接続時に無効な証明書が検出されると、暗号化の使用が停止し、暗号化せずに続行されます。

クライアントアプリケーションの信頼ストアで CA 証明書を更新した後、DB インスタンスで証明書をローテーションできます。これらの手順を開発環境またはステージング環境でテストしてから、本番環境で実装することを強くお勧めします。

証明書のローテーションの詳細については、「SSL/TLS 証明書のローテーション」を参照してください。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。MySQL DB インスタンスで SSL/TLS を使用する方法については、「Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート」を参照してください。

トピック

- SSL を使用して MySQL DB インスタンスに接続しているアプリケーションがあるかどうかの確認
- クライアントが接続するために証明書の検証が必要かどうかの確認
- アプリケーション信頼ストアの更新
- SSL 接続を確立するための Java コードの例

SSL を使用して MySQL DB インスタンスに接続しているアプリケーションがあるかどうかの確認

Amazon RDS for MySQL バージョン 5.7、8.0、または 8.4 を使用しており、Performance Schema が有効になっている場合は、次のクエリを実行すると、接続で SSL/TLS が使用されているかどうかを確認できます。Performance Schemaを有効にする方法については、MySQL ドキュメントの「Performance Schemaクイックスタート」を参照してください。

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

この出力例では、admin で SSL が使用されているため、独自のセッション (webapp1) と、ログインしているアプリケーションの両方を確認できます。

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

クライアントが接続するために証明書の検証が必要かどうかの確認

JDBC クライアントおよび MySQL クライアントを接続するために証明書の検証が必要かどうかを確認できます。

JDBC

MySQL Connector/J 8.0 を使用した次の例では、アプリケーションの JDBC 接続プロパティをチェックして、正常な接続に有効な証明書が必要かどうかを確認する 1 つの方法を示します。MySQL のすべての JDBC 接続オプションの詳細については、MySQL ドキュメントの「設定プロパティ」を参照してください。

MySQL Connector/J 8.0 を使用する場合、次の例のように、接続プロパティで `sslMode` が `VERIFY_CA` または `VERIFY_IDENTITY` に設定されている場合、SSL 接続では DB サーバー証明書に対する検証が必要です。

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> MySQL Java Connector v5.1.38 以降または MySQL Java Connector v8.0.9 以降を使用してデータベースに接続する場合、データベースへの接続時に SSL/TLS を使用するようにアプリケーションを明示的に設定しなくても、これらのクライアントドライバーはデフォルトで

SSL/TLS を使用します。また、SSL/TLS の使用時に証明書検証が部分的に実行され、データベースサーバー証明書の有効期限が切れていると、接続に失敗します。

MySQL

MySQL クライアントの以下の例では、スクリプトの MySQL 接続をチェックして、正常な接続に有効な証明書が必要かどうかを確認する 2 つの方法を示します。MySQL クライアントで使用するすべての接続オプションの詳細については、MySQL ドキュメントの「Client-Side Configuration for Encrypted Connections」を参照してください。

MySQL Client バージョン 5.7 以降を使用する場合に、次の例のように、`--ssl-mode` オプションに `VERIFY_CA` または `VERIFY_IDENTITY` を指定している場合は、SSL 接続にサーバー CA 証明書に対する検証が必要です。

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

アプリケーション信頼ストアの更新

MySQL アプリケーションの信頼ストアの更新については、MySQL ドキュメントの「Installing SSL Certificates」を参照してください。

ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

証明書をインポートするサンプルスクリプトについては、証明書を信頼ストアにインポートするためのサンプルスクリプト を参照してください。

> **Note**
>
> 信頼ストアを更新するとき、新しい証明書を追加できるだけでなく、古い証明書を保持できます。

アプリケーションで mysql JDBC ドライバーを使用している場合は、アプリケーションで以下のプロパティを設定します。

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

アプリケーションを起動するとき、以下のプロパティを設定します。

```
java -Djavax.net.ssl.trustStore=/path_to_trust_store/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_trust_store_password com.companyName.MyApplication
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

SSL 接続を確立するための Java コードの例

次のコード例では、JDBC を使用してサーバー証明書を検証する SSL 接続のセットアップ方法を示します。

```
public class MySQLSSLTest {

        private static final String DB_USER = "username";
        private static final String DB_PASSWORD = "password";
        // This trust store has only the prod root ca.
        private static final String TRUST_STORE_FILE_PATH = "file-path-to-trust-store";
        private static final String TRUST_STORE_PASS = "trust-store-password";

        public static void test(String[] args) throws Exception {
            Class.forName("com.mysql.jdbc.Driver");

            System.setProperty("javax.net.ssl.trustStore", TRUST_STORE_FILE_PATH);
            System.setProperty("javax.net.ssl.trustStorePassword", TRUST_STORE_PASS);

            Properties properties = new Properties();
            properties.setProperty("sslMode", "VERIFY_IDENTITY");
```

```
            properties.put("user", DB_USER);
            properties.put("password", DB_PASSWORD);


            Connection connection = null;
            Statement stmt = null;
            ResultSet rs = null;
            try {
                connection =
 DriverManager.getConnection("jdbc:mysql://mydatabase.123456789012.us-
east-1.rds.amazonaws.com:3306",properties);
                stmt = connection.createStatement();
                rs=stmt.executeQuery("SELECT 1 from dual");
            } finally {
                if (rs != null) {
                    try {
                        rs.close();
                    } catch (SQLException e) {
                    }
                }
                if (stmt != null) {
                   try {
                        stmt.close();
                    } catch (SQLException e) {
                   }
                }
                if (connection != null) {
                    try {
                        connection.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            }
            return;
        }
    }
```

> **Important**
>
> データベース接続で SSL/TLS を使用することを決定し、アプリケーションの信頼ストアを更新したら、rds-ca-rsa2048-g1 証明書を使用するようにデータベースを更新できます。ス

テップについては、「DB インスタンスまたはクラスターの変更による CA 証明書の更新」のステップ 3 を参照してください。
セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

Amazon RDS for MySQL での Kerberos 認証の使用

MySQL DB インスタンスに接続するユーザーを Kerberos 認証を使用して認証できるようになりました。DB インスタンスは AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) を使用して Kerberos 認証を有効にします。信頼するドメインに参加している MySQL DB インスタンスに対してユーザーが認証を行う場合、認証リクエストが転送されます。転送されたリクエストは、Directory Service で作成したドメインディレクトリに移動します。

同じディレクトリにすべての認証情報を保持することで時間と労力を節約できます。この方法により、複数の DB インスタンスの認証情報を一元的に保存および管理できます。また、ディレクトリを使用することで、セキュリティプロファイル全体を向上できます。

リージョンとバージョンの可用性

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。Kerberos 認証を使用した Amazon RDS のバージョンとリージョンの可用性の詳細については、「Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン」を参照してください。

MySQL DB インスタンスの Kerberos 認証の設定の概要

MySQL DB インスタンスの Kerberos 認証を設定するには、以下の一般的なステップを実行します (詳細は後で説明します)。

1. AWS Managed Microsoft AD を使用して AWS Managed Microsoft AD ディレクトリを作成します。AWS マネジメントコンソール、AWS CLI、Directory Service を使用して、ディレクトリを作成できます。作成の詳細については、AWS Directory Service 管理ガイドの「AWS Managed Microsoft AD ディレクトリの作成」を参照してください。
2. マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する AWS Identity and Access Management (IAM) ロールの作成 このロールにより Amazon RDS はディレクトリを呼び出すことができます。

ロールによるアクセスを許可するには、AWS Security Token Service (AWS STS) エンドポイントを AWS アカウントの AWS リージョン でアクティベートする必要があります。AWS STS エンドポイントはすべての AWS リージョン でデフォルトでアクティブになっているため、他のアクションを実行せずに、エンドポイントを使用することができます。詳細については、「IAM ユーザーガイド」の「AWS リージョンでの AWS STS のアクティブ化と非アクティブ化」を参照してください。

3. Microsoft Active Directory のツールを使用して、AWS Managed Microsoft AD ディレクトリでユーザーとグループを作成し、設定します。Active Directory にユーザーを作成する方法の詳細については、AWS 管理ガイドの「AWS Directory Service マネージド Microsoft AD でユーザーとグループを管理する」を参照してください。
4. MySQL DB インスタンスを作成または変更します。作成リクエストで CLI または RDS API を使用する場合は、`Domain` パラメータでドメイン識別子を指定します。ディレクトリの作成時に生成された `d-*` 識別子と、作成したロールの名前を使用します。

   既存の MySQL DB インスタンスを変更して Kerberos 認証を使用する場合は、DB インスタンスのドメインパラメータと IAM ロールパラメータを設定します。ドメインディレクトリと同じ VPC で DB インスタンスを見つけます。
5. Amazon RDS マスターユーザー認証情報を使用して、MySQL DB インスタンスに接続します。`CREATE USER` 句の `IDENTIFIED WITH 'auth_pam'` を使用して MySQL でユーザーを作成します。この方法で作成したユーザーは、Kerberos 認証を使用して MySQL DB インスタンスにログインできます。

MySQL DB インスタンスの Kerberos 認証の設定

AWS Managed Microsoft AD を使用して、MySQL DB インスタンスの Kerberos 認証を設定します。Kerberos 認証を設定するには、次のステップに従います。

ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する

Directory Service はフルマネージド型の Active Directory を AWS クラウド内に作成します。AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって 2 つのドメインコントローラーと 2 つのドメインネームシステム (DNS) サーバーを作成します。ディレクトリサーバーは、VPC 内の異なるサブネットで作成されます。この冗長性によって、障害が発生してもディレクトリにアクセス可能な状態を維持できます。

AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって自動的に以下のタスクを実行します。

- VPC 内に Active Directory を設定します。
- 「Admin」のユーザー名と指定されたパスワードを使用して、ディレクトリ管理者アカウントを作成します。このアカウントを使用してディレクトリを管理します。

> **Note**
>
> このパスワードは必ず保存してください。Directory Service は保存しません。パスワードはリセットできますが、取得することはできません。

- ディレクトリコントローラー用セキュリティグループを作成します。

AWS Managed Microsoft AD を起動すると、AWS は組織単位 (OU) を作成します。OU にはディレクトリのオブジェクトがすべて含まれています。この OU には、ディレクトリの作成時に入力した NetBIOS 名がドメインルートにあります。ドメインルートは AWS が所有し、管理します。

AWS Managed Microsoft AD ディレクトリに作成した管理者アカウントには、OU に関する以下の代表的な管理業務用のアクセス権限があります。

- ユーザーを作成、更新、削除する
- ファイルやプリントサーバーなどのドメインにリソースを追加して、追加したリソースへのアクセス許可を OU のユーザーとグループに割り当てる
- 追加の OU やコンテナを作成する
- 権限を委譲する
- 削除されたオブジェクトを Active Directory のごみ箱から元に戻す
- Active Directory Web Service で AD と DNS Windows PowerShell モジュールを実行する

管理者アカウントには、ドメイン全体に関する以下のアクティビティを実行する権限もあります。

- DNS 設定 (レコード、ゾーン、フォワーダーの追加、削除、更新) を管理する
- DNS イベントログを参照する
- セキュリティイベントログを参照する

AWS Managed Microsoft AD でディレクトリを作成するには

1. AWS マネジメントコンソール にサインインし、Directory Service コンソール (https://console.aws.amazon.com/directoryservicev2/)を開きます。

2. ナビゲーションペインで、[Directories]、[Set up Directory] の順に選択します。
3. [AWS Managed Microsoft AD] を選択します。現状では、AWS Managed Microsoft AD が Amazon RDS で使用できる唯一のオプションです。
4. 次の情報を入力します。

   ディレクトリの DNS 名

   ディレクトリの完全修飾名 (例: **corp.example.com**)。

   ディレクトリの NetBIOS 名

   ディレクトリの短縮名 (例: **CORP**)。

   ディレクトリの説明

   (オプション) ディレクトリの説明。

   Admin パスワード

   ディレクトリ管理者のパスワードです。ディレクトリの作成プロセスでは、ユーザー名「Admin」とこのパスワードを使用して管理者アカウントが作成されます。

   ディレクトリ管理者のパスワードに「admin」の単語を含めることはできません。パスワードは大文字と小文字を区別し、8-64 文字にします。また、以下の 4 つのカテゴリうち 3 つから少なくとも 1 文字を含める必要があります。

   - 小文字 (a ~ z)
   - 大文字 (A ~ Z)
   - 数字 (0 ~ 9)
   - アルファベット以外の文字 (~!@#$%^&*_-+=`|\(){}[]:;'"<>,.?/)

   パスワードを確認

   管理者のパスワードをもう一度入力します。
5. [Next (次へ)] を選択します。
6. [Networking] セクションに次の情報を入力し、[Next] を選択します。

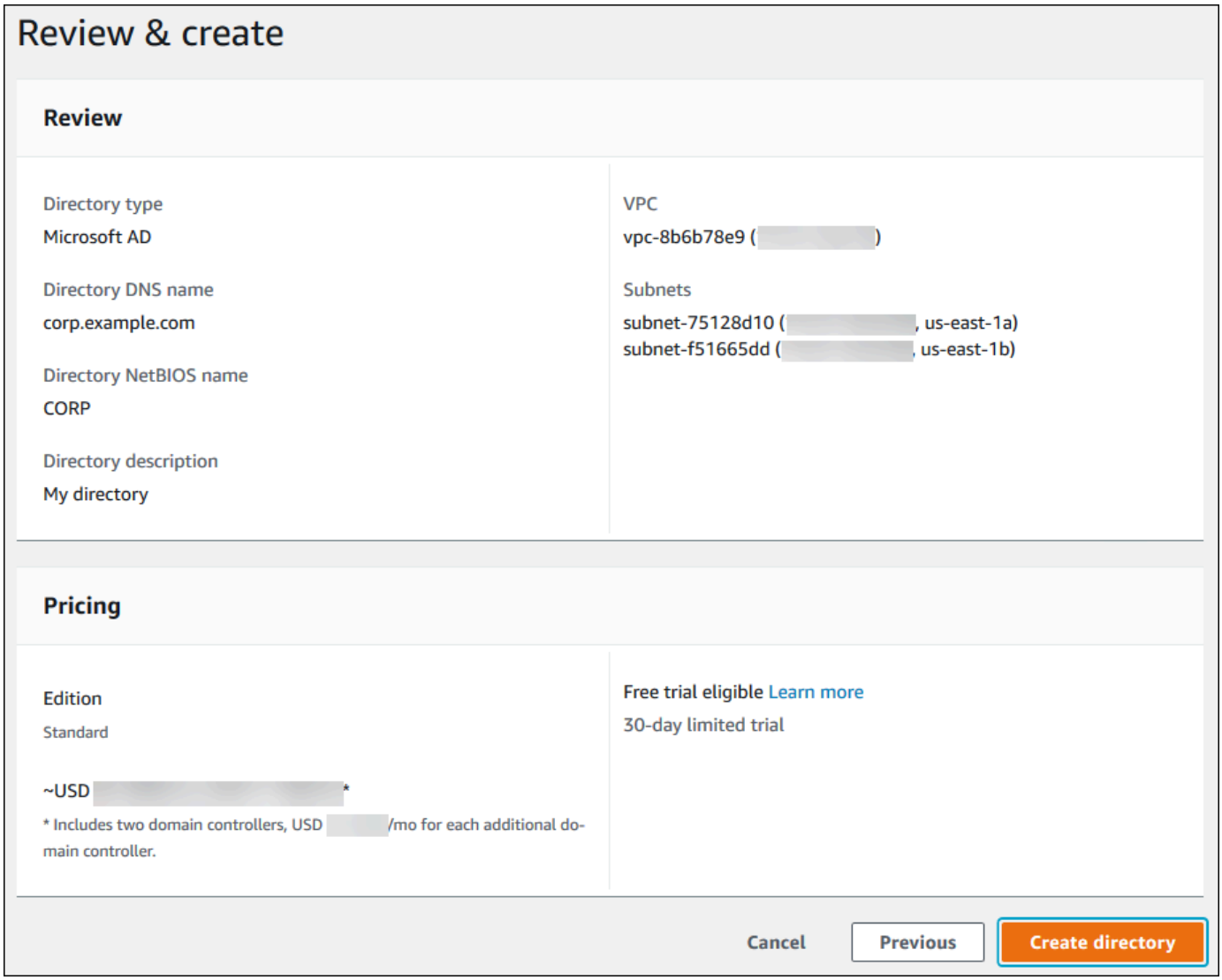
   VPC

   ディレクトリ用の VPC。この同じ VPC に MySQL DB インスタンスを作成します。

Subnets

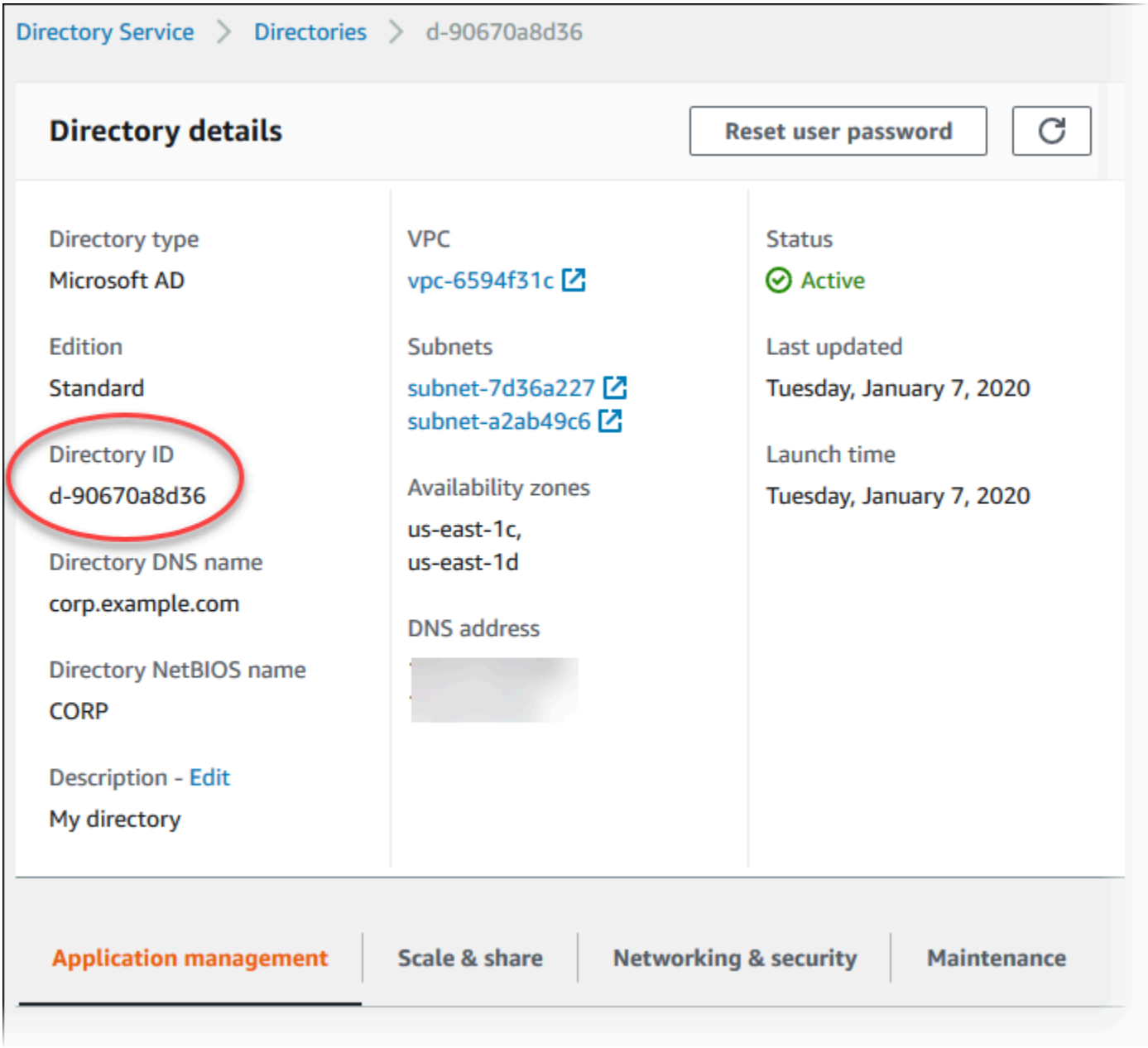
ディレクトリサーバーのサブネット。2 つのサブネットは、異なるアベイラビリティーゾーンに存在している必要があります。

7. ディレクトリ情報を確認し、必要な変更を加えます。情報が正しい場合は、[Create directory (ディレクトリの作成)] を選択します。



ディレクトリが作成されるまで、数分かかります。正常に作成されると、[Status] 値が [Active] に変わります。

ディレクトリに関する情報を表示するには、ディレクトリの一覧で、ディレクトリ名を選択します。[Directory ID] の値を書き留めます。この値は、MySQL DB インスタンスを作成または変更するときに必要になります。



ステップ 2: Amazon RDS 用の IAM ロールを作成する

Amazon RDS が Directory Service を呼び出すには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する IAM ロールが必要です。このロールにより、Amazon RDS は Directory Service への呼び出しを行うことができます。

AWS マネジメントコンソール を使用して DB インスタンスが作成され、コンソールユーザーが `iam:CreateRole` アクセス許可を持っている場合、コンソールはこのロールを自動的に作成します。この場合、ロール名は `rds-directoryservice-kerberos-access-role` です。それ以外の場合は、IAM ロールを手動で作成する必要があります。IAM ロールを作成する場合、[`Directory`

Service] を選択し、それに AWS マネージドポリシー AmazonRDSDirectoryServiceAccess をアタッチします。

サービス用の IAM ロールを作成する方法の詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Note**
>
> RDS for SQL Serverの Windows 認証に使用される IAM ロールは、RDS for MySQL には使用できません。

マネージド IAM ポリシー AmazonRDSDirectoryServiceAccess を使用する代わりに、必要なアクセス許可を使用してポリシーを作成することもできます。これを行うには、IAM ロールに次の IAM 信頼ポリシーが必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

また、ロールには、以下の IAM ロールポリシーも必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

オプトインの AWS リージョンの場合は、IAM ロールの信頼ポリシーでリージョン固有のサービスプリンシパルを使用します。これらのリージョンでサービスの信頼ポリシーを作成するときは、サービスプリンシパルでリージョンコードを指定します。

次の例では、リージョン固有のサービスプリンシパルを含む信頼ポリシーを示します。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

REGION-CODE を、特定のリージョンのコードに置き換えます。例えば、アジアパシフィック (メルボルン) リージョンでは、次のサービスプリンシパルを使用します。

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

イスラエル (テルアビブ) より後に開設された AWS リージョン (カナダ西部 (カルガリー) 以降) では、エンジン固有のサービスプリンシパル `directoryservice-`*`engine`*`.rds.`*`region-code`*`.amazonaws.com` を使用します。ここで、*`engine`* は、`mysql` です。例えば、カナダ西部 (カルガリー) リージョンの場合。

```
"Service": [
  "directoryservice-mysql.rds.ca-west-1.amazonaws.com",
  "rds.ca-west-1.amazonaws.com"
]
```

ステップ 3: ユーザーを作成して設定する

Active Directory ユーザーとコンピュータツールを使用してユーザーを作成できます。このツールは、Active Directory Domain Services ツールおよび Active Directory Lightweight Directory Services ツールの一部です。ユーザーは、ディレクトリにアクセスできる個別の人またはエンティティを表します。

Directory Service ディレクトリにユーザーを作成するには、Microsoft Windows ベースの Amazon EC2 インスタンスに接続している必要があります。このインスタンスは Directory Service ディレクトリのメンバーであることが必要です。また、このインスタンスにユーザーを作成する権限を持つユーザーとしてログインしている必要があります。詳細については、AWS Managed Microsoft AD Directory Service 管理ガイドのAWS のユーザーとグループの管理を参照してください。

ステップ 4: MySQL DB インスタンスを作成または変更する

ディレクトリ用の MySQL DB インスタンスを作成または変更します。コンソール、CLI、RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これには以下の 2 つの方法があります。

- コンソール、create-db-instance CLI コマンド、CreateDBInstance RDS API オペレーションを使用して新しい MySQL DB インスタンスを作成します。

手順については、「Amazon RDS DB インスタンスの作成」を参照してください。

- コンソール、modify-db-instance CLI コマンド、ModifyDBInstance RDS API オペレーションを使用して、既存の MySQL DB インスタンスを変更します。

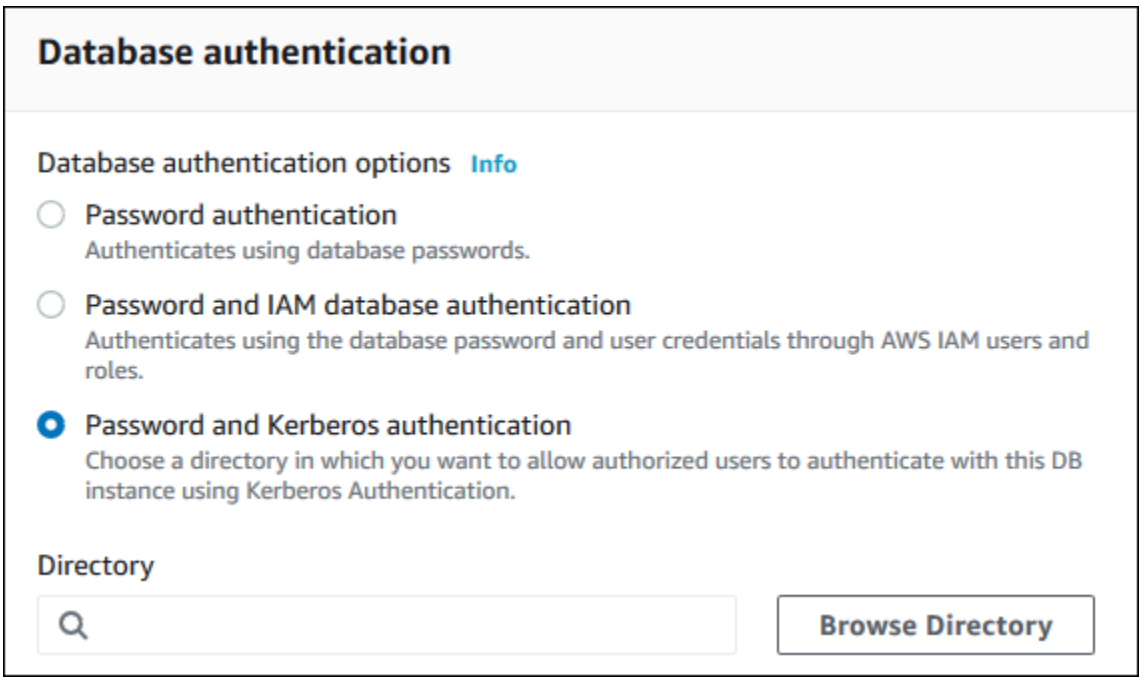
  手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、DBスナップショットからDBインスタンスを復元する CLI コマンド、または RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから MySQL DB インスタンスを復元します。

  手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、または RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、MySQL DB インスタンスをポイントインタイムに復元します。

  手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

Kerberos 認証は、VPC 内の MySQL DB インスタンスにのみサポートされています。DB インスタンスは、ディレクトリと同じ VPC または異なる VPC 内にあります。DB インスタンスがディレクトリと通信できるように、ディレクトリの VPC 内の出力を許可するセキュリティグループを使用します。

DB インスタンスを作成、変更、復元するためにコンソールを使用する場合は、データベースの認証セクションの [パスワードと Kerberos 認証] を選択します。[ディレクトリの参照] を選択してディレクトリを選択するか、[新しいディレクトリの作成] を選択します。



AWS CLI または RDS API を使用して DB インスタンスとディレクトリを関連付けます。作成したドメインディレクトリを DB インスタンスで使用できるようにするには、以下のパラメータが必要です。

- `--domain` パラメータには、ディレクトリの作成時に生成されたドメイン識別子 ("d-*" 識別子) を使用します。
- `--domain-iam-role-name` パラメータには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する作成済みのロールを使用します。

例えば、以下の CLI コマンドはディレクトリを使用するように DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
```

```
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Kerberos 認証を有効化するために DB インスタンスを変更する場合、変更後 DB インスタンスを再起動します。

ステップ 5: Kerberos 認証の MySQL ログインを作成する

Amazon RDS マスターユーザー認証情報を使用して、他の DB インスタンスと同様に MySQL DB インスタンスに接続します。DB インスタンスは、AWS Managed Microsoft AD ドメインに接続されています。したがって、このドメインの Active Directory ユーザーから MySQL ログインおよびユーザーをプロビジョンできます。データベースへのアクセス許可は、これらのログインに対して付与および取り消されるスタンダードの MySQL アクセス許可を通じて管理されます。

Active Directory ユーザーに対して MySQL での認証を許可できます。これを行うには、まず Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同様に MySQL DB インスタンスに接続します。ログインしたら、次のコマンドを実行し、MySQL で PAM (Pluggable Ahthentication Modules) を使用して外部認証ユーザーを作成します。*testuser* をユーザー名に置き換えます。

```
CREATE USER 'testuser'@'%' IDENTIFIED WITH 'auth_pam';
```

これでドメインのユーザー (人とアプリケーションの両方) は、Kerberos 認証を使用してドメインに参加しているクライアントマシンから DB インスタンスに接続できます。

> **Important**
>
> PAM 認証を使用する場合は、クライアントが SSL/TLS 接続を使用することを強くお勧めします。SSL/TLS 接続を使用しないと、パスワードがクリアテキストとして送信される場合があります。AD ユーザーに SSL/TLS による暗号化接続を要求するには、次のコマンドを実行して *testuser* をユーザー名に置き換えます。
>
> ```
> ALTER USER 'testuser'@'%' REQUIRE SSL;
> ```

詳細については、「Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート」を参照してください。

ドメインの DB インスタンスの管理

CLI または RDS API を使用して、DB インスタンスおよび、DB インスタンスとマネージド Active Directory との関係を管理できます。例えば、Kerberos 認証用に Active Directory を関連付けたり、Active Directory の関連付けを解除して Kerberos 認証を無効にしたりできます。さらに、DB インスタンスを外部認証する Active Directory を別の Active Directory に変更することもできます。

例えば、Amazon RDS API を使用して次を実行できます。

- 失敗したメンバーシップの Kerberos 認証の有効化を再試行するには、`ModifyDBInstance` API オペレーションを使用し、現在のメンバーシップのディレクトリ ID を指定します。
- メンバーシップの IAM ロール名を更新するには、`ModifyDBInstance` API オペレーションを使用し、現在のメンバーシップのディレクトリ ID と新しい IAM ロールを指定します。
- DB インスタンスの Kerberos 認証を無効にするには、`ModifyDBInstance` API オペレーションを使用し、ドメインパラメータとして `none` を指定します。
- ドメイン間で DB インスタンスを移動するには、`ModifyDBInstance` API オペレーションを使用し、新しいドメインのドメイン識別子をドメインパラメータとして指定します。
- 各 DB インスタンスのメンバーシップを一覧表示するには、`DescribeDBInstances` API オペレーションを使用します。

ドメインのメンバーシップを理解する

DB インスタンスを作成または変更すると、そのインスタンスはドメインのメンバーになります。DB インスタンスのドメインメンバーシップのステータスを表示するには、describe-db-instances CLI コマンドを実行します。DB インスタンスのステータスは、以下のいずれかです。

- `kerberos-enabled` - DB インスタンスは Kerberos 認証を有効化しました。
- `enabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を有効化中です。
- `pending-enable-kerberos` - この DB インスタンスでは、Kerberos 認証の有効化が保留になっています。
- `pending-maintenance-enable-kerberos` - AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の有効化を試みます。

- `pending-disable-kerberos` - この DB インスタンスでは、Kerberos 認証の無効化が保留になっています。
- `pending-maintenance-disable-kerberos` - AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の無効化を試みます。
- `enable-kerberos-failed` - 設定の問題により、AWS は DB インスタンス上の Kerberos 認証を有効化できませんでした。DB インスタンスの変更コマンドを再発行する前に、設定を確認して修正してください。
- `disabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を無効化中です。

ネットワーク接続の問題や正しくない IAM ロールのために、Kerberos 認証を有効化するリクエストは失敗する可能性があります。例えば、DB インスタンスを作成するか、既存の DB インスタンスを変更し、Kerberos 認証を有効化しようとして失敗したとします。この場合は、変更コマンドを再発行するか、新しく作成した DB インスタンスを変更してドメインに参加させます。

Kerberos 認証を使用した MySQL への接続

Kerberos 認証を使用して MySQL に接続するには、Kerberos 認証タイプを使用してログインする必要があります。

Kerberos 認証を使用して接続できるデータベースユーザーを作成するには、`IDENTIFIED WITH` ステートメントで `CREATE USER` 句を使用します。手順については、「ステップ 5: Kerberos 認証の MySQL ログインを作成する」を参照してください。

エラーを回避するには、MariaDB `mysql` クライアントを使用します。MariaDB のソフトウェアは https://downloads.mariadb.org/ でダウンロードできます。

コマンドプロンプトで、MySQL DB インスタンスに関連付けられているエンドポイントの 1 つに接続します。「MySQL DB インスタンスへの接続」の一般的な手順に従います。パスワードの入力を求められたら、そのユーザー名に関連付けられている Kerberos パスワードを入力します。

MySQL DB インスタンスの復元とドメインへの追加

MySQL DB インスタンスの DB スナップショットを復元するか、ポイントインタイム復元を実行して、インスタンスをドメインに追加できます。DB インスタンスを復元したら、「ステップ 4: MySQL DB インスタンスを作成または変更する」で説明している手順に従って DB インスタンスを変更し、DB インスタンスをドメインに追加します。

Kerberos 認証に関する MySQL の制限

MySQL の Kerberos 認証には、以下の制限が適用されます。

- AWS Managed Microsoft AD のみがサポートされています。ただし、同じ AWS リージョン の異なるアカウントによって所有されている共有 Managed Microsoft AD ドメインに、RDS for MySQL DB インスタンスを接続できます。
- この機能を有効にした後で、DB インスタンスを再起動する必要があります。
- ドメイン名の長さは 61 文字以下にする必要があります。
- Kerberos 認証と IAM 認証を同時に有効にすることはできません。MySQL DB インスタンスに対していずれか 1 つの認証方法を選択します。
- この機能を有効にした後では DB インスタンスポートを変更しません。
- リードレプリカでは Kerberos 認証を使用しません。
- Kerberos 認証を使用している MySQL DB インスタンスのマイナーバージョン自動アップグレードがオンになっている場合、Kerberos 認証をオフにし、自動アップグレード後に再度オンにする必要があります。マイナーバージョン自動アップグレードの詳細については、「RDS for MySQL のマイナーバージョンの自動アップグレード」を参照してください。
- この機能が有効になっている DB インスタンスを削除するには、まず機能を無効にします。これを行うには、DB インスタンスに対して `modify-db-instance` CLI コマンドを実行し、`--domain` パラメータに `none` を指定します。

  CLI または RDS API を使用してこの機能が有効になっている DB インスタンスを削除する場合は、遅延が予想されます。
- RDS for MySQL は、オンプレミスまたはセルフホスト AD と AWS Managed Microsoft AD の間のフォレストの信頼関係における Kerberos 認証をサポートしていません。

Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上

Amazon RDS Optimized Reads によって、RDS for MySQL の高速クエリ処理を実現できます。RDS Optimized Reads を使用する RDS for MySQL DB インスタンスまたはマルチ AZ DB クラスターは、これを使用しない DB インスタンスまたはクラスターに比べて、クエリ処理を最大 2 倍高速化できます。

トピック

- RDS Optimized Reads の概要
- RDS Optimized Reads のユースケース
- RDS Optimized Reads のベストプラクティス
- RDS Optimized Reads の使用
- RDS Optimized Reads を使用する DB インスタンスのモニタリング
- RDS Optimized Reads についての制限事項

RDS Optimized Reads の概要

RDS Optimized Reads が有効になっている RDS for MySQL DB インスタンスまたはマルチ AZ DB クラスターを使用する場合、インスタンスストアを使用することでクエリのパフォーマンスが高速化されます。インスタンスストアは、DB インスタンスまたはマルチ AZ DB クラスターに一時ブロックレベルのストレージを提供します。ストレージは、ホストサーバーに物理的にアタッチされた不揮発性メモリエクスプレス (NVMe) によるソリッドステートドライブ (SSD) にあります。このストレージは、低レイテンシー、優れたランダム I/O パフォーマンス、高いシーケンシャル読み取りスループットを実現するために最適化されています。

DB インスタンスまたはマルチ AZ DB クラスターが db.m5d や db.m6gd などのインスタンスストアを含む DB インスタンスクラスを使用する場合、RDS Optimized Reads はデフォルトで有効になっています。RDS Optimized Reads では、一部の一時オブジェクトがインスタンスストアに保存されます。これらの一時オブジェクトには、内部一時ファイル、内部オンディスク一時テーブル、メモリマップファイル、バイナリログ (binlog) キャッシュファイルが含まれます。インスタンスストアの詳細については、Linux インスタンス向け Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 インスタンスストア」を参照してください。

MySQL でクエリ処理用の一時オブジェクトを生成するワークロードは、インスタンスストアを利用してクエリ処理を高速化できます。このタイプのワークロードには、ソート、ハッシュ集約、高負荷

結合、共通テーブル式 (CTE)、インデックス付けされていない列に対するクエリが含まれます。これらのインスタンスストアボリュームによって、永続的な Amazon EBS ストレージに使用されるストレージ設定に関係なく、より高い IOPS とパフォーマンスを提供します。RDS Optimized Reads は一時オブジェクトのオペレーションをインスタンスストアにオフロードするため、永続的ストレージ (Amazon EBS) の 1 秒あたりの入出力オペレーション (IOPS) またはスループットを、永続オブジェクトのオペレーションに使用できるようになりました。これらのオペレーションには、通常のデータファイルの読み取りと書き込み、フラッシュやバッファ挿入のマージなどのバックグラウンドエンジンオペレーションが含まれます。

> **Note**
>
> 手動の RDS スナップショットと自動の RDS スナップショットの両方に含まれているのは、永続オブジェクトのエンジンファイルのみです。インスタンスストアで作成された一時オブジェクトは RDS スナップショットには含まれません。

RDS Optimized Reads のユースケース

クエリの実行を内部テーブルやファイルなどの一時オブジェクトに大きく依存するワークロードがある場合は、RDS Optimized Reads を有効にすると便利です。RDS Optimized Reads の候補となるユースケースは次のとおりです。

- 複雑なテーブル共通式 (CTE)、派生テーブル、グループ化オペレーションを使用して分析クエリを実行するアプリケーション
- 最適化されていないクエリで大量の読み取りトラフィックを処理するリードレプリカ
- `GROUP BY` 句や `ORDER BY` 句を含むクエリなど、複雑なオペレーションを伴うオンデマンドまたは動的レポートクエリを実行するアプリケーション
- クエリ処理に内部テンポラリテーブルを使用するワークロード

  エンジンステータス変数 `created_tmp_disk_tables` をモニタリングすることで、DB インスタンスに作成されたディスクベースの一時テーブルの数を判断できます。
- 中間結果を格納するために、直接またはプロシージャ内で大規模な一時テーブルを作成するアプリケーション
- インデックス付けされていない列をグループ化または順序付けするデータベースクエリ

RDS Optimized Reads のベストプラクティス

RDS Optimized Reads を使用するベストプラクティスは次のとおりです。

- インスタンスストアが実行中にストレージ不足によって失敗した場合に備えて、読み取り専用クエリの再試行ロジックを追加します。
- CloudWatch メトリクスの `FreeLocalStorage` を使用して、インスタンスストアで使用可能なストレージ容量をモニタリングします。DB インスタンスのワークロードが原因でインスタンスストアが上限に達している場合は、より大きな DB インスタンスクラスを使用するように DB インスタンスを変更します。
- DB インスタンスまたはマルチ AZ DB クラスターに十分なメモリがあるのにインスタンスストアのストレージ制限に達している場合は、`binlog_cache_size` 値を増やしてセッション固有のバイナリログエントリをメモリに保持します。この設定により、ディスク上の一時バイナリログキャッシュファイルにバイナリログエントリの書き込みを防止します。

  `binlog_cache_size` パラメータはセッション固有です。この値は、新しいセッションごとに変更できます。このパラメータを設定すると、ピーク負荷時に DB インスタンスのメモリ使用量が増加する可能性があります。そのため、アプリケーションのワークロードのパターンと DB インスタンスで使用可能なメモリに基づいてパラメータ値を増やすことを検討してください。
- MySQL バージョン 8.0 以前の場合、`binlog_format` パラメータのデフォルト値に `MIXED` の値を使用します。トランザクションのサイズによっては、`binlog_format` を `ROW` に設定すると、インスタンスストアのバイナリログキャッシュファイルが大きくなる可能性があります。MySQL 8.4 以降の場合、`binlog_format` パラメータのデフォルト値に `ROW` の値を使用します。
- internal_tmp_mem_storage_engine パラメータを `TempTable` に設定し、temptable_max_mmap パラメータをインスタンスストアで使用可能なストレージのサイズと一致するように設定します。
- 1 つのトランザクションで大量の変更を実行することは避けてください。このような種類のトランザクションでは、インスタンスストアに大きなバイナリログキャッシュファイルが生成され、インスタンスストアが満杯になると問題が発生する可能性があります。バイナリログキャッシュファイルのストレージ使用量を最小限に抑えるために、書き込みを複数の小さなトランザクションに分割することを検討してください。
- `binlog_error_action` パラメータには `ABORT_SERVER` のデフォルト値を使用してください。そうすることで、バックアップが有効になっている DB インスタンスのバイナリログに関する問題を回避できます。

RDS Optimized Reads の使用

シングル AZ DB インスタンスデプロイ、マルチ AZ DB インスタンスデプロイまたは Multi-AZ DB クラスターデプロイで、次の DB インスタンスクラスのいずれかを使用して RDS for MySQL DB インスタンスをプロビジョニングすると、DB インスタンスは自動的に RDS Optimized Reads を使用します。

RDS Optimized Reads をオンにするには、次のいずれかの操作を行います。

- これらの DB インスタンスクラスの 1 つを使用して、RDS for MySQL DB インスタンスまたはマルチ AZ DB クラスターを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- これらの DB インスタンスクラスの 1 つを使用して、既存の RDS for MySQL DB インスタンスまたはマルチ AZ DB クラスターを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS Optimized Reads は、ローカル NVMe SSD ストレージのある DB インスタンスクラスの 1 つ以上がサポートされているすべての AWS リージョン RDS で使用できます。DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

DB インスタンスクラスの可用性は AWS リージョン によって異なります。DB インスタンスクラスが特定の AWS リージョン でサポートされているかどうかを判断するには、「the section called “AWS リージョン での DB インスタンスクラスのサポートを決定する”」を参照してください。

RDS Optimized Reads を使用しない場合は、この機能をサポートする DB インスタンスクラスを使用しないように DB インスタンスまたはマルチ AZ DB クラスターを変更してください。

RDS Optimized Reads を使用する DB インスタンスのモニタリング

RDS Optimized Reads を使用する DB インスタンスは、次の CloudWatch メトリクスでモニタリングできます。

- `FreeLocalStorage`
- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`

- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

これらのメトリクスでは、利用可能なインスタンスストアストレージ、IOPS、スループットに関するデータを提供します。これらのメトリクスの詳細については、「Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス」を参照してください。

RDS Optimized Reads についての制限事項

RDS Optimized Reads には次の制限事項が適用されます。

- RDS Optimized Reads は、以下のバージョンでサポートされています。
  - RDS for MySQL 8.0.28 以降のメジャーバージョンとマイナーバージョン

  RDS for MySQL のバージョンの詳細については、「Amazon RDS での MySQL のバージョン」を参照してください。
- RDS Optimized Reads をサポートする DB インスタンスクラスでは、一時オブジェクトの場所を永続ストレージ (Amazon EBS) に変更することはできません。
- DB インスタンスでバイナリロギングが有効になっている場合、最大トランザクションサイズはインスタンスストアのサイズによって制限されます。MySQL では、`binlog_cache_size` の値よりも多くのストレージを必要とするセッションでは、インスタンスストアに作成される一時的なバイナリログキャッシュファイルにトランザクションの変更が書き込まれます。
- トランザクションは、インスタンスストアが満杯になるとエラーになる可能性があります。

RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上

RDS Optimized Writes for MySQL によって、書き込みトランザクションのパフォーマンスを向上させることができます。RDS for MySQL データベースで RDS Optimized Writes を使用すると、書き込みトランザクションのスループットが最大 2 倍向上します。

トピック

- RDS Optimized Writes の概要
- RDS Optimized Writes の使用
- 既存のデータベースで RDS Optimized Writes を有効にする
- RDS Optimized Writes についての制限事項

RDS Optimized Writes の概要

RDS Optimized Writes を有効にすると、RDS for MySQL データベースは、データを耐久性のあるストレージにフラッシュするときに、二重書き込みバッファを必要とせずに一度だけ書き込みます。データベースは、ACID プロパティ保護機能の提供を継続することで、信頼性の高いデータベーストランザクションを実現するとともに、パフォーマンスを向上させます。

MySQL のようなリレーショナルデータベースには、原子性、一貫性、分離性、耐久性という ACID 特性があり、信頼性の高いデータベーストランザクションを実現できます。これらの特性を発揮するために、MySQL は部分的なページ書き込みエラーを防ぐ二重書き込みバッファと呼ばれるデータストレージ領域を使用しています。これらのエラーは、データベースがページを更新中に、停電時などのハードウェア障害が発生した場合に発生します。MySQL データベースは、部分的なページ書き込みを検出し、二重書き込みバッファ内のページのコピーによって回復できます。この手法で保護が可能ですが、余分な書き込みオペレーションも必要になります。MySQL の二重書き込みバッファの詳細については、MySQL ドキュメントの「二重書き込みバッファ」を参照してください。

RDS Optimized Writes を有効にすると、RDS for MySQL データベースは、データを耐久性のあるストレージにフラッシュするときに、二重書き込みバッファを使用せずに一度だけ書き込みます。RDS Optimized Writes は、RDS for MySQL データベースで書き込みの多いワークロードを実行する場合に便利です。書き込みワークロードの高いデータベースの例としては、デジタル決済、金融取引、ゲームアプリケーションに対応したデータベースなどがあります。

これらのデータベースは、AWS Nitro System を使用する DB インスタンスクラスで実行されます。これらのシステムのハードウェア構成により、データベースは 16 KiB ページを 1 ステップで

確実かつ永続的にデータファイルに直接書き込むことができます。AWS Nitro System により RDS Optimized Writes が可能になります。

新しいデータベースパラメータ `rds.optimized_writes` を設定して、RDS for MySQL データベースの RDS Optimized Writes 機能を制御できます。RDS for MySQL バージョン 8.0 および RDS for MySQL バージョン 8.4 の DB パラメータグループから、このパラメータにアクセスします。次の値を使用してパラメータを設定します。

- `AUTO` – データベースが RDS Optimized Writes をサポートしている場合は有効にします。データベースが RDS Optimized Writes をサポートしていない場合は無効にします。この設定はデフォルトです。
- `OFF` – データベースが RDS Optimized Writes をサポートしている場合でも無効にします。

RDS Optimized Writes をサポートしないエンジンバージョン、DB インスタンスクラス、ファイルシステム形式の既存のデータベースがある場合は、ブルー/グリーンデプロイを作成することでこの機能を有効にできます。詳細については、「the section called “既存のデータベースで有効にする”」を参照してください。

RDS Optimized Writes を使用するように設定されている RDS for MySQL データベースを、この機能をサポートしていない DB インスタンスクラスに移行した場合、RDS はそのデータベースの RDS Optimized Writes を自動的に無効にします。

RDS Optimized Writes を無効にすると、データベースは MySQL の二重書き込みバッファを使用します。

RDS for MySQL データベースが RDS Optimized Writes を使用しているかどうかを判断するには、データベースの `innodb_doublewrite` パラメータの現在の値を表示します。データベースが RDS Optimized Writes を使用している場合、このパラメータは `FALSE` (`0`) に設定されています。

RDS Optimized Writes の使用

RDS コンソール、AWS CLI、または RDS API を使用して RDS for MySQL データベースを作成すると、RDS Optimized Writes を有効にできます。RDS Optimized Writes は、データベース作成時に、次の条件に両方当てはまる場合、自動的に有効になります。

- RDS Optimized Writes をサポートする DB エンジンのバージョンと DB インスタンスクラスを指定します。

- RDS Optimized Writes は、RDS for MySQL バージョン 8.0.30 以降でサポートされています。RDS for MySQL のバージョンの詳細については、「Amazon RDS での MySQL のバージョン」を参照してください。
- RDS Optimized Writes は、次の DB インスタンスクラスを使用する RDS for MySQL データベースでサポートされています。
  - db.m8g
  - db.m8gd
  - db.m7i
  - db.m7g
  - db.m6g
  - db.m6gd
  - db.m6i
  - db.m5
  - db.m5d
  - db.r8g
  - db.r8gd
  - db.r7i
  - db.r7g
  - db.r6g
  - db.r6gd
  - db.r6i
  - db.r5
  - db.r5b
  - db.r5d
  - db.x2idn
  - db.x2iedn

  DB インスタンスクラスの詳細については、「the section called “DB インスタンスクラス”」を参照してください。

  DB インスタンスクラスの可用性は AWS リージョン によって異なります。DB インスタンスクラスが特定の AWS リージョン でサポートされているかどうかを判断するには、「the section

called “AWS リージョン での DB インスタンスクラスのサポートを決定する”」を参照してください。

データベースを RDS Optimized Writes をサポートする DB インスタンスクラスにアップグレードするには、ブルー/グリーンデプロイを作成できます。詳細については、「the section called “既存のデータベースで有効にする”」を参照してください。

- データベースに関連付けられたパラメータグループでは、`rds.optimized_writes` パラメータが `AUTO` に設定されます。デフォルトのパラメータグループでは、このパラメータは常に `AUTO` に設定されます。

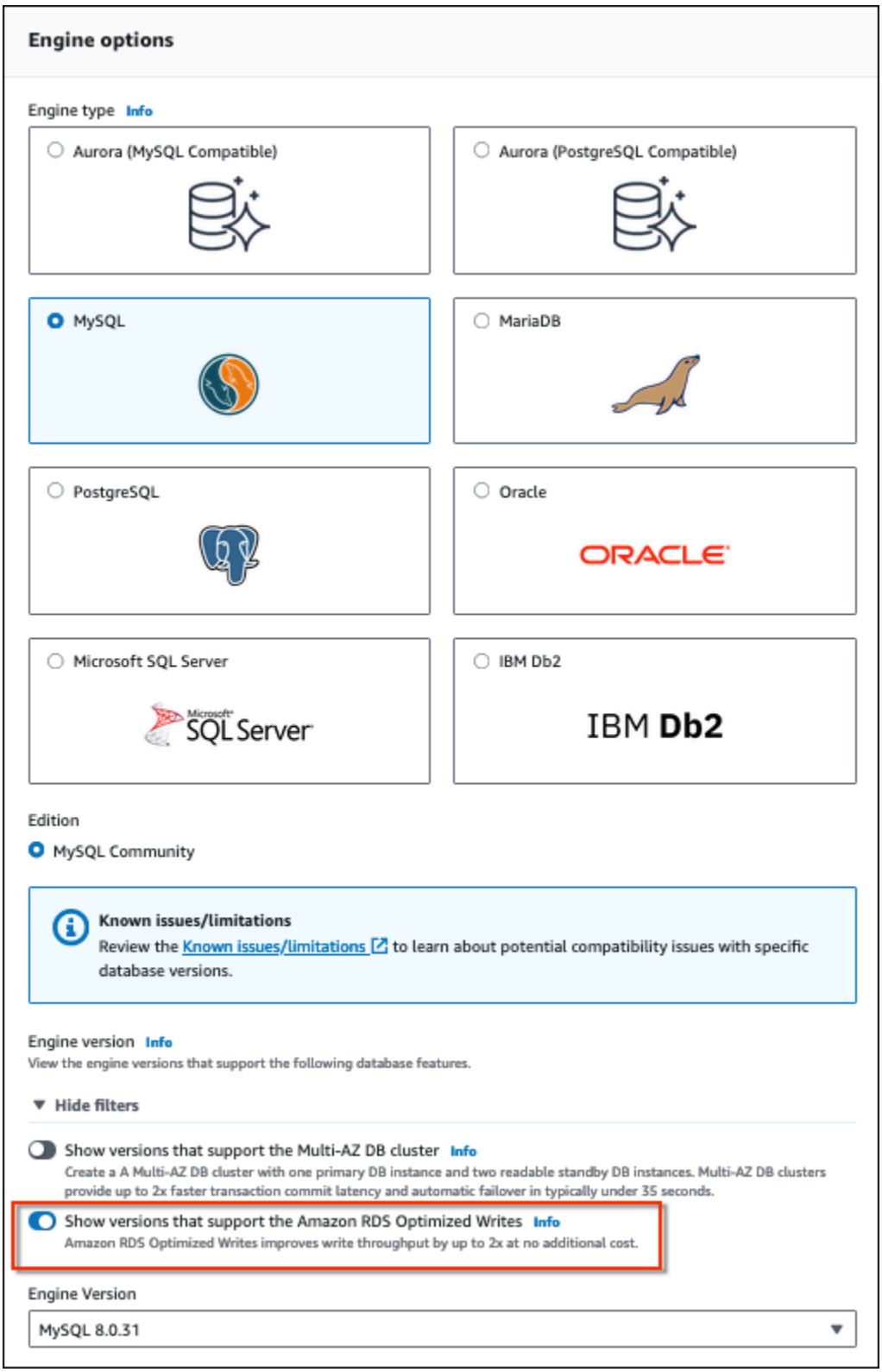
RDS Optimized Writes をサポートする DB エンジンバージョンと DB インスタンスクラスを使用し、この機能は使用しない場合は、データベースの作成時に、カスタムパラメータグループを指定します。このパラメータグループで、`rds.optimized_writes` パラメータを `OFF` に設定します。その後、データベースで RDS Optimized Writes を使用するには、パラメータを `AUTO` に設定して有効にすることができます。カスタムパラメータグループの作成とパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

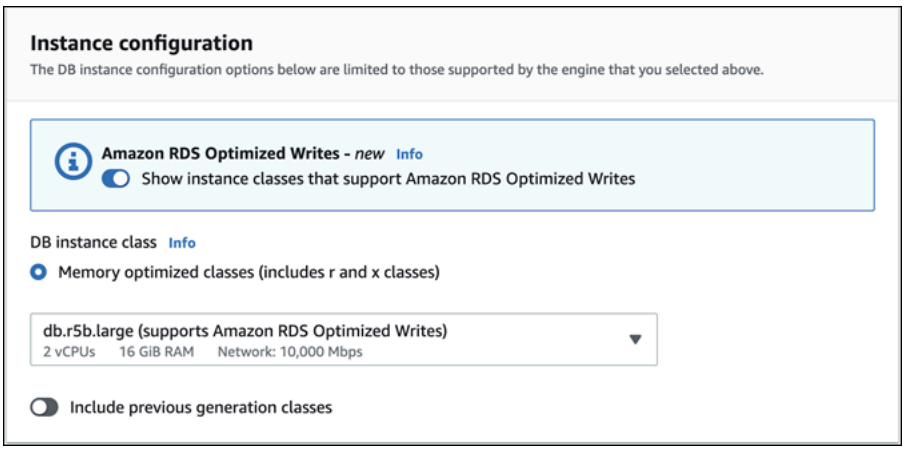
コンソール

RDS コンソールを使用して RDS for MySQL データベースを作成すると、RDS Optimized Writes をサポートする DB エンジンバージョンと DB インスタンスクラスをフィルタリングできます。フィルタリングを有効にすると、利用可能な DB エンジンのバージョンと DB インスタンスクラスから選択できるようになります。

RDS Optimized Writes をサポートする DB エンジンバージョンを選択するには、エンジンバージョンをサポートする RDS for MySQL DB のエンジンバージョンをフィルタリングして、バージョンを選択します。



[インスタンスの設定] セクションで、RDS Optimized Writes をサポートする DB インスタンスクラスをフィルタリングして、DB インスタンスクラスを選択します。



これらの選択を行った後、要件を満たす他の設定を選択し、コンソールで RDS for MySQL データベースの作成を完了します。

AWS CLI

AWS CLI を使用して DB インスタンスを作成するには、create-db-instance コマンドを実行します。`--engine-version` と `--db-instance-class` の値が RDS Optimized Writes をサポートしていることを確認してください。また、DB インスタンスに関連付けられたパラメータグループでは、`rds.optimized_writes` パラメータがAUTO に設定されていることを確認します。この例では、デフォルトパラメータグループを DB インスタンスに関連付けます。

Example RDS Optimized Writes を使用する DB インスタンスの作成

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mysql \
    --engine-version 8.0.30 \
    --db-instance-class db.r5b.large \
    --manage-master-user-password \
    --master-username admin \
    --allocated-storage 200
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine mysql ^
```

```
    --engine-version 8.0.30 ^
    --db-instance-class db.r5b.large ^
    --manage-master-user-password ^
    --master-username admin ^
    --allocated-storage 200
```

RDS API

CreateDBInstance オペレーションを使用して、DB インスタンスを作成できます。このオペレーションを使用する際は、EngineVersion と DBInstanceClass の値が RDS Optimized Writes をサポートしていることを確認してください。また、DB インスタンスに関連付けられたパラメータグループでは、rds.optimized_writes パラメータがAUTO に設定されていることを確認します。

既存のデータベースで RDS Optimized Writes を有効にする

RDS Optimized Writes を有効にするために既存の RDS for MySQL データベースを変更するには、サポート対象の DB エンジンバージョンと DB インスタンスクラスでデータベースを作成する必要があります。さらに、必要な基盤となるファイルシステム設定は、リリース前に作成されたデータベースの設定と互換性がないため、データベースは 2022 年 11 月 27 日に RDS Optimized Writes がリリースされた後に作成されたものでなければなりません。これらの条件が満たされている場合は、rds.optimized_writes パラメータを AUTO に設定して RDS Optimized Writes を有効にできます。

データベースが、サポートされているエンジンバージョン、インスタンスクラス、またはファイルシステム設定で作成されていない場合、RDS ブルー/グリーンデプロイを使用して、サポートされている設定に移行できます。ブルー/グリーンデプロイを作成して、以下の操作を行います。

- [グリーンデータベースで Optimized Writes を有効にする] を選択し、RDS Optimized Writes をサポートするエンジンバージョンと DB インスタンスクラスを指定します。サポートされているエンジンバージョンとインスタンスクラスのリストについては、「RDS Optimized Writes の使用」を参照してください。
- [ストレージ] で、[ストレージファイルシステム設定のアップグレード] を選択します。このオプションは、データベースを互換性のある基本ファイルシステム設定にアップグレードします。

ブルー/グリーンデプロイを作成すると、rds.optimized_writes パラメータが AUTO に設定されている場合、RDS Optimized Writes がグリーン環境で自動的に有効になります。その後、ブルー/グリーンデプロイを切り替えて、グリーン環境を新しい本番環境に昇格できます。

詳細については、「the section called “ブルー/グリーンデプロイの作成”」を参照してください。

RDS Optimized Writes についての制限事項

RDS for MySQL データベースをスナップショットから復元する場合、次の条件がすべて当てはまる場合のみ、データベースの RDS Optimized Writes を有効にできます。

- スナップショットが RDS Optimized Writes をサポートするデータベースから作成された。
- スナップショットは、RDS Optimized Writes のリリース後に作成されたデータベースから作成されました。
- スナップショットが RDS Optimized Writes をサポートするデータベースに復元された。
- 復元されたデータベースは、AUTO に設定された `rds.optimized_writes` パラメータに関連付けられています。

RDS for MySQL DB エンジンのアップグレード

新しいバージョンのデータベースエンジンが Amazon RDS でサポートされている場合は、DB インスタンスをその新しいバージョンにアップグレードできます。MySQL データベースのアップグレードには、メジャーバージョンのアップグレードとマイナーバージョンのアップグレードの 2 種類があります。

メジャーバージョンのアップグレード

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、DB インスタンスのメジャーバージョンアップグレードは手動で実行する必要があります。メジャーバージョンアップグレードをスタートするには、DB インスタンスを変更します。メジャーバージョンのアップグレードを行う前に、「RDS for MySQL のメジャーバージョンのアップグレード」の手順を実行することをお勧めします。

マルチ AZ DB インスタンスデプロイのメジャーバージョンアップグレードの場合、Amazon RDS はプライマリレプリカとスタンバイレプリカを同時にアップグレードします。アップグレードが完了するまで、DB インスタンスは使用できなくなります。マルチ AZ DB クラスターデプロイのメジャーバージョンアップグレードの場合、Amazon RDS はクラスターメンバーインスタンスを一度に 1 つずつアップグレードします。

> **Tip**
>
> ブルー/グリーンデプロイを使用することで、メジャーバージョンアップグレードに必要なダウンタイムを最小限に抑えることができます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

マイナーバージョンのアップグレード

マイナーバージョンアップグレードには、既存のアプリケーションとの下位互換性がある変更のみが含まれます。マイナーバージョンのアップグレードを手動でスタートするには、DB インスタンスを変更します。または、DB インスタンスの作成時または変更時に、[マイナーバージョン自動アップグレード] を有効にすることができます。これにより、Amazon RDS は、新しいバージョンがテストおよび承認されると、DB インスタンスを自動的にアップグレードします。アップグレードの実行については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

マルチ AZ DB クラスターのマイナーバージョンアップグレードを実行すると、Amazon RDS はリーダー DB インスタンスを一度に 1 つずつアップグレードします。その後、リーダー DB インスタンスの 1 つが新しいライター DB インスタンスに切り替わります。次に、Amazon RDS は、古いライターインスタンス (今ではリーダーインスタンス) をアップグレードします。

> **Note**
>
> マルチ AZ DB インスタンスデプロイのマイナーバージョンアップグレードのダウンタイムは、数分続く場合があります。マルチ AZ DB クラスターは、通常、マイナーバージョンアップグレードのダウンタイムを約 35 秒に短縮します。RDS Proxy と併用すると、ダウンタイムをさらに 1 秒以下に短縮できます。詳細については、「Amazon RDS Proxy」を参照してください。または、ProxySQL、PgBouncer、AWS Advanced JDBC Wrapper Driver などのオープンソースデータベースプロキシを使用することもできます。

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

ご使用の MySQL DB インスタンスでリードレプリカを使用している場合は、ソースインスタンスをアップグレードする前に、すべてのリードレプリカをアップグレードする必要があります。

トピック

- MySQL アップグレードに関する考慮事項
- 有効なアップグレードターゲットの検索
- MySQL バージョン番号
- RDS for MySQL の RDS バージョン番号
- RDS for MySQL のメジャーバージョンのアップグレード
- RDS for MySQL アップグレードのテスト
- MySQL DB インスタンスをアップグレードする
- RDS for MySQL のマイナーバージョンの自動アップグレード
- RDS for MySQL データベースのアップグレード時にリードレプリカを使用したダウンタイムの短縮
- イベントによる RDS for MySQL エンジンのアップグレードのモニタリング

MySQL アップグレードに関する考慮事項

Amazon RDS によってアップグレードプロセス中に 2 つ以上の DB スナップショットが作成されます。Amazon RDS は、アップグレードを変更する前に DB インスタンスのスナップショットを最大 2 つ作成します。アップグレードがデータベースに対して機能しない場合は、これらのスナップショットの 1 つを復元して、以前のバージョンを実行する DB インスタンスを作成できます。Amazon RDS は、アップグレードが完了すると、DB インスタンスのもう 1 つのスナップショットを作成します。Amazon RDS は、AWS Backup が DB インスタンスのバックアップを管理するかどうかにかかわらず、これらのスナップショットを作成します。

> **Note**
>
> DB インスタンスのバックアップ保持期間を 0 より大きく設定した場合にのみ、Amazon RDS は DB スナップショットを作成します。バックアップ保持期間を変更するには、「Amazon RDS DB インスタンスを変更する」を参照してください。

アップグレードが完了したら、データベースエンジンの前のバージョンに戻すことはできません。前のバージョンに戻す必要がある場合は、作成された初期の DB スナップショットを復元して、新しい DB インスタンスを作成します。

Amazon RDS でサポートされている新しいバージョンに DB インスタンスをアップグレードするタイミングを制御します。このレベルの管理によって、特定のデータベースのバージョンとの互換性を維持しながら、新しいバージョンを本稼働環境にデプロイする前にアプリケーションに対してテストできます。準備が整ったら、自分のスケジュールに最適なタイミングでバージョンアップグレードを実行できます。

ご使用の DB インスタンスでリードレプリカを使用している場合は、ソースインスタンスをアップグレードする前に、すべてのリードレプリカをアップグレードする必要があります。

有効なアップグレードターゲットの検索

AWS マネジメントコンソール を使用して DB インスタンスをアップグレードする場合、DB インスタンスの有効なアップグレードターゲットが表示されます。次の AWS CLI コマンドを実行して、DB インスタンスの有効なアップグレードターゲットを特定することもできます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
```

```
  --engine mysql \
  --engine-version version_number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine mysql ^
  --engine-version version_number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

例えば、MySQL バージョン 8.0.23 DB インスタンスの有効なアップグレードターゲットを特定するには、次の AWS CLI コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
  --engine mysql \
  --engine-version 8.0.28 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine mysql ^
  --engine-version 8.0.28 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

MySQL バージョン番号

RDS for MySQL データベースエンジンのバージョン番号付けシーケンスは、major.minor.patch.YYYYMMDD または major.minor.patch の形式です。例えば、8.0.33.R2.20231201 または 5.7.44 です。使用される形式は、MySQL エンジンのバージョンによって異なります。RDS 延長サポートのバージョン番号付けの詳細については、「Amazon RDS 延長サポートバージョンの命名規則」を参照してください。

major

メジャーバージョン番号は、8.0 など、バージョン番号の整数部分と 1 つ目の小数部分の両方です。メジャーバージョンのアップグレードでは、バージョン番号の主要な部分が大きくなります。例えば、5.7.44 から 8.0.33 へのアップグレードはメジャーバージョンアップグレードであり、5.7 と 8.0 はメジャーバージョン番号です。

minor

マイナーバージョン番号は、バージョン番号の 3 番目の部分です。例えば、8.0.33 の 33 です。

patch

パッチはバージョン番号の 4 番目の部分です。例えば、8.0.33.R2 の R2 です。RDS パッチバージョンには、リリース後にマイナーバージョンに追加された重要なバグ修正が含まれています。

YYYYMMDD

日付はバージョン番号の 5 番目の部分です。例えば、8.0.33.R2.20231201 の 20231201 です。RDS 日付バージョンは、リリース後にマイナーバージョンに追加された重要なセキュリティ修正を含むセキュリティパッチです。エンジンの動作を変更する可能性のある修正は含まれていません。

次の表に、RDS for MySQL バージョン 8.4 の命名スキームを示します。

8.4 マイナーバージョン	命名スキーム
≥ 3	新しい DB インスタンスは、major.minor.patch.YYMMDD を使用します。例えば、8.4.3.R2.20241201 です。 既存の DB インスタンスは、次のメジャーまたはマイナーバージョンアップグレードまで、8.4.3.R2 など、major.minor.patch を使用する場合があります。

次の表に、RDS for MySQL バージョン 8.0 の命名スキームを示します。

8.0 マイナーバージョン	命名スキーム
≥ 33	新しい DB インスタンスは、major.minor.patch.YYMMDD を使用します。例えば、8.0.33.R2.20231201 です。

8.0 マイナーバージョン	命名スキーム
	既存の DB インスタンスは、次のメジャーまたはマイナーバージョンアップグレードまで、8.0.33.R2 など、major.minor.patch を使用する場合があります。
< 33	既存の DB イスタンスは、major.minor.patch を使用します。例えば、8.0.32.R2 です。

次の表に、RDS for MySQL バージョン 5.7 の命名スキームを示します。

5.7 マイナーバージョン	命名スキーム
≥ 42	新しい DB インスタンスは、major.minor.patch.YYMMDD を使用します。例えば、5.7.42.R2.20231201 です。 既存の DB インスタンスは、次のメジャーまたはマイナーバージョンアップグレードまで、5.7.42.R2 など、major.minor.patch を使用する場合があります。

RDS for MySQL の RDS バージョン番号

RDS バージョン番号は *major*.*minor*.*patch* または *major*.*minor*.*patch*.*YYYYMMDD* 命名規則を使用します。Amazon RDS 延長サポートバージョンでは、*minor-RDS*.*YYYYMMDD* というマイナーバージョン命名スキームを使用します。

RDS パッチバージョンには、リリース後にマイナーバージョンに追加された重要なバグ修正が含まれています。RDS 日付バージョン (*YYYYMMDD*) はセキュリティパッチです。セキュリティパッチには、エンジンの動作を変更する可能性のある修正は含まれていません。RDS 延長サポートのバージョン番号付けの詳細については、「Amazon RDS 延長サポートバージョンの命名規則」を参照してください。

RDS for MySQL データベースの RDS バージョン番号は、次の SQL クエリで確認できます。

```
mysql> select mysql.rds_version();
```

ば、RDS for MySQL 8.0.34 データベースをクエリすると、次の出力が返されます。

```
+---------------------+
| mysql.rds_version() |
+---------------------+
| 8.0.34.R2.20231201  |
+---------------------+
1 row in set (0.01 sec)
```

RDS for MySQL のメジャーバージョンのアップグレード

Amazon RDS では現在、MySQL データベースエンジンで以下のインプレースのメジャーバージョンのアップグレードがサポートされています。

- MySQL 5.7 から MySQL 8.0 へ
- MySQL 8.0 から MySQL 8.4 へ

> **Note**
>
> 最新世代の DB インスタンスクラスと現行世代の DB インスタンスクラスでは、MySQL バージョン 5.7、8.0、および 8.4 の DB インスタンスのみ作成できます。
> 場合によっては、旧世代の DB インスタンスクラスで実行されている DB インスタンスを、それ以降のバージョンの MySQL エンジンを使用した DB インスタンスにアップグレードする必要があります。このような場合は、まず最新世代または現行世代の DB インスタンスクラスを使用するように DB インスタンスを変更します。変更したら、上位バージョンの MySQL データベースエンジンを使用するように DB インスタンスを変更してください。Amazon RDS DB インスタンスクラスについては、「DB インスタンスクラス」を参照してください。

トピック

- MySQL のメジャーバージョンのアップグレードの概要
- アップグレードの事前チェック
- アップグレードに失敗した後のロールバック

MySQL のメジャーバージョンのアップグレードの概要

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、メジャーバージョンのアップグレードは自動的に Amazon RDS に適用されません。手動で DB インスタンスを変更する必要があります。本稼働インスタンスへの適用前に、いずれのアップグレードも徹底的にテストすることをお勧めします。

メジャーバージョンアップグレードを実行するには、最初に OS アップデートを実行します。OS のアップデートが完了したら、それぞれのメジャーバージョン (例えば 5.7 から 8.0、8.0 から 8.4 など) にアップグレードします。RDS for MySQL マルチ AZ DB クラスターのアップグレードについては、「Amazon RDS のマルチ AZ DB クラスターのエンジンバージョンのアップグレード」を参照してください。2014 年 4 月 24 日以前に作成された MySQL DB は、更新が適用されるまで使用可能な OS の更新を表示します。OS 更新の詳細については、「DB インスタンスへの更新の適用」を参照してください。

MySQL のメジャーバージョンアップグレード中、必要に応じて Amazon RDS によって MySQL バイナリ `mysql_upgrade` が実行され、テーブルがアップグレードされます。また、メジャーバージョンアップグレード中に Amazon RDS によって `slow_log` および `general_log` テーブルが空にされます。ログ情報を保持するには、メジャーバージョンアップグレードの前にログファイルの内容を保存します。

MySQL のメジャーバージョンアップグレードは、通常約 10 分で完了します。一部のアップグレードでは、DB インスタンスクラスのサイズのため、またはインスタンスが「Amazon RDS のベストプラクティス」の運用ガイドラインに従っていないため、この時間が長くなることがあります。Amazon RDS コンソールから DB インスタンスをアップグレードする場合、アップグレードが完了すると、DB インスタンスのステータスが表示されます。AWS Command Line Interface (AWS CLI) を使用してアップグレードする場合は、describe-db-instances コマンドを使用し、`Status` 値を確認します。

アップグレードの事前チェック

Amazon RDS は、アップグレード前に事前チェックを実行して非互換性を確認します。これらの非互換性は、アップグレード先の MySQL バージョンによって異なります。

事前チェックには、MySQL に含まれている証明書と Amazon RDS チーム用によって特別に作成された証明書が含まれます。MySQL が提供する事前確認については、Upgrade Checker Utility」を参照してください。

DB インスタンスがアップグレードで停止される前に事前チェックが実行されます。つまり、実行時にダウンタイムが発生することはありません。事前チェックで非互換性が見つかった場合、DB

インスタンスが停止する前に、Amazon RDS により自動的にアップグレードがキャンセルされます。Amazon RDS では、非互換性のためのイベントも生成されます。Amazon RDS イベントの詳細については、「Amazon RDS イベント通知の操作」を参照してください。

Amazon RDS は、ログファイル `PrePatchCompatibility.log` に各非互換性に関する詳細情報を記録します。ほとんどの場合、ログエントリには非互換性を修正するための MySQL のドキュメントへのリンクが含まれています。ログの表示の詳細については、「データベースログファイルの表示とリスト化」を参照してください。

事前チェックの性質上、データベース内のオブジェクトが分析されます。この分析によりリソースが消費され、アップグレードが完了するまでの時間が長くなります。

トピック

- MySQL 8.0 から 8.4 へのアップグレードの事前確認
- MySQL 5.7 から 8.0 へのアップグレードの事前確認

MySQL 8.0 から 8.4 へのアップグレードの事前確認

MySQL 8.4 には MySQL 8.0 との非互換性がいくつかあります。このような非互換性が原因で MySQL 8.0 から MySQL 8.4 へのアップグレード中に問題が生じる可能性があります。そのため、アップグレードを成功させるには、データベースで何らかの準備が必要になる場合があります。このような非互換性の一般的なリストを以下に示します。

- テーブルで古いデータ型や関数を使用してはいけません。
- トリガーの definer が欠落しているか、空である、またはトリガーに無効な作成コンテキストが含まれていてはいけません。
- キーワードや予約語に違反してはいけません。MySQL 8.4 では、以前に予約されていなかったキーワードもあります。

  詳細については、MySQL ドキュメントの「キーワードと予約語」を参照してください。
- MySQL 8.0 の `mysql` システムデータベースに、MySQL 8.4 データディクショナリで使用されるテーブルと同じ名前のテーブルがあってはいけません。
- `sql_mode` システム可変設定で、古い SQL モードを定義してはいけません。
- 長さが 255 文字または 1,020 バイトを超える `ENUM` または `SET` 列要素をそれぞれ持つテーブルまたはストアドプロシージャがあってはいけません。
- MySQL 8.4 でサポートされていない機能を MySQL 8.0 のインストールで使用することはできません。

詳細については、MySQL ドキュメントの「MySQL 8.4 で削除された機能」を参照してください。

- 64 文字を超える外部キーの制約名があってはいけません。
- Unicode サポートを改善するには、次の情報を確認してください。
  - utf8mb3 文字セットを使用するオブジェクトを、utf8mb4 文字セットを使用するように変換することを検討してください。utf8mb3 文字セットは廃止されました。
  - 文字セット参照に utf8mb4 を utf8 の代わりに使用することを検討してください。現在 utf8 は utf8mb3 文字セットの別名であるためです。可能であれば、まず utf8 を utf8mb4 に変更してから、データベースをアップグレードします。
  - 古いクライアントは utf8mb3 に対して不明な文字セットエラーを受け取る可能性があるため、データベースをアップグレードする前にデータベースクライアントをアップグレードしてください。

  詳細については、MySQL ドキュメントの「utf8mb3 文字セット (3 バイトの UTF-8 Unicode エンコード)」を参照してください。

  文字セットを変更するには、データベースのバックアップ、復元、レプリケーションを手動で実行できます。または、Amazon RDS のブルー/グリーンデプロイを使用できます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

MySQL 8.0 から 8.4 へのアップグレードをスタートすると、Amazon RDS では、これらの非互換性を検出するために自動的に事前チェックが実行されます。MySQL 8.4 へのアップグレードについては、MySQL ドキュメントの「MySQL のアップグレード」を参照してください。

これらの事前確認は必須です。スキップすることはできません。事前チェックには次の利点があります。

- アップグレード中の計画外のダウンタイムを回避することができます。
- 非互換性がある場合、Amazon RDS でアップグレードすることはできません。詳細を示すログが出力されます。ログを使用して、非互換性を排除することにより、データベースを MySQL 8.4 にアップグレードする準備を行うことができます。非互換性の排除の詳細については、MySQL ドキュメントの「アップグレードのためのインストールの準備」を参照してください。

MySQL 5.7 から 8.0 へのアップグレードの事前確認

MySQL 8.0 には MySQL 5.7 との非互換性がいくつかあります。このような非互換性が原因で MySQL 5.7 から MySQL 8.0 へのアップグレード中に問題が生じる可能性があります。そのため、アップグレードを成功させるには、データベースで何らかの準備が必要になる場合があります。このような非互換性の一般的なリストを以下に示します。

- テーブルで古いデータ型や関数を使用してはいけません。
- 孤立した *.frm ファイルがあってはいけません。
- トリガーの definer が欠落しているか、空である、またはトリガーに無効な作成コンテキストが含まれていてはいけません。
- ネイティブのパーティショニングをサポートしていないストレージエンジンを使用するパーティショニングされたテーブルがあってはいけません。
- キーワードや予約語に違反してはいけません。MySQL 8.0 では、以前に予約されていなかったキーワードもあります。

  詳細については、MySQL ドキュメントの「キーワードと予約語」を参照してください。
- MySQL 5.7 の `mysql` システムデータベースに、MySQL 8.0 データディクショナリで使用されるテーブルと同じ名前のテーブルがあってはいけません。
- `sql_mode` システム可変設定で、古い SQL モードを定義してはいけません。
- 長さが 255 文字または 1,020 バイトを超える `ENUM` または `SET` 列要素をそれぞれ持つテーブルまたはストアドプロシージャがあってはいけません。
- MySQL 8.0.13 以降にアップグレードする前に、共有 InnoDB テーブルスペースに存在するテーブルパーティションがあってはいけません。
- `ASC` 句に `DESC` または `GROUP BY` 修飾子を使用する、MySQL 8.0.12 以前のクエリおよびストアドプログラム定義があってはなりません。
- MySQL 8.0でサポートされていない機能を MySQL 5.7 のインストールで使用することはできません。

  詳細については、MySQL ドキュメントの「MySQL 8.0 で削除された機能」を参照してください。
- 64 文字を超える外部キーの制約名があってはいけません。
- Unicode サポートを改善するには、次の情報を確認してください。
  - `utf8mb3` 文字セットを使用するオブジェクトを、`utf8mb4` 文字セットを使用するように変換することを検討してください。`utf8mb3` 文字セットは廃止されました。

- 文字セット参照に utf8mb4 を utf8 の代わりに使用することを検討してください。現在 utf8 は utf8mb3 文字セットの別名であるためです。可能であれば、まず utf8 を utf8mb4 に変更してから、データベースをアップグレードします。
- 古いクライアントは utf8mb3 に対して不明な文字セットエラーを受け取る可能性があるため、データベースをアップグレードする前にデータベースクライアントをアップグレードしてください。

詳細については、MySQL ドキュメントの「utf8mb3 文字セット (3 バイトの UTF-8 Unicode エンコード)」を参照してください。

文字セットを変更するには、データベースのバックアップ、復元、レプリケーションを手動で実行できます。または、Amazon RDS のブルー/グリーンデプロイを使用できます。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

MySQL 5.7 から 8.0 へのアップグレードをスタートすると、Amazon RDS では、これらの非互換性を検出するために自動的に事前チェックが実行されます。MySQL 8.0 へのアップグレードについては、MySQL ドキュメントの「MySQL のアップグレード」を参照してください。

これらの事前確認は必須です。スキップすることはできません。事前チェックには次の利点があります。

- アップグレード中の計画外のダウンタイムを回避することができます。
- 非互換性がある場合、Amazon RDS でアップグレードすることはできません。詳細を示すログが出力されます。ログを使用して、非互換性を排除することにより、データベースを MySQL 8.0 にアップグレードする準備を行うことができます。非互換性の排除の詳細については、MySQL ドキュメントの「アップグレードのためのインストールの準備」と「MySQL 8.0? へのアップグレード」を参照してください。MySQL Server ブログの「知っておくべきこと」を参照してください。

アップグレードに失敗した後のロールバック

DB インスタンスを MySQL バージョン 5.7 から MySQL バージョン 8.0、または MySQL バージョン 8.0 から 8.4 にアップグレードすると、アップグレードが失敗することがあります。特に、事前チェックでキャプチャされなかった非互換性がデータディクショナリに含まれていると、失敗する可能性があります。この場合、データベースは新しい MySQL 8.0 または 8.4 バージョンで正常に起動できません。この時点で、Amazon RDS は、アップグレードに対して実行された変更をロールバックします。ロールバック後、MySQL DB インスタンスは元のバージョンを実行しています。

- MySQL バージョン 8.0 (MySQL 8.4 からロールバックする場合)
- MySQL バージョン 5.7 (MySQL 8.0 からロールバックする場合)

アップグレードが失敗してロールバックされると、Amazon RDS は、イベント ID RDS-EVENT-0188 のイベントを生成します。

通常、DB インスタンス内のデータベースとターゲットの MySQL バージョン間のメタデータに非互換性があるため、アップグレードは失敗します。アップグレードが失敗した場合、`upgradeFailure.log` ファイルでこのような互換性に関する詳細を確認できます。アップグレードを再試行する前に、非互換性を解決してください。

アップグレードの試行とロールバックが失敗すると、DB インスタンスが再起動されます。保留中のパラメータの変更は、再起動時に適用され、ロールバック後も保持されます。

MySQL 8.0 へのアップグレードの詳細については、MySQL ドキュメントの次のトピックを参照してください。

- アップグレードのためのインストールの準備
- MySQL 8.0 にアップグレードしますか? こちらをお読みください..。

MySQL 8.4 へのアップグレードの詳細については、MySQL ドキュメントの「アップグレードのためのインストールの準備」を参照してください。

RDS for MySQL アップグレードのテスト

DB インスタンスでメジャーバージョンアップグレードを実行する前に、新しいバージョンとの互換性についてデータベースを徹底的にテストしてください。また、データベースにアクセスするすべてのアプリケーションの新しいバージョンとの互換性についても徹底的にテストします。以下の手順を実行することをお勧めします。

メジャーバージョンのアップグレードをテストするには

1. データベースエンジンの新しいバージョンについてアップグレードドキュメントを参照して、データベースやアプリケーションに影響を与える可能性のある互換性の問題があるかどうかを確認します。

   - MySQL 5.7 での変更
   - MySQL 8.0 での変更

- MySQL 8.4 での変更

2. DB インスタンスがカスタム DB パラメータグループに関連付けられている場合は、新しいメジャーバージョンと互換性のある既存の設定で新しい DB パラメータグループを作成します。その新しい DB パラメータグループをテストインスタンスのアップグレード時に指定することで、アップグレードのテストでインスタンスが正常に機能することを確認できます。DB パラメータグループの作成の詳細については、「Amazon RDS のパラメータグループ」を参照してください。
3. アップグレードする DB インスタンスの DB スナップショットを作成します。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
4. DB スナップショットを復元して、新しいテスト DB インスタンスを作成します。詳細については、「DB インスタンスへの復元」を参照してください。
5. この新しいテスト DB インスタンスを変更して新しいバージョンにアップグレードするには、次に説明するいずれかの方法を使用します。ステップ 2 で新しいパラメータグループを作成した場合は、そのパラメータグループを指定します。
6. アップグレードしたインスタンスによって使用されるストレージを評価して、アップグレードに追加のストレージが必要かどうかを判断します。
7. データベースとアプリケーションが新しいバージョンで正常に動作することが確認されるまで、アップグレードした DB インスタンスに対する品質保証テストを必要な回数だけ実行します。ステップ 1 で特定した互換性の問題の影響を評価するための新しいテストを実行します。すべてのストアドプロシージャと関数をテストします。アプリケーションのテストバージョンを、アップグレードした DB インスタンスに割り振ります。
8. すべてのテストに合格したら、本稼働 DB インスタンスのアップグレードを実行します。すべてが正常に動作していることを確認するまでは、DB インスタンスへの書き込みオペレーションは許可しないことをお勧めします。

MySQL DB インスタンスをアップグレードする

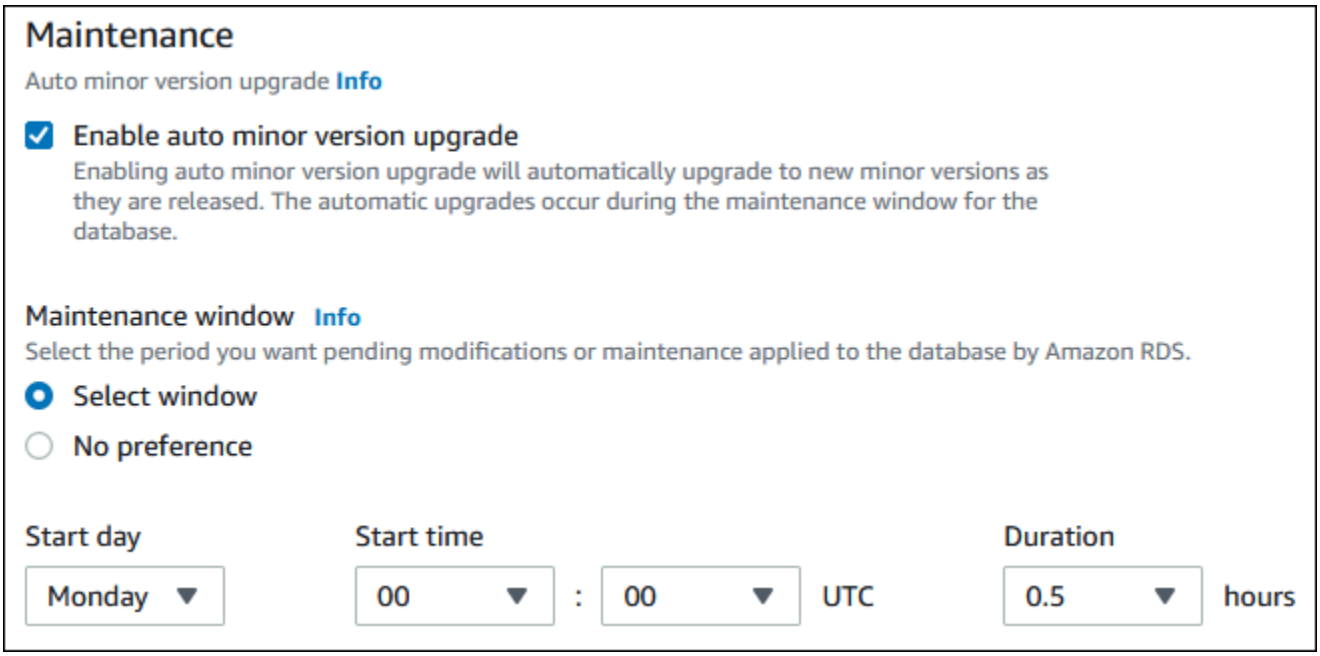
MySQL DB インスタンスの手動または自動アップグレードについては、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

RDS for MySQL のマイナーバージョンの自動アップグレード

DB インスタンスの作成または変更時に次の設定を指定すると、DB インスタンスを自動的にアップグレードできます。

- [マイナーバージョンの自動アップグレード] の設定は有効です。
- バックアップ保持期間の設定は 0 より大きいです。

AWS マネジメントコンソールの場合、これらの設定はAAdditional configuration (追加設定) の下にあります。下図は、Auto minor version upgrade (自動マイナーバージョンアップグレード) 設定を示しています。



これらの設定の詳細については、「DB インスタンスの設定」をご参照ください。

一部の AWS リージョン の RDS for MySQL メジャーバージョンでは、RDS によって 1 つのマイナーバージョンが自動アップグレードバージョンとして指定されます。Amazon RDS でマイナーバージョンのテストと承認が完了すると、メンテナンスウィンドウの間にマイナーバージョンアップグレードが自動的に行われます。RDS では、新しくリリースされたマイナーバージョンが自動アップグレードバージョンとして自動的に設定されることはありません。RDS によって新しい自動アップグレードバージョンが指定される前に、以下のような複数の基準が考慮されます。

- 既知のセキュリティの問題
- MySQL コミュニティバージョンのバグ
- マイナーバージョンがリリースされてからのフリート全体の安定性

次の AWS CLI コマンドを実行して、特定の AWS リージョン で指定された MySQL マイナーバージョンの現在の自動マイナーアップグレードターゲットバージョンを確認できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version minor_version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version minor_version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

例えば、次の AWS CLI コマンドにより、米国東部 (オハイオ)AWS リージョン (us-east-2) の MySQL マイナーバージョン 8.0.11 の自動マイナーアップグレードターゲットを特定できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine mysql \
--engine-version 8.0.11 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine mysql ^
--engine-version 8.0.11 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

以下のような出力が生成されます。

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  8.0.15         |
|  False       |  8.0.16         |
|  False       |  8.0.17         |
|  False       |  8.0.19         |
|  False       |  8.0.20         |
|  False       |  8.0.21         |
|  True        |  8.0.23         |
|  False       |  8.0.25         |
+--------------+-----------------+
```

この例では、MySQL バージョン 8.0.23 のAutoUpgrade値はTrueです。したがって、自動マイナーアップグレードターゲットは MySQL バージョン 8.0.23 であり、出力でハイライトされています。

MySQL DB インスタンスは、以下の基準を満たしている場合、メンテナンスウィンドウの間に自動的にアップグレードされます。

- [マイナーバージョンの自動アップグレード] の設定は有効です。
- バックアップ保持期間の設定は 0 より大きいです。
- DB インスタンスでは、現在の自動アップグレードマイナーバージョン未満の DB エンジンのマイナーバージョンが実行されています。

詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。

RDS for MySQL データベースのアップグレード時にリードレプリカを使用したダウンタイムの短縮

ほとんどの場合、MySQL DB インスタンスをアップグレードする際のダウンタイムを減らすには、ブルー/グリーンデプロイが最適なオプションです。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。

ブルー/グリーンデプロイを使用できず、MySQL DB インスタンスが現在本稼働アプリケーションで使用されている場合は、次の手順を使用して DB インスタンスのデータベースバージョンをアップグレードできます。この手順により、アプリケーションの停止時間を短縮できます。

リードレプリカを使用して、ほとんどのメンテナンスステップを事前に実行し、実際の停止中に必要な変更を最小限に抑えることができます。この方法で、既存の DB インスタンスを変更せずに、新しい DB インスタンスのテストおよび準備ができます。

次の手順は、MySQL バージョン 5.7 から MySQL バージョン 8.0 へのアップグレードの例です。他のメジャーバージョンへのアップグレードに、この同じ一般的なステップを使用できます。他のメジャーバージョンへのアップグレードに、この同じ一般的なステップを使用できます。

> **Note**
>
> MySQL バージョン 5.7 から MySQL バージョン 8.0、または MySQL バージョン 8.0 から MySQL バージョン 8.4 にアップグレードする場合、アップグレードを実行する前に事前確認を完了してください。詳細については、MySQL 5.7 から 8.0 へのアップグレードの事前確認およびMySQL 8.0 から 8.4 へのアップグレードの事前確認を参照してください。

DB インスタンスの使用中に MySQL データベースをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール https://console.aws.amazon.com/rds/ を開きます。
2. MySQL 5.7 DB インスタンスのリードレプリカを作成します。このプロセスにより、データベースのアップグレード可能なコピーが作成されます。DB インスタンスの他のリードレプリカも存在する場合があります。

   a. コンソールで [データベース] を選択し、アップグレードする DB インスタンスを選択します。
   b. [アクション] で [リードレプリカの作成] を選択します。
   c. リードレプリカの [DB インスタンス識別子] の値を指定し、[DB インスタンスクラス] などの設定が MySQL 5.7 DB インスタンスと一致することを確認します。
   d. [Create read replica] を選択します。
3. (オプション) リードレプリカが作成され、ステータスが [利用可能] になったら、リードレプリカをマルチ AZ 配置に変換し、バックアップを有効にします。

   デフォルトでは、バックアップが無効な状態でリードレプリカが作成されます。リードレプリカは最終的に本番 DB インスタンスになるため、ベストプラクティスは、マルチ AZ 配置を設定してバックアップを有効にすることです。

   a. コンソールで [データベース] を選択し、作成したばかりのリードレプリカを選択します。

b. Modify を選択します。

c. マルチ AZ 配置で、[スタンバイインスタンスの作成] を選択します。

d. [Backup Retention Period] (バックアップ保持期間) として、0 以外の正の値 (3 日など) を選択し、[Continue] (続行) を選択します。

e. [変更のスケジューリング] で、[すぐに適用] を選択します。

f. [DB インスタンスの変更] を選択します。

4. リードレプリカのステータスが [利用可能] になったら、リードレプリカを MySQL 8.0 にアップグレードします。

a. コンソールで [データベース] を選択し、作成したばかりのリードレプリカを選択します。

b. Modify を選択します。

c. [DB エンジンのバージョン] で、アップグレードする MySQL 8.0 のバージョンを選択し、[続行] を選択します。

d. [変更のスケジューリング] で、[すぐに適用] を選択します。

e. [Modify DB instance] を選択してアップグレードをスタートします。

5. アップグレードが完了し、[ステータス] が [利用可能] になったら、アップグレードしたリードレプリカがソース MySQL 5.7 DB インスタンスで最新であることを確認します。確認するには、リードレプリカに接続して、`SHOW REPLICA STATUS` コマンドを実行します。`Seconds_Behind_Master` フィールドが `0` である場合、レプリカは更新されています。

> **Note**
>
> MySQL の旧バージョンは、`SHOW REPLICA STATUS` ではなく `SHOW SLAVE STATUS` を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

6. (オプション) リードレプリカのリードレプリカを作成します。

DB インスタンスをスタンドアロン DB インスタンスに昇格した後、リードレプリカを追加する場合、すぐにリードレプリカを作成できます。

a. コンソールで [データベース] を選択し、アップグレードしたばかりのリードレプリカを選択します。

b. [アクション] で [リードレプリカの作成] を選択します。

c. リードレプリカの [DB インスタンス識別子] の値を指定し、[DB インスタンスクラス] などの設定が MySQL 5.7 DB インスタンスと一致することを確認します。

d. [Create read replica] を選択します。

7. (オプション) リードレプリカのカスタム DB パラメータグループを設定します。

   DB インスタンスをスタンドアロン DB インスタンスに昇格した後、カスタムパラメータグループを使用する場合、すぐにDB パラメータグループを作成して、リードレプリカに関連付けられます。

   a. MySQL 8.0 のカスタム DB パラメータグループを作成します。手順については、「Amazon RDS での DB パラメータグループの作成」を参照してください。

   b. 作成したばかりの DB パラメータグループで変更するパラメータを変更します。手順については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

   c. コンソールで [データベース] を選択し、リードレプリカを選択します。

   d. Modify を選択します。

   e. [DB パラメータグループ] で、作成したばかりの MySQL 8.0 DB パラメータグループを選択し、[続行] を選択します。

   f. [変更のスケジューリング] で、[すぐに適用] を選択します。

   g. [Modify DB instance] を選択してアップグレードをスタートします。

8. MySQL 8.0 リードレプリカをスタンドアロン DB インスタンスにします。

   Important

   MySQL 8.0 のリードレプリカをスタンドアロン DB インスタンスに昇格すると、MySQL 5.7 DB インスタンスのレプリカではなくなります。MySQL 8.0 のリードレプリカの昇格は、ソース MySQL 5.7 DB インスタンスが読み取り専用モードで、すべての書き込みオペレーションが停止されているメンテナンスウィンドウ中に実行することをお勧めします。昇格が完了したら、アップグレードした MySQL 8.0 DB インスタンスに書き込みオペレーションを割り振ることで、書き込みオペレーションが失われないようにします。
   また、MySQL 8.0 のリードレプリカの昇格前に、必要なすべてのデータ定義言語 (DDL) のオペレーションを MySQL 8.0 のリードレプリカで実行することをお勧めします。例えば、インデックスの作成があります。これにより、昇格後の MySQL 8.0 のリードレ

プリカのパフォーマンスへの悪影響を避けることができます。リードレプリカを昇格させるには、次の手順に従います。

a. コンソールで [データベース] を選択し、アップグレードしたばかりのリードレプリカを選択します。
b. [アクション] で、[Promote (昇格)] を選択します。
c. [はい] を選択して、リードレプリカインスタンスの自動バックアップを有効にします。詳細については、「バックアップの概要」を参照してください。
d. [Continue] (続行) をクリックします。
e. [Promote Read Replica] を選択します。

9. これで、MySQL データベースのアップグレードバージョンが作成されました。この時点で、アプリケーションを新しい MySQL 8.0 DB インスタンスに割り振ることができます。

イベントによる RDS for MySQL エンジンのアップグレードのモニタリング

RDS for MySQL データベースのエンジンバージョンをアップグレードすると、Amazon RDS はプロセスの各フェーズで特定のイベントを発行します。アップグレードの進行状況を追跡するには、これらのイベントを表示またはサブスクライブします。

RDS イベントの詳細については、「Amazon RDS イベントのモニタリング」を参照してください。

エンジンのアップグレード中に発生する特定の Amazon RDS イベントの詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

MySQL DB スナップショットエンジンバージョンのアップグレード

Amazon RDS を使用すると、MySQL DB インスタンスのストレージボリュームの DB スナップショットを作成できます。DB スナップショットを作成するとき、スナップショットは DB インスタンスが使用するエンジンバージョンに基づきます。DB スナップショットのエンジンバージョンをアップグレードできます。

RDS for MySQL の場合、バージョン 5.7 のスナップショットをバージョン 8.0 に、またはバージョン 8.0 のスナップショットをバージョン 8.4 にアップグレードできます。暗号化されている DB スナップショットまたは暗号化されていない DB スナップショットをアップグレードできます。

RDS for MySQL DB スナップショットで使用可能なエンジンバージョンを表示するには、次の AWS CLI 例を使用します。

```
aws rds describe-db-engine-versions --engine mysql --include-all --engine-
version example-engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text
```

スナップショットの結果が表示されない場合は、エンジンのバージョンが非推奨である可能性があります。エンジンのバージョンが非推奨の場合は、最新のメジャーバージョンのアップグレードターゲットにアップグレードするか、そのバージョンで利用可能な他のアップグレードターゲットにアップグレードすることをお勧めします。詳細については、「RDS for MySQL でサポートされていないエンジンバージョンの DB スナップショットのアップグレードオプション」を参照してください。

新しいエンジンバージョンにアップグレードした DB スナップショットをリストアしたら、アップグレードに問題がないか必ずテストしてください。主なバージョンアップグレードの詳細は、the section called “MySQL DB エンジンのアップグレード” を参照してください。DB スナップショットをリストアする方法については the section called “DB インスタンスへの復元” をご覧ください。

> **Note**
>
> 自動バックアッププロセス中に作成された自動 DB スナップショットをアップグレードすることはできません。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して DB スナップショットをアップグレードできます。

Console

AWS マネジメントコンソールを使用して DB スナップショットエンジンのバージョンをアップグレードするには、次の手順を使用します。

DB スナップショットをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、「Snapshots」 を選択します。
3. アップグレードしたいスナップショットを選択します。
4. [アクション] は、[スナップショットのアップグレード] を選択します。[スナップショットのアップグレード] ページが表示されます。
5. アップグレードする [新しいエンジンバージョン] を選択します。
6. [変更を保存] を選択してスナップショットをアップグレードします。

   アップグレード中は、この DB スナップショットに関するスナップショット操作が、すべて無効になります。また、DB スナップショットのステータスは、利用可能からアップグレード中に変わり、プロセスが完了するとアクティブに変わります。スナップショットの破損問題で DB スナップショットをアップグレードできない場合、ステータスは使用不可になります。この状態からスナップショットを復元することはできません。

   > **Note**
   >
   > DB スナップショットアップグレードが失敗した場合、スナップショットは元の状態にロールバックします。

AWS CLI

DB スナップショットを新しいバージョンのデータベースエンジンにアップグレードするには、AWS CLI の modify-db-snapshot コマンドを実行します。

オプション

- `--db-snapshot-identifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

- `--engine-version` – DB スナップショットをこのエンジンバージョンにアップグレードする。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot \

    --db-snapshot-identifier my_db_snapshot \
    --engine-version new_version
```

Windows の場合:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

Amazon RDS API

DB スナップショットを新しいデータベースエンジンバージョンにアップグレードするには、RDS API の ModifyDBSnapshot オペレーションを呼び出します。

パラメータ

- `DBSnapshotIdentifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。
- `EngineVersion` – DB スナップショットをこのエンジンバージョンにアップグレードする。

RDS for MySQL でサポートされていないエンジンバージョンの DB スナップショットのアップグレードオプション

次の表は、RDS for MySQL DB スナップショットでサポートされていないエンジンバージョンからアップグレードできるエンジンバージョンを示しています。

> **Note**
>
> 選択したエンジンバージョンにアップグレードするには、DB スナップショットを複数回アップグレードする必要がある場合があります。

DB スナップショットエンジンのバージョン	アップグレード可能なエンジンバージョン
5.5.8	5.5.62、5.6.51
5.5.12	5.5.62、5.6.51
5.5.20	5.5.62、5.6.51
5.5.23	5.5.62、5.6.51
5.5.25a	5.5.62、5.6.51
5.5.27	5.5.62、5.6.51
5.5.31	5.5.62、5.6.51
5.5.33	5.5.62、5.6.51
5.5.37	5.5.62、5.6.51
5.5.38	5.5.62、5.6.51
5.5.40	5.5.62、5.6.51
5.5.40a	5.5.62、5.6.51
5.5.40b	5.5.62、5.6.51
5.5.41	5.5.62、5.6.51
5.5.42	5.5.62、5.6.51
5.5.59	5.5.62、5.6.51

DB スナップショットエンジンのバージョン	アップグレード可能なエンジンバージョン
5.6.12	5.6.51、5.7.44
5.6.13	5.6.51、5.7.44
5.6.17	5.6.51、5.7.44
5.6.19	5.6.51、5.7.44
5.6.19a	5.6.51、5.7.44
5.6.19b	5.6.51、5.7.44
5.6.21	5.6.51、5.7.44
5.6.21b	5.6.51、5.7.44
5.6.22	5.6.51、5.7.44
5.6.23	5.6.51、5.7.44
5.6.27	5.6.51、5.7.44
5.6.27a	5.6.51、5.7.44
5.7.10	5.7.44、5.7.44-rds.20240408、5.7.44-rds.20240529、5.7.44-rds.20250103、5.7.44-rds.20250213、8.0.32、8.0.33、8.0.34、8.0.35、8.0.36、8.0.37、8.0.39、8.0.40、8.0.41
5.7.11	5.7.44、5.7.44-rds.20240408、5.7.44-rds.20240529、5.7.44-rds.20250103、5.7.44-rds.20250213、8.0.32、8.0.33、8.0.34、8.0.35、8.0.36、8.0.37、8.0.39、8.0.40、8.0.41
5.7.12	5.7.44、5.7.44-rds.20240408、5.7.44-rds.20240529、5.7.44-rds.20250103、5.7.44-rds.20250213、8.0.32、8.0.33、8.0.34、8.0.35、8.0.36、8.0.37、8.0.39、8.0.40、8.0.41

Amazon RDS for MySQL DB インスタンスへのデータのインポート

RDS for MySQL DB インスタンスへのデータのインポートには、さまざまな手法を使用できます。最適のアプローチは、いくつかの要因によって異なります。

- データのソース
- データ量
- 1 回限りのインポートか継続的なインポートか
- ダウンタイムの長さ

データとともにアプリケーションを移行する場合は、ダウンタイムの長さは重要な考慮事項です。

次の表は、RDS for MySQL DB インスタンスにデータをインポートする手法の一覧です。

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
オンプレミスまたは Amazon EC2 の既存の MySQL データベース	すべて	1 回	ある程度	オンプレミスデータベースのバックアップを作成して Amazon S3 に保存し、次に MySQL を実行する新しい Amazon RDS DB インスタンスにバックアップファイルを復元できます。	Amazon RDS for MySQL DB インスタンスへのバックアップの復元
オンプレミスまたは Amazon	いずれか	継続的	最小限	レプリケーション元として既存の MySQL データベースを使用してレプリケーションを設定します。	外部のソースインスタンス

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
EC2 の既存の MySQL データベース					を使用したバイナリログファイル位置のレプリケーションの設定 ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存のデータベース	すべて	1 回または進行中	最小限	AWS Database Migration Service を使用してダウンタイムを最小限にしてデータベースを移行し、多くのデータベース DB エンジンで継続的なレプリケーションを継続します。	「AWS Database Migration Service ユーザーガイド」の「AWS Database Migration Service とは?」および「MySQL 互換データベースの AWS DMS のターゲットとしての使用」

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存の MySQL DB インスタンス	すべて	1 回または進行中	最小限	継続的なレプリケーション用のリードレプリカを作成します。新しい DB インスタンスの 1 回限りの作成用のリードレプリカを昇格させます。	DB インスタンスのリードレプリカの操作
既存の MySQL データベース	スモール	1 回	ある程度	コマンドラインユーティリティを使用して、MySQL DB インスタンスに直接データをコピーします。	外部の MySQL データベースから Amazon RDS for MySQL DB インスタンスにデータをインポートする

ソース	データ量	1 回または進行中	アプリケーションのダウンタイム	手法	詳細情報
既存のデータベースに保存されないデータ	ミディアム	1 回	ある程度	フラットファイルを作成し、MySQL `LOAD DATA LOCAL INFILE` ステートメントを使用してインポートします。	任意のソースから Amazon RDS for MySQL DB インスタンスにデータをインポートする

> **Note**
>
> `mysql` システムデータベースには、DB インスタンスへのログインとデータへのアクセスに必要な認証と認可の情報が含まれています。DB インスタンスにある `mysql` データベースのテーブル、データ、または他のコンテンツを削除、変更、名前変更、または切り取りを行うとエラーが発生し、DB インスタンスとデータにアクセスできなくなる場合があります。この場合は、AWS CLI の restore-db-instance-from-db-snapshot コマンドを使用して、スナップショットから DB インスタンスを復元できます。AWS CLI の restore-db-instance-to-point-in-time コマンドを使用して、DB インスタンスを復元できます。

MySQL のデータインポートに関する考慮事項

以下のコンテンツには、MySQL へのデータのロードに関連する技術情報が含まれています。このコンテンツは、MySQL サーバーアーキテクチャに精通しているユーザーを対象としています。

バイナリログ記録

バイナリログ記録を有効にすると、データロードのパフォーマンスが低下し、ログを無効にした場合と比較して最大 4 倍の追加のディスク領域が必要になります。データのロードに使用されるトランザクションサイズは、システムのパフォーマンスとディスク容量のニーズに直接影響します。トランザクションが大きいほど、より多くのリソースが必要になります。

トランザクションサイズ

トランザクションサイズは、MySQL データロードの以下の側面に影響します。

- リソース消費
- ディスクスペースの使用率
- プロセスの再開
- 復旧時間
- 入力形式 (フラットファイルまたは SQL)

このセクションでは、トランザクションサイズがバイナリログ作成に与える影響について説明し、大量のデータロード中にバイナリログ作成を無効にするケースを作成します。Amazon RDS 自動バックアップ保持期間を設定することで、バイナリログ作成を有効または無効にすることができます。値が 0 以外の場合はバイナリログ作成が有効になり、0 の場合は無効になります。詳細については、「バックアップの保存期間」を参照してください。

このセクションでは、大規模なトランザクションが InnoDB に与える影響と、トランザクションサイズを小さく保つことが重要な理由についても説明します。

小さいトランザクション

トランザクションが小さい場合、バイナリログ作成により、データのロードに必要なディスク書き込み数が 2 倍になります。この影響は、他のデータベースセッションのパフォーマンスを著しく低下させ、データのロードに必要な時間を長くします。劣化の程度は、以下の要因に一部依存します。

- アップロードレート
- ロード中に発生するその他のデータベースアクティビティ
- Amazon RDS DB インスタンスの容量

バイナリログは、バックアップおよび削除されるまでにロードされたデータの量とほぼ同じディスク容量を消費します。Amazon RDS は、バイナリログを頻繁にバックアップして削除することで、この容量を最小限に抑えます。

大きいトランザクション

大規模なトランザクションの場合、バイナリログによって、次の理由により IOPS とディスク使用量が 3 倍になります。

- バイナリログキャッシュは、トランザクションデータを一時的にディスクに保存します。
- このキャッシュはトランザクションのサイズに応じて大きくなり、ディスク領域を消費します。
- トランザクション (コミットまたはロールバック) が完了すると、システムはキャッシュをバイナリログにコピーします。

このプロセスでは、データのコピーが 3 つ作成されます。

- 元のデータ
- ディスク上のキャッシュ
- 最後のバイナリログエントリ

書き込みオペレーションごとに追加の IO が発生し、パフォーマンスにさらに影響を及ぼします。

このため、バイナリログ記録には、ログ記録を無効にした場合と比較して 3 倍のディスク容量が必要です。例えば、1 つのトランザクションとして 10 GiB のデータをロードすると、次の 3 つのコピーが作成されます。

- テーブルデータに 10 GiB
- バイナリログキャッシュに 10 GiB
- バイナリログファイルに 10 GiB

必要な一時ディスク容量の合計は 30 GiB です。

ディスク容量に関する重要な考慮事項:

- キャッシュファイルは、セッションが終了するか、新しいトランザクションによって別のキャッシュが作成されるまで保持されます。

- バイナリログはバックアップされるまで保持され、長期間にわたって 20 GiB (キャッシュとログ) が保持される可能性があります。

`LOAD DATA LOCAL INFILE` を使用してデータをロードする場合、ロード前に行われたバックアップからデータベースを復元する必要がある場合に備えて、データ復旧によって 4 番目のコピーが作成されます。復旧時に、MySQL はバイナリログからフラットファイルにデータを抽出します。その後、MySQL は `LOAD DATA LOCAL INFILE` を実行します。上記の例の場合、この復旧には合計 40 GiB、つまりテーブル、キャッシュ、ログ、ローカルファイルごとに 10 GiB の一時ディスク領域が必要です。少なくとも 40 GiB の空きディスク容量がないと、復旧は失敗します。

大規模なデータロードの最適化

大量のデータをロードする場合は、バイナリログを無効にして、オーバーヘッドとディスク領域の要件を削減します。バイナリログ記録を無効にするには、バックアップ保持期間を 0 に設定します。ロードが完了したら、バックアップ保持期間を適切なゼロ以外の値に復元します。詳細については、設定テーブルの「Amazon RDS DB インスタンスを変更する」および「バックアップ保持期間」を参照してください。

> **Note**
>
> DB インスタンスがリードレプリカのソース DB インスタンスである場合、バックアップ保持期間を 0 に設定することはできません。

データをロードする前に、DB スナップショットを作成することをお勧めします。詳細については、「手動バックアップの管理」を参照してください。

InnoDB

undo ログと復旧のオプションに関する以下の情報は、InnoDB トランザクションを小さくしてデータベースのパフォーマンスを最適化することをサポートしています。

InnoDB の undo ログについて

undo ログは、トランザクションのロールバックを有効にし、マルチバージョン同時実行制御 (MVCC) をサポートするログ記録メカニズムです。

MySQL 5.7 以前のバージョンでは、undo ログは InnoDB システムテーブルスペース (通常は ibdata1) に保存され、パージスレッドによって削除されるまで保持されます。その結果、大規模な

データロードトランザクションによってシステムテーブルスペースが非常に大きくなり、データベースを再作成しない限り再利用できないディスク領域が消費される可能性があります。

すべての MySQL バージョンでは、最も古いアクティブなトランザクションがコミットまたはロールバックされるまで、パージスレッドは undo ログの削除を待機する必要があります。データベースが、ロード中に他のトランザクションを処理する場合、そのトランザクションがコミットされ、他のトランザクションが MVCC の undo ログを必要としない場合でも、それらの undo ログも蓄積され、削除できません。この場合、読み取り専用トランザクションを含むすべてのトランザクションが遅くなります。このスローダウンは、すべてのトランザクションが、ロードトランザクションだけでなく、任意のトランザクションが変更するすべての行にアクセスするために発生します。実際には、トランザクションは、長時間実行されるロードトランザクションによって undo ログのクリーンアップ中に消去されなかった undo ログをスキャンする必要があります。これは、変更された行にアクセスするオペレーションのパフォーマンスに影響します。

InnoDB トランザクション復旧オプション

InnoDB はコミットオペレーションを最適化しますが、大規模なトランザクションのロールバックは遅くなります。復旧を高速化するには、ポイントインタイムリカバリを実行するか、DB スナップショットを復元します。詳細については、「ポイントインタイムリカバリ」および「DB インスタンスへの復元」を参照してください。

データインポート形式

MySQL は、フラットファイルと SQL の 2 つのデータインポート形式をサポートしています。各形式に関する情報を確認して、ニーズに最適なオプションを決定します。

フラットファイル

小規模なトランザクションの場合は、`LOAD DATA LOCAL INFILE` を使用してフラットファイルをロードします。このデータインポート形式は、SQL を使用する場合に比べて次のようなメリットがあります。

- ネットワークトラフィックの削減
- データ転送コストの削減
- データベース処理オーバーヘッドの減少
- より高速な処理

`LOAD DATA LOCAL INFILE` はフラットファイル全体を 1 つのトランザクションとしてロードします。個々のファイルのサイズを小さく保つと、次の利点があります。

- 再開機能 - どのファイルがロードされたかを追跡できます。ロード中に問題が発生した場合、中断したところから再開できます。一部のデータを Amazon RDS に再送信する必要がある場合でも、ファイルが小さいと、再送信される量は最小限ですみます。
- 同時データロード – 単一ファイルのロードに十分な IOPS とネットワーク帯域幅がある場合、同時にロードすれば時間を節約できます。
- ロード速度制御 – データロードが他のプロセスに悪影響を及ぼす場合は、ファイル間の間隔を増やすことでロード速度を制御できます。

大規模なトランザクションでは、`LOAD DATA LOCAL INFILE` を使用してデータをインポートするメリットが減ります。大量のデータを小さなファイルに分割できない場合は、SQL の使用を検討してください。

SQL

SQL にはフラットファイルに比べて、トランザクションサイズを簡単に小さく保つことができるという大きなメリットがあります。ただし、SQL のロードにはフラットファイルよりもかなり時間がかかる場合があります。また、障害が発生した後、どこから再開するかを判断するのが難しい場合があります。mysqldump ファイルを再開することはできません。mysqldump ファイルのロード中にエラーが発生した場合、ロードを再開するにはファイルを変更するか置き換える必要があります。または、障害の原因を修正した後、ロード前の時点に復元してファイルを再開する必要があります。詳細については、「ポイントインタイムリカバリ」を参照してください。

データベースチェックポイントに Amazon RDS DB スナップショットを使用する

バイナリログ記録を使用せずに、数時間または数日など長期間にわたってデータをロードする場合は、DB スナップショットを使用して、データの安全性を確保するために定期的なチェックポイントを提供します。各 DB スナップショットは、システム障害やデータ破損イベント時の復旧ポイントとして機能するデータベースインスタンスの一貫したコピーを作成します。DB スナップショットは高速であるため、頻繁なチェックポイント作成によるロードパフォーマンスへの影響は最小限に抑えられます。データベースの耐久性や復旧機能に影響を与えることなく、以前の DB スナップショットを削除できます。DB スナップショットの詳細については、「手動バックアップの管理」を参照してください。

データベースのロード時間を短縮する

以下の項目は、ロード時間を短縮するための追加のヒントです。

- MySQL データベースにデータをロードする前に、すべてのセカンダリインデックスを作成します。他のデータベースシステムとは異なり、MySQL はセカンダリインデックスを追加または変更

するときにテーブル全体を再構築します。このプロセスでは、インデックスが変更された新しいテーブルが作成され、すべてのデータがコピーされ、元のテーブルが削除されます。

- プライマリキーの順序でデータをロードします。InnoDB テーブルの場合、これによりロード時間が 75%～80% 短縮され、データファイルサイズが 50% 短縮されます。
- `foreign_key_checks` を `0` に設定して、外部キーの制約を無効にします。これは多くの場合、`LOAD DATA LOCAL INFILE` を使用してロードされたフラットファイルで必要となります。どのロードでも、外部キーチェックを無効にすると、データのロードが高速化されます。ロードが完了したら、`foreign_key_checks` を `1` に設定して制約を再度有効にし、データを検証します。
- リソース制限に近づかない限り、データを同時にロードします。複数のテーブルセグメント間で同時ロードを有効にするには、必要に応じてパーティションテーブルを使用します。
- SQL 実行のオーバーヘッドを減らすには、複数の `INSERT` ステートメントを単一の複数値 `INSERT` オペレーションに結合します。`mysqldump` は、この最適化を自動的に実装します。
- `innodb_flush_log_at_trx_commit` を `0` に設定することで、InnoDB ログ IO オペレーションを減らします。ロードが完了したら、`innodb_flush_log_at_trx_commit` を `1` に戻します。

> **Warning**
>
> `innodb_flush_log_at_trx_commit` を `0` に設定すると、InnoDB はコミットごとではなく毎秒ログをフラッシュします。この設定はパフォーマンスを向上させますが、システム障害時にトランザクションが失われるリスクがあります。

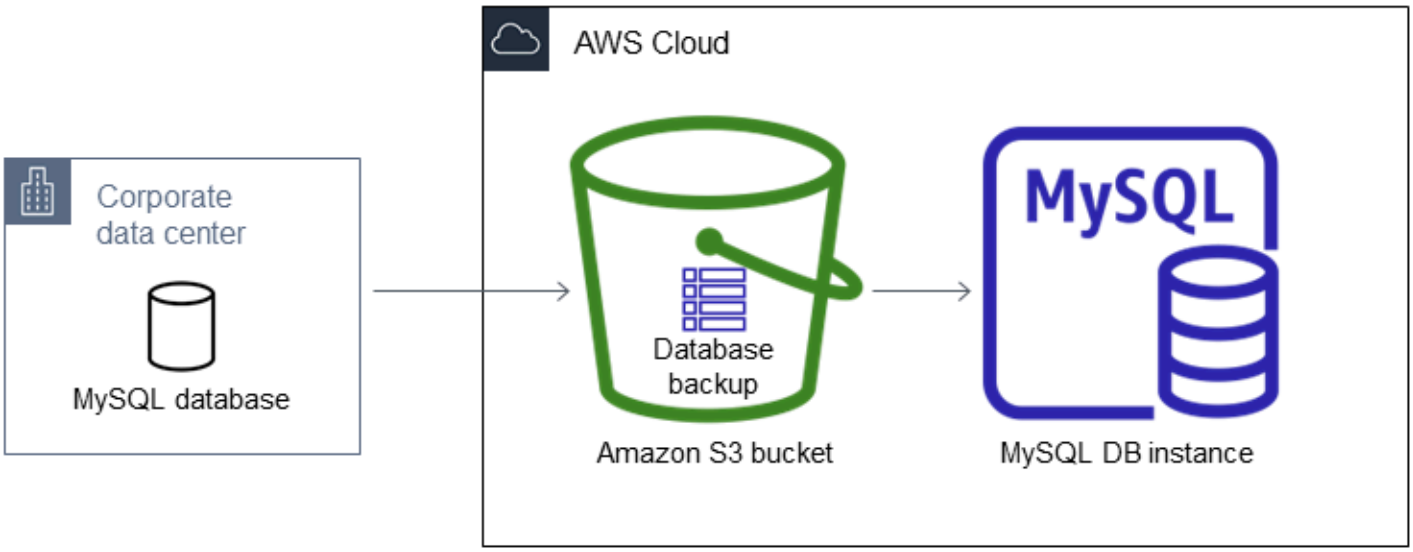
- リードレプリカがない DB インスタンスにデータをロードする場合は、`sync_binlog` を `0` に設定します。ロードが完了したら、`sync_binlog parameter` を `1` に戻します。
- DB インスタンスをマルチ AZ 配置に変換する前に、シングル AZ DB インスタンスにデータをロードします。DB インスタンスが既にマルチ AZ 配置を使用している場合は、データロードのためにシングル AZ 配置に切り替えることはお勧めしません。これで実現できる改善はわずかだけです。

Amazon RDS for MySQL DB インスタンスへのバックアップの復元

Amazon RDS では、バックアップファイルを使用して MySQL データベースをインポートできます。データベースのバックアップを作成して、そのバックアップファイルを Amazon S3 に保存し、その後、MySQL を実行する新しい Amazon RDS DB インスタンスにバックアップファイルを復元できます。Amazon RDS は、すべての AWS リージョンで Amazon S3 からのバックアップファイルのインポートをサポートしています。

このセクションで説明するシナリオでは、オンプレミスデータベースのバックアップを復元します。データベースがアクセス可能であれば、この手法は Amazon EC2 や他のクラウドサービスなど、他の場所にあるデータベースにも使用できます。

次の図は、サポートされるシナリオを示しています。



バックアップファイルの作成、コピー、復元時にオンプレミスデータベースをオフラインにできる場合、データベースを Amazon RDS にインポートするには、バックアップファイルの使用をお勧めします。データベースをオフラインにできない場合は、次のいずれかの方法を使用できます。

- バイナリログ – このトピックで説明されているように、まず Amazon S3 から Amazon RDS にバックアップファイルをインポートします。次に、バイナリログ (binlog) レプリケーションを使用してデータベースを更新します。詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。
- AWS Database Migration Service – AWS Database Migration Service を使用して、データベースを Amazon RDS に移行します。詳細については、「What is AWS Database Migration Service?」を参照してください。

Amazon S3 から Amazon RDS にバックアップファイルをインポートするための設定の概要

Amazon S3 から Amazon RDS にバックアップファイルをインポートするには、次のコンポーネントが必要です。

- バックアップファイルを保存する Amazon S3 バケット。

Amazon S3 バケットが既にある場合は、そのバケットを使用できます。Amazon S3 バケットがない場合は、新しいバケットを作成します。詳細については、「バケットの作成」を参照してください。

- Percona XtraBackup で作成されたオンプレミスデータベースのバックアップ。

  詳細については、「データベースバックアップの作成」を参照してください。
- Amazon RDS から S3 バケットへのアクセスを許可する AWS Identity and Access Management (IAM) ロール。

  IAM ロールが既にある場合は、そのロールを使用して信頼ポリシーとアクセス許可ポリシーをアタッチできます。詳細については、「IAM ロールの手動作成」を参照してください。

  IAM ロールがない場合は、次の 2 つのオプションがあります。
  - 新しい IAM ロールを手動で作成することができます。詳細については、「IAM ロールの手動作成」を参照してください。
  - Amazon RDS で新しい IAM ロールを作成することを選択できます。Amazon RDS で新しい IAM ロールを作成する場合は、「Amazon S3 から新しい MySQL DB インスタンスにデータをインポートする」セクションの AWS マネジメントコンソール を使用する手順に従ってください。

データベースバックアップの作成

Percona XtraBackup ソフトウェアを使用してバックアップを作成します。Percona XtraBackup の最新バージョンの使用をお勧めします。Percona XtraBackup は、Percona ウェブサイトのソフトウェアのダウンロードからインストールできます。

> **Warning**
>
> データベースのバックアップを作成する際、XtraBackup では認証情報をxtrabackup_info ファイルに保存することがあります。xtrabackup_info ファイルの `tool_command` 設定に機密情報が含まれていないことを確認してください。

使用する Percona XtraBackup のバージョンは、バックアップする MySQL のバージョンによって異なります。

- MySQL 8.4 – Percona XtraBackup バージョン 8.4 を使用します。

- MySQL 8.0 – Percona XtraBackup バージョン 8.0 を使用します。

> **Note**
>
> Percona XtraBackup 8.0.12 以降のバージョンでは、MySQL 8.0 のすべてのバージョンの移行がサポートされています。RDS for MySQL 8.0.32 以降に移行する場合は、Percona XtraBackup 8.0.12 以降を使用する必要があります。

- MySQL 5.7 – Percona XtraBackup バージョン 2.4 を使用します。

Percona XtraBackup を使用して MySQL データベースファイルの完全なバックアップを作成できます。または、Percona XtraBackup を使用して MySQL データベースファイルをバックアップ済みである場合は、既存の完全および増分バックアップディレクトリおよびファイルをアップロードできます。

Percona XtraBackup を使用したデータベースのバックアップの詳細については、Percona ウェブサイトの「Percona XtraBackup - Documentation」を参照してください。

Percona XtraBackup での完全バックアップの作成

Amazon RDS が Amazon S3 から復元できる、MySQL データベースファイルの完全なバックアップを作成するには、Percona XtraBackup ユーティリティ (`xtrabackup`) を使用します。

例えば、次のコマンドは MySQL データベースのバックアップを作成し、ファイルを `/on-premises/s3-restore/backup` フォルダに保存します。

```
xtrabackup --backup --user=myuser --password=password --target-dir=/on-premises/s3-restore/backup
```

バックアップを 1 つのファイル (必要に応じて後で複数のファイルに分割できます) に圧縮する場合、MySQL のバージョンに応じて、以下のいずれかの形式でバックアップを保存できます。

- Gzip (.gz) – MySQL 5.7 以前のバージョン用
- tar (.tar) – MySQL 5.7 以前のバージョン用
- Percona xbstream (.xbstream) – すべての MySQL バージョン用

> **Note**
>
> Percona XtraBackUp 8.0 以降では、圧縮には Percona xbstream のみがサポートされています。

MySQL 5.7 以前のバージョン

次のコマンドでは、複数の Gzip ファイルに分割された MySQL データベースのバックアップを作成します。値を自分の情報に置き換えます。

```
xtrabackup --backup --user=my_user --password=password --stream=tar \
   --target-dir=/on-premises/s3-restore/backup | gzip - | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.tar.gz
```

MySQL 5.7 以前のバージョン

次のコマンドでは、複数の tar ファイルに分割された MySQL データベースのバックアップを作成します。値を自分の情報に置き換えます。

```
xtrabackup --backup --user=my_user --password=password --stream=tar \
   --target-dir=/on-premises/s3-restore/backup | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.tar
```

MySQL のすべてのバージョン

例えば、次のコマンドでは、複数の xbstream ファイルに分割された MySQL データベースのバックアップを作成します。値を自分の情報に置き換えます。

```
xtrabackup --backup --user=myuser --password=password --stream=xbstream \
   --target-dir=/on-premises/s3-restore/backup | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.xbstream
```

> **Note**
>
> 次のエラーが表示される場合は、コマンド内でファイル形式が混在している可能性があります。
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Percona XtraBackup での増分バックアップの使用

Percona XtraBackup を使用して MySQL データベースファイルの完全および増分バックアップを作成済みである場合は、完全バックアップを作成して Amazon S3 にアップロードする必要はありません。代わりに、既存のバックアップのディレクトリおよびファイルを Amazon S3 バケットにコピーして、時間を節約できます。Percona XtraBackup を使用した増分バックアップの作成の詳細については、「Create an incremental backup」を参照してください。

既存の完全および増分バックアップファイルを Amazon S3 バケットにコピーするときは、ベースディレクトリのコンテンツを再帰的にコピーする必要があります。これらのコンテンツには、完全バックアップと、すべての増分バックアップディレクトリおよびファイルが含まれます。このコピーには、Amazon S3 バケットのディレクトリ構造を維持する必要があります。Amazon RDS では、すべてのファイルとディレクトリが反復処理されます。Amazon RDS では、増分バックアップごとに含まれる `xtrabackup-checkpoints` ファイルを使用してベースディレクトリが識別され、ログシーケンス番号 (LSN) の範囲に従って増分バックアップが実行されます。

Percona XtraBackup のバックアップに関する考慮事項

Amazon RDS では、ファイル名に基づいてバックアップファイルを使用します。ファイル形式に基づいた適切なファイル拡張子でバックアップファイルの名前を付けます。例えば、Percona xbstream 形式を使用して保存されるファイルには `.xbstream` を使用します。

Amazon RDS では、アルファベット順および通常の数値順にバックアップファイルを使用します。バックアップファイルが適切な順序で書き込まれ、名前が付けられるようにするには、`split` コマンドを発行する際に必ず `xtrabackup` オプションを使用します。

Amazon RDS では、Percona XtraBackup を使用して作成された部分バックアップがサポートされていません。データベースのソースファイルをバックアップするときに部分バックアップを作成するには、次のオプションは使用できません。

- `--tables`
- `--tables-exclude`
- `--tables-file`
- `--databases`
- `--databases-exclude`
- `--databases-file`

IAM ロールの手動作成

IAM ロールがない場合、手動で新しいロールを作成できます。ただし、AWS マネジメントコンソール を使用してデータベースを復元する場合、RDS にこの新しい IAM ロールを作成させることをお勧めします。Amazon RDS でこのロールを作成するには、「Amazon S3 から新しい MySQL DB インスタンスにデータをインポートする」セクションの手順に従います。

Amazon S3 からデータベースをインポートするための新しい IAM ロールを手動で作成するには、Amazon RDS から Amazon S3 バケットにアクセス許可を委任するロールを作成します。IAM ロールを作成したら、信頼ポリシーとアクセス権限ポリシーをアタッチします。バックアップファイルを Amazon S3 からインポートするには、以下の例のような信頼ポリシーとアクセス許可ポリシーを使用します。ロールの作成の詳細については、「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

信頼ポリシーとアクセス権限ポリシーを使用するには、Amazon リソースネーム (ARN) を指定する必要があります。ARN 形式の詳細については、「Amazon リソースネーム (ARN) と AWS のサービスの名前空間」を参照してください。

Example Amazon S3 からインポートするための信頼ポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleForBackup",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Example Amazon S3 からインポートするためのアクセス許可ポリシー — IAM ユーザーのアクセス許可

次の例で、*iam_user_id* を独自の値に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3AccessRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/S3Access"
        }
    ]
}
```

Example Amazon S3 からインポートするためのアクセス許可ポリシー — ロールのアクセス許可

次の例で、*amzn-s3-demo-bucket* と *prefix* を独自の値に置き換えます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
        "Effect": "Allow",
        "Action":
```

```
            [
                "s3:GetObject"
            ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "kms:Decrypt"
            ],
        "Resource": [
            "arn:aws:kms:us-east-1:111122223333:key/key_id*"
            ]
        }
    ]
}
```

> **Note**
>
> ファイル名プレフィックスを含める場合は、プレフィックスの後にアスタリスク (*) を含めます。プレフィックスを指定しない場合は、アスタリスクのみを指定します。

Amazon S3 から新しい MySQL DB インスタンスにデータをインポートする

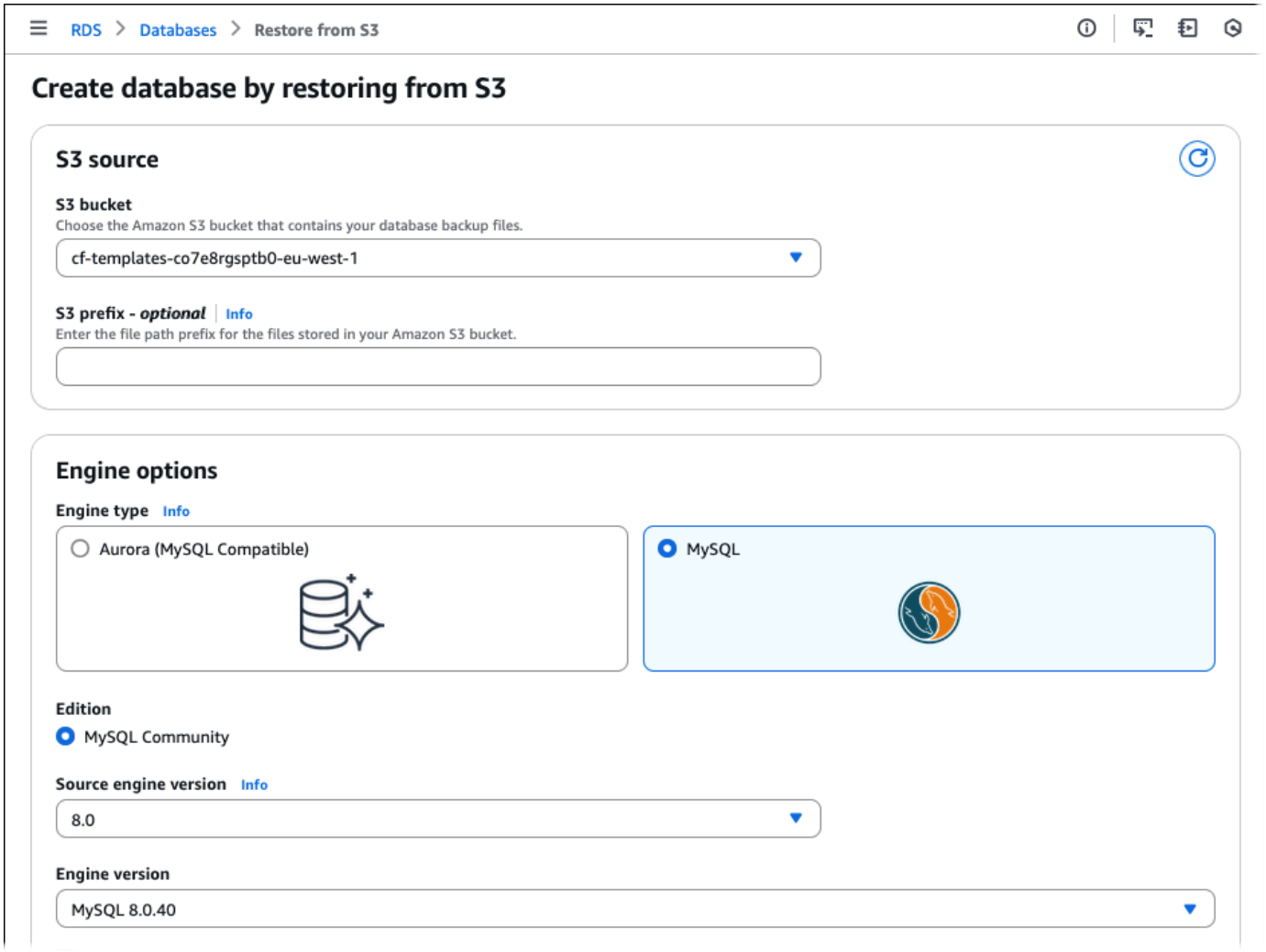
AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、Amazon S3 から新しい MySQL DB インスタンスにデータをインポートできます。

コンソール

Amazon S3 から新しい MySQL DB インスタンスにデータをインポートするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスを作成する AWS リージョンを選択します。データベースバックアップを含む Amazon S3 バケットと同じ AWS リージョン を選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [S3 から復元する] を選択します。

[S3 から復元してデータベースを作成する] ページが表示されます。



5. S3 ソースで:

   a. バックアップを含む S3 バケットを選択します。

   b. (オプション) [S3 プレフィックス] で、Amazon S3 バケットに保存されているファイルのファイルパスのプレフィックスを入力します。

      プレフィックスを指定しない場合、Amazon RDS は S3 バケットのルートフォルダにあるすべてのファイルとフォルダを使用して DB インスタンスを作成します。プレフィックスを指定すると、Amazon RDS はファイルのパスが指定されたプレフィックスで始まる S3 バケットのファイルとフォルダを使用して DB インスタンスを作成します。

      例えば、複数のバックアップファイルのセットを S3 のサブフォルダ (backup) 内に保存し、各セットは独自のディレクトリ (gzip_backup1、gzip_backup2 など) 内にあるとし

ます。この場合、gzip_backup1 フォルダのファイルから復元するには、プレフィックス backups/gzip_backup1 を指定します。

6. [エンジンオプション]で

   a. [エンジンタイプ]は[MySQL]を選択します。

   b. [ソースエンジンバージョン]では、ソースデータベースの MySQL メジャーバージョンを選択します。

   c. [エンジンバージョン] では、AWS リージョン の MySQL メジャーバージョンのデフォルトのマイナーバージョンを選択します。

      AWS マネジメントコンソール では、デフォルトのマイナーバージョンのみを使用できます。インポートが完了した後 DB インスタンスをアップグレードできます。

7. IAM ロールでは、Amazon RDS が Amazon S3 バケットにアクセスできるように、必要な信頼ポリシーとアクセス許可ポリシーを持つ IAM ロールを作成または選択します。次のいずれかのアクションを実行します。

   - (推奨) [新しいロールの作成] を選択し、IAM ロール名を入力します。このオプションを使用すると、Amazon RDS は信頼ポリシーとアクセス許可ポリシーを使用してロールを自動的に作成します。
   - 既存の IAM ロールを選択します。このロールが the section called “IAM ロールの手動作成” のすべての基準を満たしていることを確認します。

   > **Note**
   >
   > アカウントでロールマネージャーが有効になっている場合、Amazon RDS は自動的にロールをアタッチします。また、ここで説明するロールオプションは [カスタマイズ] オプションに置き換えられます。別のロールを使用するには、[カスタマイズ] を選択します。詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

8. DB インスタンス情報を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。

   > **Note**
   >
   > 復元を成功させるためには、新しい DB インスタンスに十分なストレージを必ず割り当ててください。

将来の拡張を自動的に実行するには、[追加のストレージ設定] で、[ストレージの自動スケーリングを有効にする] を選択します。

9. 必要に応じて、追加の設定を選択します。
10. [データベースの作成] を選択します。

AWS CLI

AWS CLI を使用して Amazon S3 から新しい MySQL DB インスタンスにデータをインポートするには、次のオプションを使用して restore-db-instance-from-s3 コマンドを実行します。各設定の詳細については、「DB インスタンスの設定」を参照してください。

> **Note**
>
> 復元を成功させるためには、新しい DB インスタンスに十分なストレージを必ず割り当ててください。
> ストレージのオートスケーリングを有効にし、将来の拡張を自動的に実行するには、`--max-allocated-storage` オプションを使用します。

- `--allocated-storage`
- `--db-instance-identifier`
- `--db-instance-class`
- `--engine`
- `--master-username`
- `--manage-master-user-password`
- `--s3-bucket-name`
- `--s3-ingestion-role-arn`
- `--s3-prefix`
- `--source-engine`
- `--source-engine-version`

Example

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-s3 \
    --allocated-storage 250 \
    --db-instance-identifier my_identifier \
    --db-instance-class db.m5.large \
    --engine mysql \
    --master-username admin \
    --manage-master-user-password \
    --s3-bucket-name amzn-s3-demo-bucket \
    --s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename \
    --s3-prefix bucket_prefix \
    --source-engine mysql \
    --source-engine-version 8.0.32 \
    --max-allocated-storage 1000
```

Windows の場合:

```
aws rds restore-db-instance-from-s3 ^
    --allocated-storage 250 ^
    --db-instance-identifier my_identifier ^
    --db-instance-class db.m5.large ^
    --engine mysql ^
    --master-username admin ^
    --manage-master-user-password ^
    --s3-bucket-name amzn-s3-demo-bucket ^
    --s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename ^
    --s3-prefix bucket_prefix ^
    --source-engine mysql ^
    --source-engine-version 8.0.32 ^
    --max-allocated-storage 1000
```

RDS API

Amazon RDS API を使用して Amazon S3 から新しい MySQL DB インスタンスにデータをインポートするには、RestoreDBInstanceFromS3 オペレーションを呼び出します。

Amazon S3 から Amazon RDS へのバックアップファイルのインポートに関する制限と考慮事項

Amazon S3 から RDS for MySQL DB インスタンスへのバックアップファイルのインポートには、次の制限と考慮事項が適用されます。

- データは新しい DB インスタンスにのみ移行でき、既存の DB インスタンスにはできません。

- Percona XtraBackup を使用してデータを Amazon S3 にバックアップする必要があります。詳細については、「データベースバックアップの作成」を参照してください。
- Amazon S3 バケットと RDS for MySQL DB インスタンスは同じ AWS リージョンに存在する必要があります。
- 次のソースからは復元することはできません。
  - Amazon S3 にエクスポートされた DB インスタンスのスナップショット。また、DB インスタンススナップショットエクスポートから、Amazon S3 バケットにデータを移行することもできません。
  - 暗号化されたソースデータベース。ただし、移行するデータを暗号化することはできます。移行プロセス中にデータを非暗号化のまま維持することもできます。
  - MySQL 5.5 または 5.6 データベース。
- RDS for MySQL では、`mysql schema` に `compression_dictionary*` テーブルが含まれる可能性があるため、Percona Server for MySQL をソースデータベースとしてサポートされていません。
- RDS for MySQL では、メジャーバージョンとマイナーバージョンのどちらでも後方移行はサポートされていません。例えば、MySQL バージョン 8.0 から RDS for MySQL 5.7 に移行することはできません。また、MySQL バージョン 8.0.32 から RDS for MySQL バージョン 8.0.26 に移行することもできません。
- Amazon RDS では、Amazon S3 から db.t2.micro DB インスタンスクラスへのインポートはサポートされていません。ただし、異なる DB インスタンスクラスに復元してから、後で DB インスタンスクラスを変更することができます。インスタンスクラスの詳細については、「DB インスタンスクラスのハードウェア仕様」を参照してください。
- Amazon S3 では、Amazon S3 バケットにアップロードするファイルのサイズが 5 TB に制限されます。バックアップファイルが 5 TB を超える場合は、バックアップファイルを小さいファイルに分割する必要があります。
- Amazon RDS では、Amazon S3 バケットにアップロードするファイルの数が百万までに制限されます。データベースのバックアップデータ (すべての完全および増分バックアップを含む) が百万ファイルを超える場合は、Gzip (.gz)、tar (.tar.gz)、Percona xbstream (.xbstream) ファイルを使用して完全および増分バックアップファイルを Amazon S3 バケットに保存します。Percona XtrabackUp 8.0 は Percona xbstream のみを圧縮用にサポートしています。
- 各 DB インスタンスに管理サービスを提供するために、Amazon RDS では DB インスタンスの作成時に `rdsadmin` ユーザーが作成されます。`rdsamin` は Amazon RDS の予約ユーザーであるため、以下の制限が適用されます。

  - Amazon RDS では、`'rdsadmin'@'localhost'` が定義者である関数、プロシージャ、ビュー、イベント、トリガーは、インポートされません。詳細については、「定義者として 'rdsamin'@'localhost' を使用して保存されたオブジェクト」および「マスターユーザーアカウント権限」を参照してください。
  - DB インスタンスを作成するとき、Amazon RDS はサポートされている最大権限を持つマスターユーザーを作成します。バックアップから復元する際、Amazon RDS では、インポート中のユーザーに割り当てられたサポートされていない権限は自動的に削除されます。

    この影響を受ける可能性のあるユーザーを特定するには、「サポートされていない権限を持つユーザーアカウント」を参照してください。RDS for MySQL でサポートされる権限の詳細については、「RDS for MySQL のロールベースの権限モデル」を参照してください。
- Amazon RDS は、`mysql` スキーマ内のユーザーが作成したテーブルを移行しません。
- `innodb_data_file_path` パラメータを、デフォルトのデータファイル名 `ibdata1:12M:autoextend` を使用するデータファイル 1 つのみで設定する必要があります。この方法を使用すると、2 つのデータファイルを持つデータベースや名前が異なるデータファイルを持つデータベースを移行できます。

  次の例は、Amazon RDS で許可されていないファイル名です。
  - `innodb_data_file_path=ibdata1:50M`
  - `ibdata2:50M:autoextend`
  - `innodb_data_file_path=ibdata01:50M:autoextend`
- デフォルトの MySQL データディレクトリ外で定義されたテーブルを含むソースデータベースから移行することはできません。
- この方法を使用した非圧縮バックアップでサポートされる最大サイズは、64 TiB に制限されています。圧縮バックアップの場合、圧縮解除に必要なスペースを考慮してこの制限は小さくなります。このような場合、サポートされる最大バックアップサイズは (`64 TiB - compressed backup size`) です。

  RDS for MySQL でサポートされている最大データベースサイズについては、「汎用 SSD ストレージ」および「プロビジョンド IOPS SSD ストレージ」を参照してください。
- Amazon RDS は MySQL やその他の外部コンポーネントやプラグインのインポートをサポートしていません。
- Amazon RDS は、データベースからすべてを復元するわけではありません。ソース MySQL システムデータベースからデータベーススキーマと以下の項目の値を保存してから、作成後に復元された RDS for MySQL DB インスタンスに追加することをお勧めします。

- ユーザーアカウント
- 関数
- ストアドプロシージャ
- タイムゾーン情報。タイムゾーン情報は、RDS for MySQL DB インスタンスのローカルオペレーティングシステムからロードされます。詳細については、「MySQL DB インスタンスのローカルタイムゾーン」を参照してください。

定義者として 'rdsamin'@'localhost' を使用して保存されたオブジェクト

Amazon RDS では、'rdsadmin'@'localhost' が定義者である関数、プロシージャ、ビュー、イベント、トリガーは、インポートされません。

ソース MySQL データベースで以下の SQL スクリプトを使用すると、サポートされていない定義子を持つストアド オブジェクトを一覧表示できます。

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
```

```
FROM
    information_schema.events
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

サポートされていない権限を持つユーザーアカウント

RDS for MySQL でサポートされていない権限を持つユーザーアカウントは、サポートされていない権限なしでもインポートされます。サポートされている権限のリストについては、「RDS for MySQL のロールベースの権限モデル」を参照してください。

ソースデータベースで次の SQL クエリを実行すると、サポートされていない権限を持つユーザーアカウントを一覧表示できます。

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

外部の MySQL データベースから Amazon RDS for MySQL DB インスタンスにデータをインポートする

既存の MySQL データベースから RDS for MySQL DB インスタンスに、データをインポートできます。これは、データベースを mysqldump でコピーして、RDS for MySQL DB インスタンスに直接パイプ処理することで行います。mysqldump コマンドラインユーティリティは、データのバックアッ

プや、MySQL サーバーから別の場所への転送によく使用されます。このユーティリティは、MySQL クライアントソフトウェアに含まれています。

> **Note**
>
> MySQL DB インスタンスを使用して大量のデータをインポートまたはエクスポートする場合、xtrabackup バックアップファイルと Amazon S3 を使用して Amazon RDS との間でデータを移動する方が信頼性が高く、高速です。詳細については、「Amazon RDS for MySQL DB インスタンスへのバックアップの復元」を参照してください。

外部のデータベースから Amazon RDS DB インスタンスにデータを移動するための一般的な mysqldump コマンドは、次の例のようになります。値を自分の情報に置き換えます。

```
mysqldump -u local_user \
    --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u RDS_user \
        --port=port_number \
        --host=host_name \
        -pRDS_password
```

> **Important**
>
> -p オプションと入力するパスワードの間にスペースを残していないことを確認します。セキュリティのベストプラクティスとして、この例に表示されているプロンプト以外の認証情報を指定してください。

次の推奨事項と考慮事項に注意してください。

- ダンプファイルから次のスキーマを除外します。
  - sys
  - performance_schema

  - `information_schema`

  `mysqldump` ユーティリティを使用すると、これらのスキーマをデフォルトで除外できます。
- ユーザーや権限を移行する必要がある場合は、再作成するデータ制御言語 (DCL) を生成するツールの使用を検討します。例えば、pt-show-grants ユーティリティがあります。
- インポートを実行するには、そのユーザーに DB インスタンスへのアクセスが許可されていることを確認してください。詳細については、「セキュリティグループによるアクセス制御」を参照してください。

使用するパラメータは次のとおりです。

- `-u` *`local_user`* - ユーザー名を指定します。このパラメータの初回の使用では、`--databases` パラメータによって識別されたローカル MySQL データベースのユーザーアカウントの名前を指定します。
- `--databases` *`database_name`* — Amazon RDS にインポートするローカル MySQL インスタンスのデータベースの名前を指定します。
- `--single-transaction` - ローカルデータベースからロードされたすべてのデータが、ある時点において一貫していることを確認します。`mysqldump` によるデータの読み取り中に他のプロセスがデータを変更する場合は、このパラメータを使用することでデータの整合性を維持できます。
- `--compress` - ローカルデータベースからのデータを、Amazon RDS に送信する前に圧縮することで、ネットワーク帯域幅の消費量を削減します。
- `--order-by-primary` - 各テーブルのデータをプライマリキーでソートすることで、ロード時間を短縮します。
- `--routines` – ストアドプロシージャや関数などのルーチンが、コピーするデータベースに存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にルーチンを除外します。その後、Amazon RDS データベースでルーチンを手動で再作成します。
- `--triggers` – コピーするデータベースにトリガーが存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にトリガーを除外します。その後、Amazon RDS データベースでトリガーを手動で再作成します。
- `--events` – コピーするデータベースにイベントが存在する場合に使用します。パラメータを `0` に設定し、インポートプロセス中にイベントを除外します。その後、Amazon RDS データベースでイベントを手動で再作成します。
- `-p`*`local_password`* - パスワードを指定します。このパラメータの初回の使用では、初期の `-u` パラメータにより識別されるユーザーアカウントのパスワードを指定します。

- -u *RDS_user* - ユーザー名を指定します。このパラメータの 2 回目の使用では、--host パラメータによって識別された MySQL DB インスタンスのデフォルトデータベースのユーザーアカウントの名前を指定します。
- --port *port_number* – MySQL DB インスタンスのポートを指定します。DB インスタンスの作成時に値を変更していない限り、デフォルトでは 3306 です。
- --host *host_name* — Amazon RDS DB インスタンスのエンドポイント、例えば からのドメインネームシステム (DNS) 名を指定します。myinstance.123456789012.us-east-1.rds.amazonaws.comエンドポイントの値は、Amazon RDS コンソールの DB インスタンスの詳細で確認できます。
- -p*RDS_password* - パスワードを指定します。このパラメータの 2 回目の使用では、2 回目の -u パラメータにより識別されるユーザーアカウントのパスワードを指定します。

Amazon RDS データベースで、ストアドプロシージャ、トリガー、関数、イベントを必ず手動で作成してください。コピーするデータベースにこれらのオブジェクトのいずれかが含まれる場合は、mysqldump の実行時に除外します。これを行うには、mysqldump コマンドに次のパラメータを含めます。

- --routines=0
- --triggers=0
- --events=0

例

次の例では、ローカルホストにある world サンプルデータベースを、RDS for MySQL DB インスタンスにコピーします。値を自分の情報に置き換えます。

Linux、macOS、Unix の場合:

```
sudo mysqldump -u local_user \
    --databases world \
    --single-transaction \
    --compress \
    --order-by-primary  \
    --routines=0 \
    --triggers=0 \
    --events=0 \
    -plocal_password | mysql -u rds_user \
        --port=3306 \
```

```
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com \
        -pRDS_password
```

Windows の場合:

Windows プログラムメニューの [コマンドプロンプト] を右クリックし、[管理者として実行] を選択して開いたコマンドプロンプトで次のコマンドを実行します。値を自分の情報に置き換えます。

```
mysqldump -u local_user ^
    --databases world ^
    --single-transaction ^
    --compress ^
    --order-by-primary  ^
    --routines=0 ^
    --triggers=0 ^
    --events=0 ^
    -plocal_password | mysql -u RDS_user ^
        --port=3306 ^
        --host=my_instance.123456789012.us-east-1.rds.amazonaws.com ^
        -pRDS_password
```

> **Note**
>
> セキュリティのベストプラクティスとして、例に表示されているプロンプト以外の認証情報を指定します。

ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする

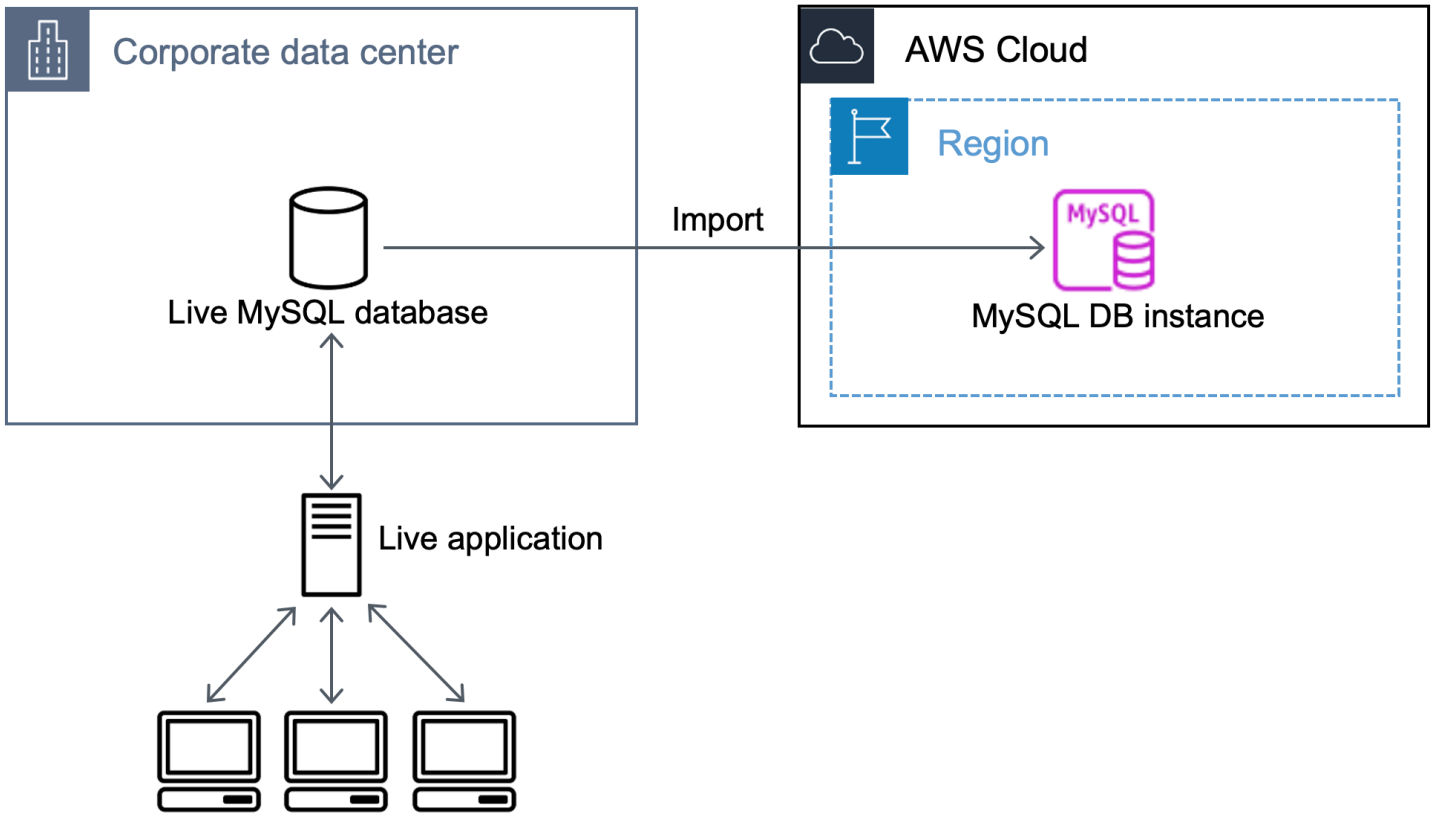
場合によっては、ライブアプリケーションをサポートする外部の MySQL データベースから RDS for MySQL DB インスタンス、または RDS for MySQL マルチ AZ DB クラスターにデータをインポートする必要があります。次の手順を使用して、アプリケーションの可用性への影響を最小限に抑えることができます。この手順は、巨大なデータベースを使用する場合にも役立ちます。この手順を使用すると、ネットワーク経由で AWS に渡されるデータ量を削減することで、インポートのコストを削減できます。

この手順では、データベースデータのコピーを Amazon EC2 インスタンスに送信し、そのデータを新しい Amazon RDS データベースにインポートします。次に、レプリケーションを使用して、Amazon RDS データベースをライブ外部インスタンスで最新の状態にした後、アプリケーショ

ンを Amazon RDS データベースにリダイレクトします。バイナリログの調整に基づいてレプリケーションを設定します。

> **Note**
>
> RDS for MySQL DB インスタンスにデータをインポートする際、シナリオでサポートされている場合は、バックアップファイルと Amazon S3 を使用して、Amazon RDS との間でデータを移動することをお勧めします。詳細については、「Amazon RDS for MySQL DB インスタンスへのバックアップの復元」を参照してください。

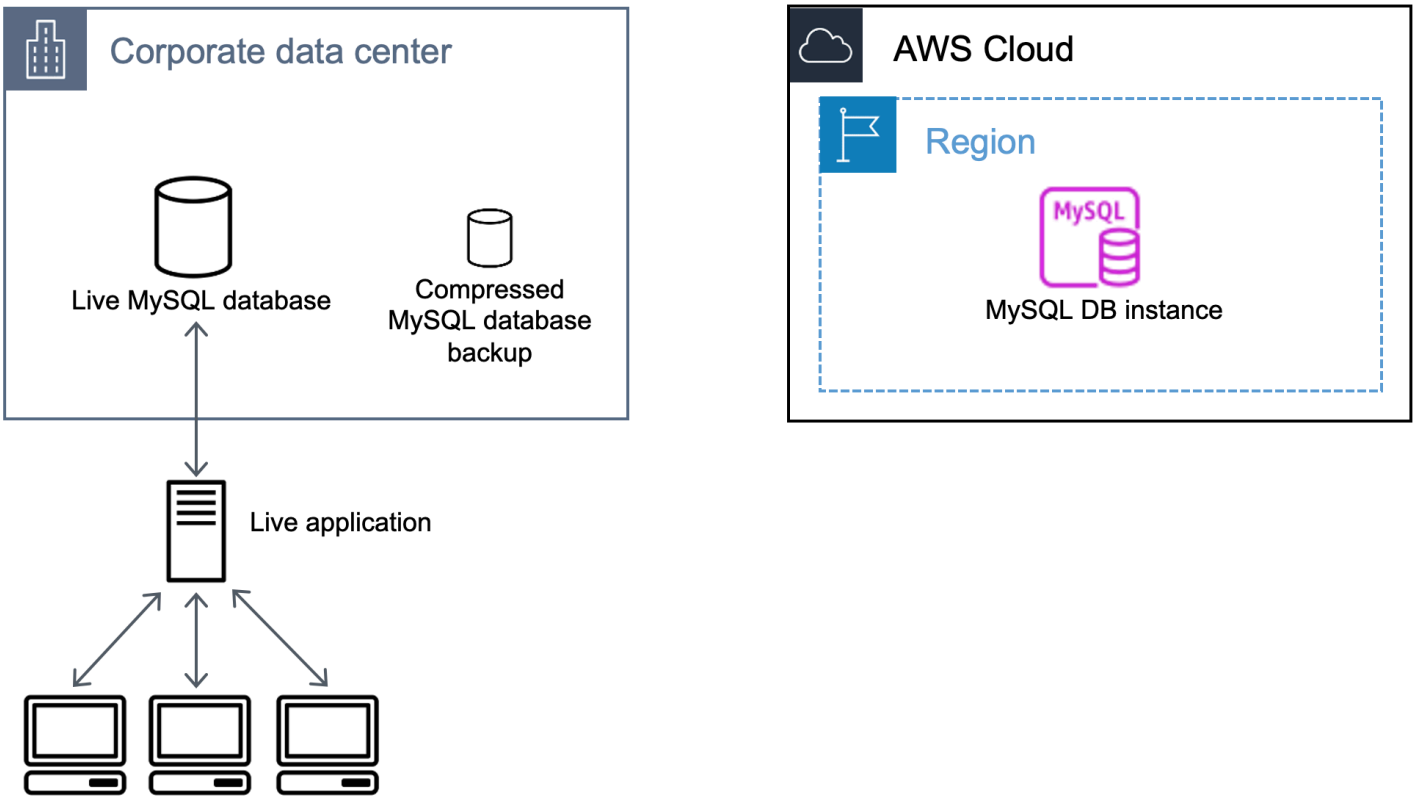
次の図は、外部 MySQL データベースを Amazon RDS 上の MySQL データベースにインポートする様子を示しています。



タスク 1: 既存のデータベースのコピーを作成する

最小限のダウンタイムで RDS for MySQL データベースに大量のデータを移行するプロセスでは、最初のステップとしてソースデータのコピーを作成します。

次の図は、MySQL データベースのバックアップの作成を示しています。



SQL 形式または区切り文字付きテキスト形式でデータベースのバックアップを作成するには、mysqldump ユーティリティを使用できます。非運用環境で各形式のテスト実行を行って、どちらの方法が mysqldump の実行時間が短いか確認することをお勧めします。

また、ロードに区切り文字付きテキスト形式を使用することでもたらされるメリットに対して、mysqldump のパフォーマンスの重み付けをすることをお勧めします。区切り文字付きテキスト形式を使用したバックアップでは、ダンプされる各テーブルについてタブ区切りテキストファイルを作成されます。データベースのインポートに必要な時間を短縮するため、LOAD DATA LOCAL INFILE コマンドを使用してこれらのファイルを同時にロードできます。詳細については、任意のソースからデータをインポートする手順の「ステップ 5: データをロードする」を参照してください。

バックアップ操作をスタートする前に、Amazon RDS にコピーする MySQL データベースでレプリケーションのオプションを設定してください。レプリケーションのオプションには、バイナリログ記録の有効化や一意のサーバー ID の設定が含まれます。これらのオプションを設定すると、サーバーはデータベーストランザクションのログ作成をスタートし、このプロセスの後でこのサーバーをソースレプリケーションインスタンスにするために準備します。

次の推奨事項と考慮事項に注意してください。

- --single-transaction オプションは、データベースの一貫した状態をダンプするため、mysqldump と共に使用します。有効なダンプファイルを確保するために、mysqldump の実行中はデータ定義言語 (DDL) ステートメントを実行しないでください。これらのオペレーションに対してメンテナンスウィンドウをスケジュールできます。
- ダンプファイルから次のスキーマを除外します。
  - sys
  - performance_schema
  - information_schema

  mysqldump ユーティリティを使用すると、これらのスキーマをデフォルトで除外できます。
- ユーザーや権限を移行する必要がある場合は、再作成するデータ制御言語 (DCL) を生成するツールの使用を検討します。例えば、pt-show-grants ユーティリティがあります。

レプリケーションオプションを設定するには

1. my.cnf ファイルを編集します。このファイルは /etc にあります。

   ```
   sudo vi /etc/my.cnf
   ```

   log_bin オプションと server_id オプションを [mysqld] に追加します。log_bin オプションは、バイナリログファイルのファイル名識別子を提供します。server_id オプションは、ソースとレプリカの関係のサーバーに一意の識別子を提供します。

   次の例は、my.cnf ファイルの更新された [mysqld] セクションを示しています。

   ```
   [mysqld]
   log-bin=mysql-bin
   server-id=1
   ```

   詳細については、MySQL ドキュメントの「Setting the Replication Source Configuration」を参照してください。
2. マルチ AZ DB クラスターでのレプリケーションでは、ENFORCE_GTID_CONSISTENCY および GTID_MODE パラメータを ON に設定します。

   ```
   mysql> SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
   ```

```
mysql> SET @@GLOBAL.GTID_MODE = ON;
```

これらの設定は、DB インスタンスでのレプリケーションには必要ありません。

3. mysql サービスを再起動します。

```
sudo service mysqld restart
```

既存のデータベースのバックアップコピーを作成するには

1. mysqldump ユーティリティを使用し、SQL 形式または区切り文字付きテキスト形式を指定して、データのバックアップを作成します。

   MySQL 8.0.25 以前のバージョンでは、サーバー間のレプリケーションを開始するために使用できるバックアップファイルを作成するには、--master-data=2 を指定します。MySQL 8.0.26 以降のバージョンでは、サーバー間のレプリケーションを開始するために使用できるバックアップファイルを作成するには、--source-data=2 を指定します。詳細については、MySQL ドキュメントの「mysqldump — A Database Backup Program」を参照してください。

   パフォーマンスを向上させ、データ整合性を確保するために、mysqldump で --order-by-primary オプションと --single-transaction オプションを使用します。

   MySQL システムデータベースをバックアップに含めないためには、mysqldump で --all-databases オプションを使用しないでください。詳細については、MySQL ドキュメントの「Creating a Data Snapshot Using mysqldump」を参照してください。

   バックアップファイルが作成されるディレクトリを書き込み可能にするために、必要に応じて chmod を使用します。

   > **Important**
   >
   > Windows で、管理者としてコマンドウィンドウを実行します。

   - SQL 出力を作成するには、次のコマンドを使用します。

     Linux、macOS、Unix の場合:

```
sudo mysqldump \
    --databases database_name \
    --master-data=2  \
    --single-transaction \
    --order-by-primary \
    -r backup.sql \
    -u local_user \
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。

Windows の場合:

```
mysqldump ^
    --databases database_name ^
    --master-data=2  ^
    --single-transaction ^
    --order-by-primary ^
    -r backup.sql ^
    -u local_user ^
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。

- 区切り文字付きテキスト出力を作成するには、次のコマンドを使用します。

  Linux、macOS、Unix の場合:

```
sudo mysqldump \
    --tab=target_directory \
    --fields-terminated-by ',' \
    --fields-enclosed-by '"' \
```

```
    --lines-terminated-by 0x0d0a \
    database_name \
    --master-data=2 \
    --single-transaction \
    --order-by-primary \
    -ppassword
```

Windows の場合:

```
mysqldump ^
    --tab=target_directory ^
    --fields-terminated-by "," ^
    --fields-enclosed-by """ ^
    --lines-terminated-by 0x0d0a ^
    database_name ^
    --master-data=2 ^
    --single-transaction ^
    --order-by-primary ^
    -ppassword
```

> **Note**
>
> セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定します。
> Amazon RDS データベースで、ストアドプロシージャ、トリガー、関数、イベントを必ず手動で作成してください。コピーするデータベースにこれらのオブジェクトのいずれかが含まれる場合は、mysqldump の実行時に除外します。これを行うには、mysqldump コマンドで以下の引数を含めます。
>
> - --routines=0
> - --triggers=0
> - --events=0

MySQL 8.0.22 以前のバージョンでは、mysqldump を実行して区切りテキスト形式を指定すると、CHANGE MASTER TO コメントが返されます。このコメントには、マスターログのファイル名と場所が含まれます。MySQL 8.0.23 以降のバージョンでは、区切りテキスト形式を使用して mysqldump を実行すると、CHANGE REPLICATION SOURCE TO コメントが返されます。このコメントには、ソースログファイルの名前と位置が含まれます。外部インスタンスが

MySQL 8.0.23 以降の場合は、MASTER_LOG_FILE および MASTER_LOG_POS の値を書き留めてください。これらの値は、レプリケーションを設定するときに必要です。

MySQL 8.0.22 以前のバージョンでは、次の出力が返されます。

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

MySQL 8.0.23 以降のバージョンでは、次の出力が返されます。

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE SOURCE TO SOURCE_LOG_FILE='mysql-bin-changelog.000031',
 SOURCE_LOG_POS=107;
```

MySQL 8.0.22 以前のバージョンで SQL 形式を使用している場合は、バックアップファイル内の CHANGE MASTER TO コメントでマスターログファイル名と位置を取得できます。MySQL 8.0.23 以降のバージョンで SQL 形式を使用している場合は、バックアップファイル内の CHANGE REPLICATION SOURCE TO コメントでソースログファイル名と位置を取得できます。

2. Amazon RDS データベースにデータをコピーするために必要なネットワークリソースの量を減らすために、コピーされたデータを圧縮します。バックアップファイルのサイズをメモします。この情報は、作成する Amazon EC2 インスタンスの大きさを決定するときに必要です。作業が終了したら、GZIP または任意の圧縮ユーティリティを使用してバックアップファイルを圧縮します。
   - SQL 出力を圧縮するには、次のコマンドを使用します。

     ```
     gzip backup.sql
     ```

   - 区切り文字付きテキスト出力を圧縮するには、次のコマンドを使用します。

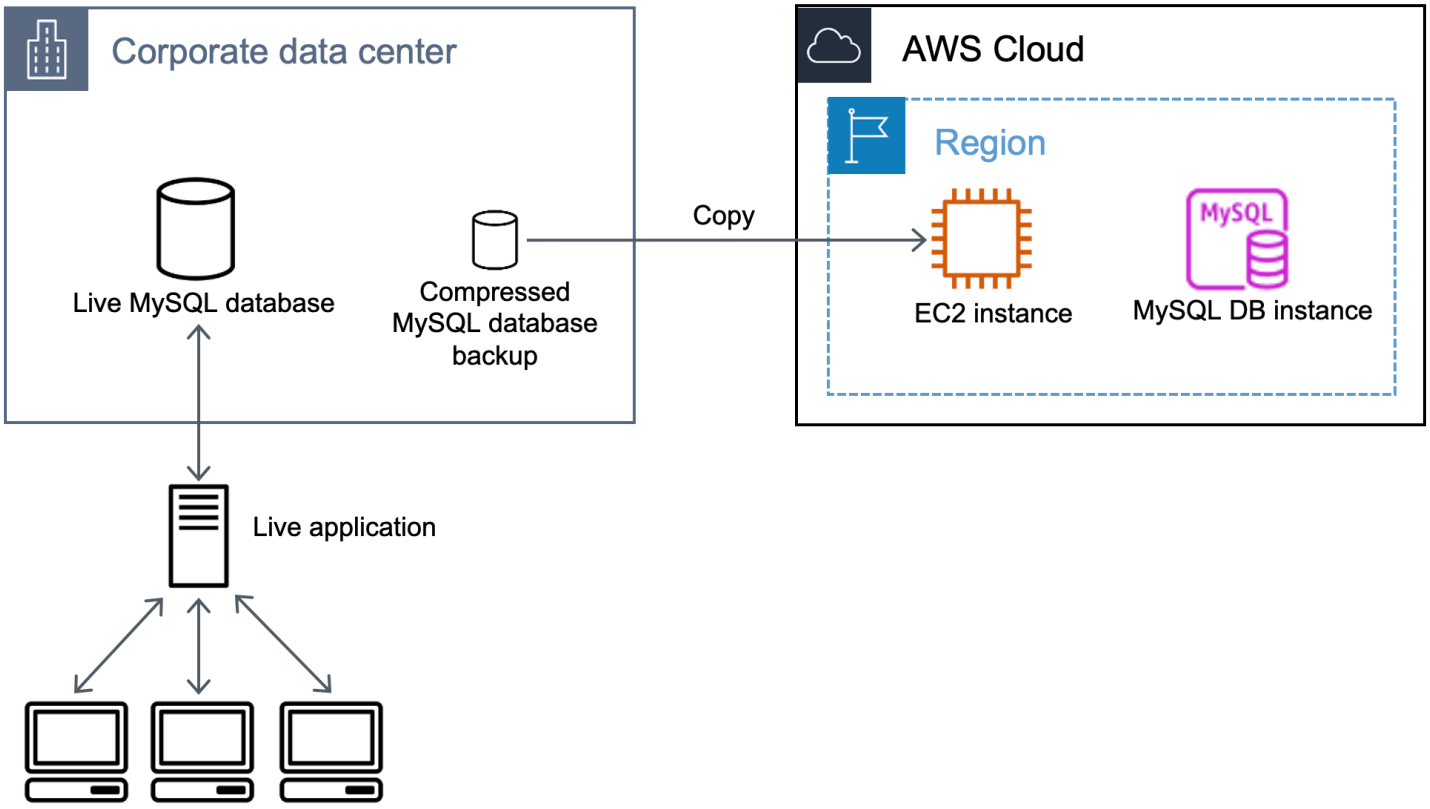
     ```
     tar -zcvf backup.tar.gz target_directory
     ```

タスク 2: Amazon EC2 インスタンスを作成し、圧縮したデータベースをコピーする

圧縮したデータベースのバックアップファイルを Amazon EC2 インスタンスにコピーする場合、データベースインスタンス間で非圧縮データを直接コピーするよりも必要なネットワークリソース

は少なくなります。データを Amazon EC2 にコピーしたら、そこから MySQL データベースに直接コピーできます。ネットワークリソースのコストを節約するには、Amazon EC2 インスタンスが Amazon RDS DB インスタンスと同じ AWS リージョンに存在している必要があります。Amazon EC2 インスタンスを Amazon RDS データベースと同じ AWS リージョンに配置することで、インポート時のネットワークレイテンシーも低減されます。

次の図は、Amazon EC2 インスタンスへのデータベースバックアップのコピーを示しています。



Amazon EC2 インスタンスを作成し、データをコピーするには

1. Amazon RDS データベースを作成する予定の AWS リージョンに、仮想プライベートクラウド (VPC)、VPC セキュリティグループ、および VPC サブネットを作成します。VPC セキュリティグループのインバウンドルールで、アプリケーションが AWS に接続するために必要な IP アドレスを許可していることを確認します。IP アドレスの範囲 (`203.0.113.0/24` など) や別の VPC セキュリティグループを指定できます。Amazon VPC コンソールを使用して、VPC、サブネット、セキュリティグループを作成および管理できます。詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「Amazon VPC の開始方法」を参照してください。
2. Amazon EC2 管理コンソールを開き、Amazon EC2 インスタンスと Amazon RDS データベースの両方が含まれる AWS リージョンを選択します。ステップ 1 で作成した VPC、サブネット、セキュリティグループを使用して Amazon EC2 インスタンスを起動します。非圧縮の場合のデー

タベースバックアップファイルに十分なストレージを備えたインスタンスタイプを選択していることを確認します。Amazon EC2 インスタンスの詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EC2 の使用を開始する」を参照してください。

3. Amazon EC2 インスタンスから Amazon RDS データベースに接続するには、VPC セキュリティグループを編集します。EC2 インスタンスのプライベート IP アドレスを指定するインバウンドルールを追加します。このプライベート IP アドレスは、EC2 コンソールウィンドウの [Instance] ペインの [Details] タブで確認できます。VPC セキュリティグループを編集してインバウンドルールを追加するには、EC2 コンソールのナビゲーションペインの [Security Groups] (セキュリティグループ) でセキュリティグループを選択してから、EC2 インスタンスのプライベート IP アドレスを指定して MySQL または Aurora のインバウンドルールを追加します。VPC セキュリティグループにインバウンドルールを追加する方法については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルール」を参照してください。
4. ローカルシステムから Amazon EC2 インスタンスに、圧縮されたデータベースバックアップファイルをコピーします。必要に応じて chmod を使用して、Amazon EC2 インスタンスのターゲットディレクトリに対する書き込みアクセス許可があることを確認します。scp または Secure Shell (SSH) クライアントを使用してファイルをコピーできます。以下は scp コマンドの例です。

```
scp -r -i key pair.pem backup.sql.gz ec2-user@EC2 DNS:/target_directory/backup.sql.gz
```

> **Important**
>
> 機密データをコピーするときは、必ず安全なネットワーク転送プロトコルを使用してください。

5. Amazon EC2 インスタンスに接続し、次のコマンドを使用して最新のアップデートと MySQL クライアントツールをインストールします。

```
sudo yum update -y
sudo yum install mysql -y
```

詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の Linux インスタンスの「インスタンスをクリーンアップする」を参照してください。

> **Important**
>
> この例では、MySQL クライアントを Amazon Linux ディストリビューションの Amazon マシンイメージ (AMI) にインストールします。この例では、Ubuntu や Red Hat Enterprise

Linux などの別のディストリビューションには MySQL クライアントをインストールできません。MySQL のインストールの詳細については、MySQL ドキュメントの「Installing MySQL」を参照してください。

6. Amazon EC2 インスタンスに接続されている間に、データベースバックアップファイルを解凍します。次のコマンドはその例です。

   - SQL 出力を解凍するには、次のコマンドを使用します。

     ```
     gzip backup.sql.gz -d
     ```

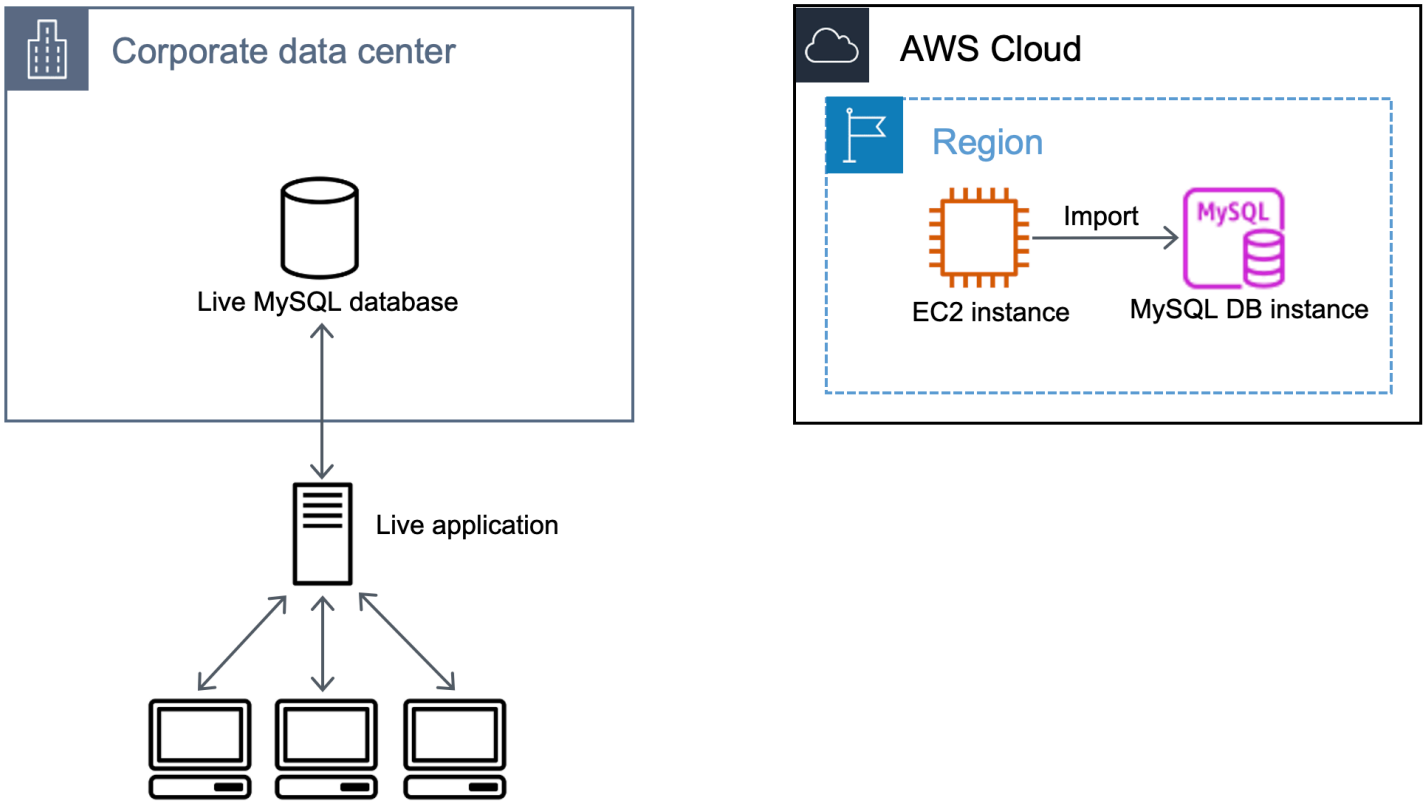
   - 区切り文字付きテキスト出力を解凍するには、次のコマンドを使用します。

     ```
     tar xzvf backup.tar.gz
     ```

タスク 3: MySQL データベースを作成し、Amazon EC2 インスタンスからデータをインポートする

Amazon EC2 インスタンスと同じ AWS リージョンに RDS for MySQL DB インスタンス、または RDS for MySQL マルチ AZ DB クラスターを作成することにより、Amazon EC2 からのデータベースのバックアップファイルをインターネット経由よりも速くインポートできます。

次の図は、Amazon EC2 インスタンスから MySQL データベースにバックアップをインポートする様子を示しています。



MySQL データベースを作成し、データをインポートするには

1. この Amazon RDS データベースについて予想されるワークロードをサポートするのに必要な DB インスタンスクラスとストレージ領域の容量を決定します。このプロセスの一環として、データロードの手順に十分な領域と処理能力を決定します。また、本番稼働ワークロードの処理に必要なものも決定します。これは、ソースの MySQL データベースのサイズおよびリソースに基づいて見積もることができます。詳細については、「DB インスタンスクラス」を参照してください。
2. Amazon EC2 インスタンスを含む AWS リージョンに、DB インスタンスまたはマルチ AZ DB クラスターを作成します。

   RDS for MySQL マルチ AZ DB クラスターを作成するには、「Amazon RDS 用のマルチ AZ DB クラスターの作成」の手順に従います。

   RDS for MySQL DB インスタンスを作成するには、「Amazon RDS DB インスタンスの作成」の手順に従い、以下のガイドラインに従ってください。

   - ソース DB インスタンスと互換性のある DB エンジンのバージョンを指定します。

- Amazon EC2 インスタンスと同じ仮想プライベートクラウド (VPC) と VPC セキュリティグループを指定します。これにより、Amazon EC2 インスタンスと Amazon RDS インスタンスはネットワーク上で相互に表示されることが確認できます。DB インスタンスがパブリックにアクセスできることを確認してください。以下のセクションで説明するように、ソースデータベースでレプリケーションをセットアップするには、DB インスタンスにパブリックアクセス可能である必要があります。
- データベースのバックアップのインポートが完了するまで、複数のアベイラビリティーゾーン、バックアップ保持、リードレプリカを設定しないでください。インポートが完了したら、本稼働インスタンスについて、マルチ AZ とバックアップ保持を設定できます。

3. Amazon RDS データベースのデフォルトの設定オプションを確認します。データベースのデフォルトパラメータグループに必要な設定オプションがない場合は、別のパラメータグループを検索するか、新しいパラメータグループを作成します。パラメータグループの作成の詳細については、「Amazon RDS のパラメータグループ」を参照してください。
4. マスターユーザーとして、新しい Amazon RDS データベースに接続します。DB インスタンスにアクセスする必要がある管理者、アプリケーション、およびサービスのサポートに必要なユーザーを作成します。Amazon RDS データベースのホスト名は、この DB インスタンスの [エンドポイント] の値からポート番号を除いた値 (例: `mysampledb.123456789012.us-west-2.rds.amazonaws.com`) です。エンドポイントの値は、Amazon RDS コンソールのデータベースの詳細で確認できます。
5. Amazon EC2 インスタンスに接続します。詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の Linux インスタンスの「インスタンスをクリーンアップする」を参照してください。
6. `mysql` コマンドを使用して、Amazon EC2 インスタンスからリモートホストとして Amazon RDS データベースに接続します。コマンドの例を次に示します。

   ```
   mysql -h host_name -P 3306 -u db_master_user -p
   ```

   host_name は、Amazon RDS データベースのエンドポイントです。
7. `mysql` プロンプトで、データベースダンプファイルの名前を渡して `source` コマンドを実行します。このコマンドは、Amazon RDS DB インスタンスにデータをロードします。

   - SQL 形式の場合は、次のコマンドを使用します。

     ```
     mysql> source backup.sql;
     ```

- 区切り文字付きテキスト形式の場合は、Amazon RDS データベースをセットアップしたときに作成したデフォルトのデータベースではない場合、まず、データベースを作成します。

```
mysql> create database database_name;
mysql> use database_name;
```

次にテーブルを作成します。

```
mysql> source table1.sql
mysql> source table2.sql
etc...
```

次にデータをインポートします。

```
mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TERMINATED BY
 ',' ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
etc...
```

パフォーマンスを向上させるために、複数の接続からこれらのオペレーションをパラレル実行して、すべてのテーブルを同時に作成およびロードすることができます。

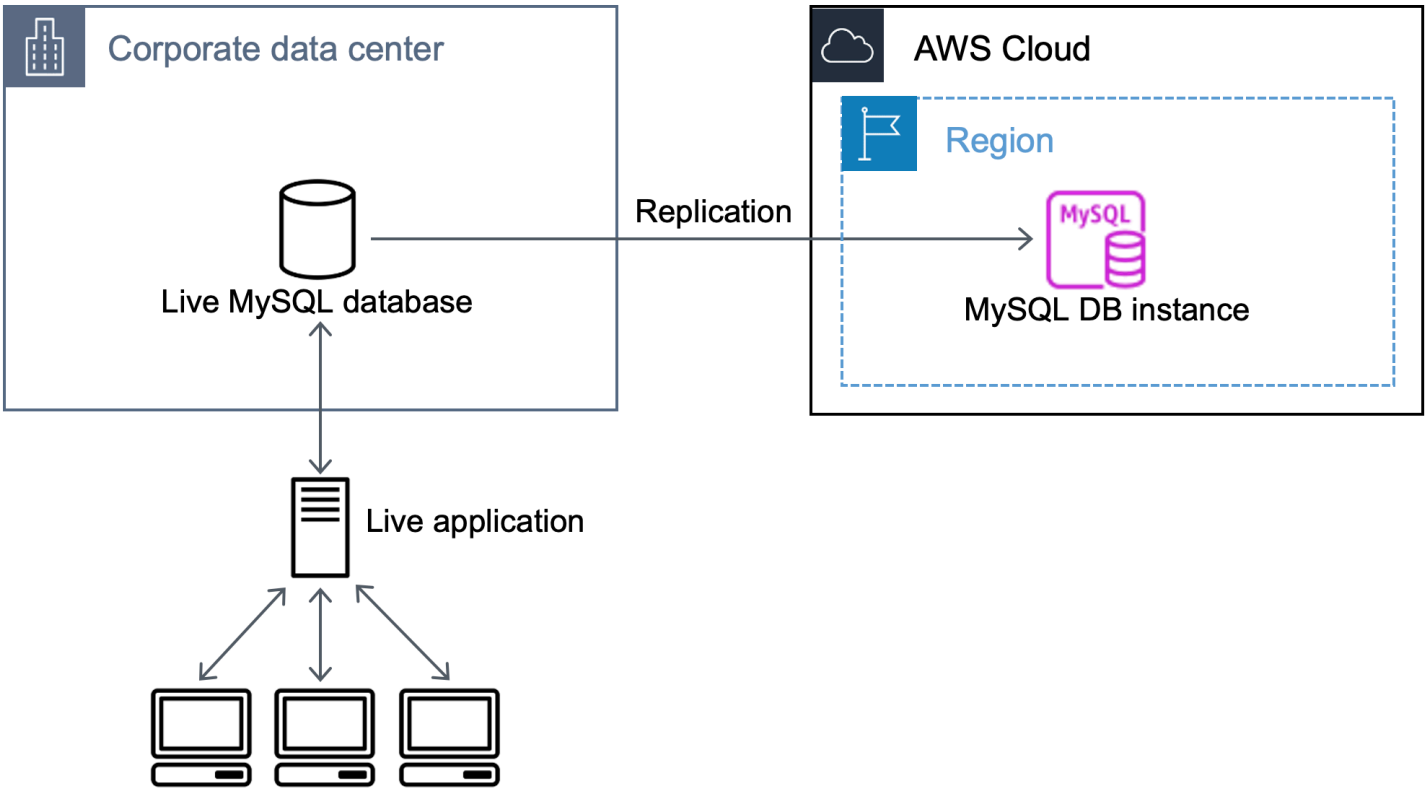
> **Note**
>
> 最初にテーブルをダンプした際、mysqldump でデータをフォーマットするオプションを使用した場合は、データファイルのコンテンツが適切に解釈できるように、LOAD DATA LOCAL INFILE でも必ず同じオプションを使用する必要があります。

8. インポートしたデータベースの単一のテーブルまたは 2 つのテーブルに対してシンプルな SELECT クエリを実行して、インポートが正常に完了したかを検証します。

この手順で使用された Amazon EC2 インスタンスが今後不要な場合は、EC2 インスタンスを終了して、AWS リソース使用率を減らします。EC2 インスタンスを終了するには、「Amazon Elastic Compute Cloud ユーザーガイド」の「インスタンスの終了」を参照してください。

タスク 4: 外部データベースから新しい Amazon RDS データベースにデータをレプリケートする

ソースデータベースは、データをコピーして MySQL データベースに転送するまでに更新された可能性があります。その場合は、レプリケーションを使用して、コピーしたデータベースをソースデータベースで最新のものにすることができます。



Amazon RDS データベースでレプリケーションを開始するために必要なアクセス許可は限定されており、Amazon RDS マスターユーザーは利用できません。このため、メジャーエンジンバージョンに適した Amazon RDS ストアドプロシージャを使用します。

- mysql_rds_set_external_master (RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)
- レプリケーションを設定するには mysql.rds_set_external_master_gtid、レプリケーションを開始するには mysql.rds_start_replication

レプリケーションをスタートするには

タスク 1 のレプリケーションオプションの設定時に、バイナリログ記録を有効にし、ソースデータベースに一意のサーバー ID を設定しました。これで、ライブデータベースをソースレプリケーションインスタンスとして、Amazon RDS データベースをレプリカとしてセットアップできます。

1. Amazon RDS コンソールで、ソースデータベースをホストするサーバーの IP アドレスを Amazon RDS データベースの VPC セキュリティグループに追加します。VPC セキュリティグループの詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルールを設定する」を参照してください。

   ソースインスタンスと通信できるためには、Amazon RDS データベースの IP アドレスからの接続を許可するようにローカルネットワークを設定することも必要になる場合があります。Amazon RDS データベースの IP アドレスを確認するには、host コマンドを使用します。

   ```
   host host_name
   ```

   host_name は、Amazon RDS データベースのエンドポイントの DNS 名 (例: myinstance.123456789012.us-east-1.rds.amazonaws.com) です。エンドポイントの値は、Amazon RDS コンソールの DB インスタンスの詳細で確認できます。

2. 選択したクライアントを使用して、ソースインスタンスに接続し、レプリケーションに使用するユーザーを作成します。このアカウントはレプリケーション専用に使用され、セキュリティを強化するためにドメインに制限する必要があります。コマンドの例を次に示します。

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

3. ソースインスタンスについて、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。たとえば、すべてのデータベースに対する REPLICATION CLIENT と REPLICATION SLAVE の特権を、「repl_user」ユーザーに付与するには、次のコマンドを実行します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

4. Amazon RDS データベースをレプリカにします。マスターユーザーとして Amazon RDS データベースに接続し、適切な Amazon RDS ストアドプロシージャを使用して、ソースデータベースをソースレプリケーションインスタンスとして指定します。

   - mysql_rds_set_external_master (RDS for MySQL メジャーバージョン 8.0 以前)
   - mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)

   SQL 形式のバックアップファイルがある場合は、ステップ 4 で確認したマスターログのファイル名とマスターログの位置を使用します。区切り文字テキスト形式を使用した場合は、バックアップファイルの作成時に決定した名前と場所を使用します。次のコマンドはその例です。

   MySQL バージョン 8.4 以降

   ```
   CALL mysql.rds_set_external_source ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   MySQL 8.0 以前のバージョン

   ```
   CALL mysql.rds_set_external_master ('myserver.mydomain.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   > **Note**
   >
   > セキュリティのベストプラクティスとして、ここに表示されているプロンプト以外の認証情報を指定してください。

5. Amazon RDS データベースでレプリケーションを開始するには、mysql.rds_start_replication ストアドプロシージャを使用する次のコマンドを実行します。

   ```
   CALL mysql.rds_start_replication;
   ```

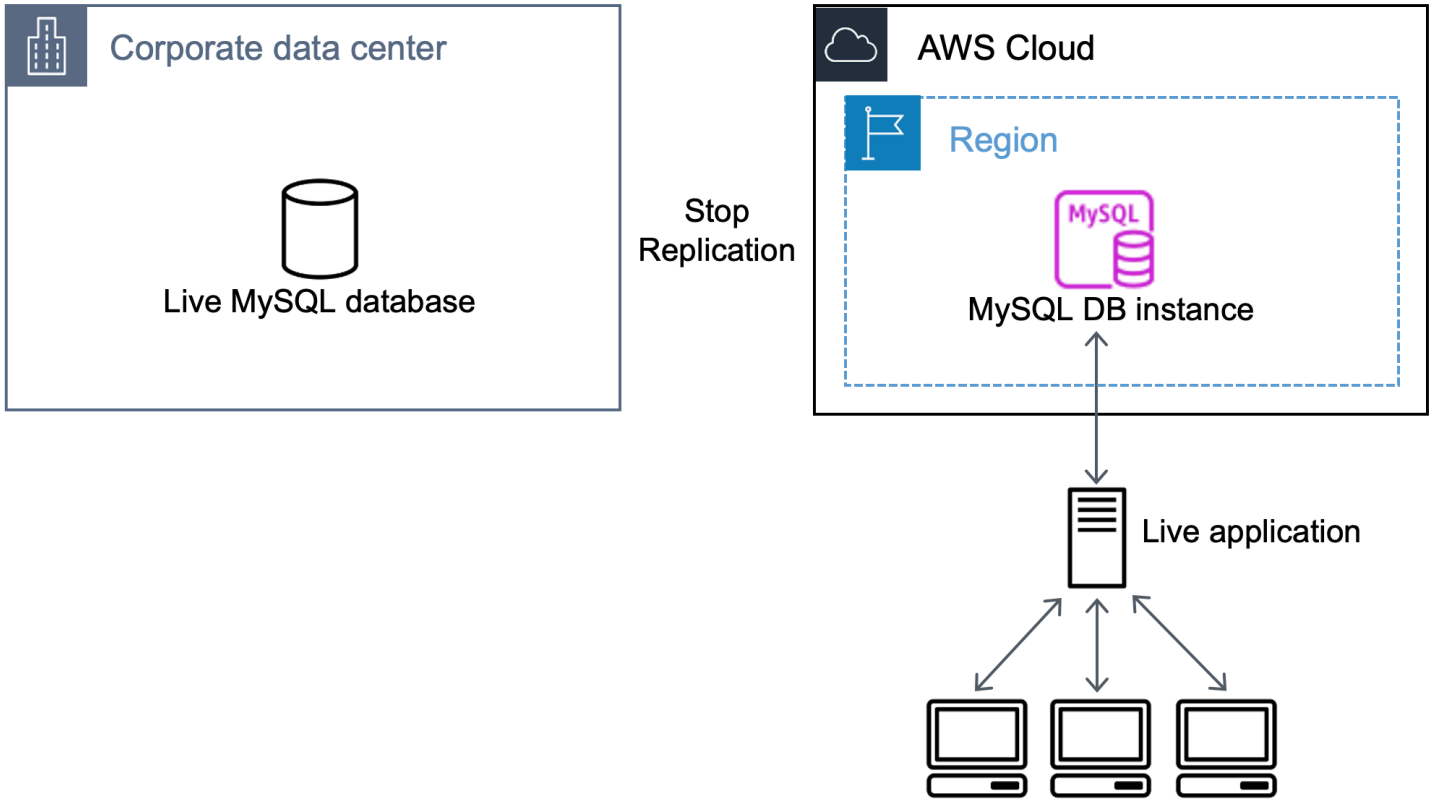
6. Amazon RDS データベースで、レプリカがいつソースレプリケーションインスタンスの最新の状態に更新されるかを特定するには、SHOW REPLICA STATUS コマンドを実行します。SHOW REPLICA STATUS コマンドの結果には、Seconds_Behind_Master フィールドが含まれます。Seconds_Behind_Master フィールドが 0 を返す場合、レプリカはソースレプリケーションインスタンスで最新の状態になります。

> **Note**
>
> MySQL の旧バージョンは、`SHOW REPLICA STATUS` ではなく `SHOW SLAVE STATUS` を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

7. Amazon RDS データベースが最新の状態になったら、必要に応じてデータベースを復元できるように、自動バックアップを有効にします。Amazon RDS コンソールを使用して、Amazon RDS データベースの自動バックアップを有効化または変更できます。詳細については、「バックアップの概要」を参照してください。

タスク 5: ライブアプリケーションを Amazon RDS インスタンスにリダイレクトする

MySQL データベースがソースレプリケーションインスタンスで最新の状態になったら、ライブアプリケーションを更新して、Amazon RDS インスタンスを使用できます。



ライブアプリケーションを MySQL データベースにリダイレクトしてレプリケーションを停止するには

1. Amazon RDS データベースの VPC セキュリティグループを追加するには、アプリケーションをホストするサーバーの IP アドレスを追加します。VPC セキュリティグループの変更の詳細については、「Amazon Virtual Private Cloud ユーザーガイド」の「セキュリティグループのルールを設定する」を参照してください。
2. SHOW REPLICA STATUS コマンドの結果の `Seconds_Behind_Master` フィールドが 0 であることを確認します。この値は、レプリカがソースレプリケーションインスタンスの最新の状態であることを示します。

   ```
   SHOW REPLICA STATUS;
   ```

   > **Note**
   >
   > MySQL の旧バージョンは、`SHOW REPLICA STATUS` ではなく `SHOW SLAVE STATUS` を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

3. トランザクションが終了したら、ソースへのすべての接続を閉じます。
4. Amazon RDS データベースを使用するようにアプリケーションを更新します。この更新には、一般に、Amazon RDS データベースのホスト名とポート、接続に使用するユーザーアカウントとパスワード、および使用するデータベースを識別する接続設定の変更が含まれます。
5. DB インスタンスに接続します。

   マルチ AZ DB クラスターの場合は、ライター DB インスタンスに接続します。
6. mysql.rds_stop_replication ストアドプロシージャを使用する次のコマンドを実行して、Amazon RDS インスタンスのレプリケーションを停止します。

   ```
   CALL mysql.rds_stop_replication;
   ```

7. Amazon RDS データベースで適切な Amazon RDS ストアドプロシージャを使用して、このインスタンスがレプリカとして識別されないようにレプリケーション設定をリセットします。
   - mysql_rds_reset_external_master (RDS for MySQL メジャーバージョン 8.0 以前)
   - mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降)

   MySQL バージョン 8.4 以降

```
CALL mysql.rds_reset_external_source;
```

MySQL 8.0 以前のバージョン

```
CALL mysql.rds_reset_external_master;
```

8. マルチ AZ のサポートやリードレプリカなど、Amazon RDS のその他の機能を有効にします。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」および「DB インスタンスのリードレプリカの操作」を参照してください。

任意のソースから Amazon RDS for MySQL DB インスタンスにデータをインポートする

Amazon RDS を使用すると、既存の MySQL データを任意のソースから RDS for MySQL DB インスタンスに移行できます。オンプレミスデータベース、他のクラウドプロバイダー、または既存の RDS for MySQL DB インスタンスからターゲット RDS for MySQL DB インスタンスにデータを転送できます。この機能を使用すると、データベースの統合、ディザスタリカバリソリューションの実装、セルフマネージド型データベースからの移行を行うことができます。一般的なシナリオには、セルフホスト型 MySQL サーバーからフルマネージド型の Amazon RDS DB インスタンスへの移行、複数の MySQL データベースの 1 つの DB インスタンスへの統合、本番データを使用したテスト環境の作成などがあります。以下のセクションでは、`mysqldump`、バックアップファイル、レプリケーションなどのメソッドを使用して MySQL データをインポートする詳細な手順について説明します。

ステップ 1: ロードするデータを含むフラットファイルを作成する

カンマ区切り値 (CSV) などの一般的な形式を使用して、ロードするデータを保存します。各テーブルには独自のファイルが必要です。複数のテーブルのデータを同じファイルに結合することはできません。各ファイルに、対応するテーブルと同じ名前を付けます。ファイル拡張子は何でもかまいません。例えば、テーブル名が `sales` の場合、ファイル名は `sales.csv` または `sales.txt` になります。

可能であれば、ロードされるテーブルのプライマリキーでデータをソートします。これにより、ロード時間が大幅に短縮され、ディスクストレージの要件が最小限に抑えられます。

この手順の速度と効率は、ファイルのサイズを小さく保つかどうかによって決まります。個々のファイルの非圧縮サイズが 1 GiB を超える場合、複数のファイルに分割して、1 つずつロードしてください。

Unix のようなシステム (Linux など) では、split コマンドを使用します。例えば、次のコマンドでは sales.csv ファイルが 1 GiB 未満の複数のファイルに分割されます。ただし、分割されるのは改行でのみです (-C 1024m)。新しいファイルの名前には、昇順の数値サフィックスが含まれます。次のコマンドは、sales.part_00 や sales.part_01 などの名前のファイルを生成します。

```
split -C 1024m -d sales.csv sales.part_
```

他のオペレーティングシステムにも同様のユーティリティを使用できます。

フラットファイルはどこにでも保存できます。ただし、ステップ 5 でデータをロードするときは、ファイルが存在するのと同じ場所から mysql シェルを呼び出すか、LOAD DATA LOCAL INFILE の実行時にファイルの絶対パスを使用する必要があります。

ステップ 2: ターゲット DB インスタンスにアクセスしているすべてのアプリケーションを停止する

大きなロードをスタートする前に、ロード先のターゲット DB インスタンスにアクセスするすべてのアプリケーションアクティビティを停止します。特に、他のセッションでロード中のテーブルや参照するテーブルを変更する場合は、これをお勧めします。これにより、ロード中に発生する制約違反のリスクが軽減され、ロードパフォーマンスが向上します。また、ロードに関係しないプロセスによって行われた変更を失うことなく、DB インスタンスをロードの直前の時点に復元することもできます。

もちろん、これは可能でない場合や実用的ではない場合があります。アプリケーションによる DB インスタンスへのアクセスをロード前に停止できない場合は、データの可用性と整合性を確保するための手順を実行してください。必要となる具体的なステップは、実際のユースケースとサイト要件によって大きく異なります。

ステップ 3: DB スナップショットを作成する

データが含まれない新しい DB インスタンスにデータをロードする場合、このステップをスキップできます。それ以外の場合は、データロードの前後にターゲットの Amazon RDS DB インスタンスの DB スナップショットを作成することをお勧めします。Amazon RDS DB スナップショットは DB インスタンスの完全なバックアップであり、DB インスタンスを既知の状態に復元するために使用できます。DB スナップショットをスタートすると、データベースのバックアップのため DB インスタンスに対する I/O 操作が一時停止されます。

ロード直前に DB スナップショットを作成すると、必要が生じた場合にデータベースをロード前の状態に復元できます。ロード直後に DB スナップショットを作成しておくと、何らかの事故のとき

にそのスナップショットを使用すれば、データを再ロードせずに済みます。ロード後に DB スナップショットを使用して、新しいデータベースインスタンスにデータをインポートすることもできます。

次の例では、AWS CLI の create-db-snapshot コマンドを実行して、AcmeRDS インスタンスの DB スナップショットを作成し、その DB スナップショットに識別子 "preload" を指定します。

Linux、macOS、Unix の場合:

```
aws rds create-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds create-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

DB スナップショットからの復元機能を使用して、リハーサル用のテスト DB インスタンスを作成したり、ロード中に加えられた変更を元に戻すこともできます。

DB スナップショットからデータベースを復元すると、すべての DB インスタンスと同様に一意の識別子とエンドポイントを持つ新しい DB インスタンスが作成される点に留意してください。エンドポイントを変更せずに DB インスタンスを復元するには、エンドポイントを再利用できるように、まず DB インスタンスを削除します。

例えば、リハーサルや他のテスト用の DB インスタンスを作成するには、DB インスタンスに独自の識別子を指定します。この例での識別子は「AcmeRDS-2」です。この例では、AcmeRDS-2 に関連付けられたエンドポイントを使用して DB インスタンスに接続します。詳細については、「restore-db-instance-from-db-snapshot」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS-2 \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds restore-db-instance-from-db-snapshot ^
```

```
    --db-instance-identifier AcmeRDS-2 ^
    --db-snapshot-identifier preload
```

既存のエンドポイントを再利用するには、まず DB インスタンスを削除してから、復元されたデータベースに同じ識別子を指定します。詳細については、「delete-db-instance」を参照してください。

次の例でも、DB インスタンスを削除する前に最終的な DB スナップショットを取得しています。これはオプションですが推奨されます。

Linux、macOS、Unix の場合:

```
aws rds delete-db-instance \
    --db-instance-identifier AcmeRDS \
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

Windows の場合:

```
aws rds delete-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

ステップ 4 (オプション): Amazon RDS 自動バックアップをオフにする

> **Warning**
>
> ポイントインタイムリカバリを実行する必要がある場合は、自動バックアップをオフにしないでください。

自動バックアップをオフにすると、パフォーマンスが最適化します。これはデータのロードに必須ではありません。自動バックアップをオフにすると、既存のバックアップがすべて消去されます。その結果、自動バックアップをオフにすると、ポイントインタイムリカバリができなくなります。自動

バックアップを無効にしても、手動 DB スナップショットに影響が及ぶことはありません。既存のすべての手動 DB スナップショットは引き続き復元で使用可能です。

自動バックアップを無効にするとロード時間が約 25% 短縮し、ロード時に必要なストレージ容量が減少します。データが含まれない新しい DB インスタンスにデータをロードする場合、バックアップを無効にすると、ロードを簡単に高速化でき、バックアップに必要な追加のストレージを使用する必要がなくなります。しかし、状況によっては、既にデータが含まれている DB インスタンスにロードする場合もあります。その場合、バックアップを無効にするメリットと、ポイントインタイムリカバリを実行できなくなることの影響を比較検討する必要があります。

DB インスタンスでは、デフォルトで自動バックアップが有効になっています (保持期間は 1 日です)。自動バックアップを無効にするには、バックアップ保持期間を 0 に設定します。ロード後、バックアップ保持期間を 0 以外の値に設定することでバックアップを再度有効にすることができます。バックアップを有効または無効にするには、Amazon RDS で DB インスタンスをシャットダウンおよび再起動して、MySQL のログ記録を有効または無効にします。

AWS CLI の `modify-db-instance` コマンドを実行し、バックアップ保持期間を 0 に設定して変更をすぐに適用します。保持期間を 0 に設定するには DB インスタンスを再起動する必要があるため、続行前に再起動が完了するまで待っています。詳細については、「modify-db-instance」を参照してください。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --apply-immediately \
    --backup-retention-period 0
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --apply-immediately ^
    --backup-retention-period 0
```

AWS CLI の describe-db-instances コマンドで DB インスタンスのステータスを確認できます。次の例では、AcmeRDS DB インスタンスにおける DB インスタンスのステータスを表示します。

```
aws rds describe-db-instances --db-instance-identifier AcmeRDS --query "*[].
{DBInstanceStatus:DBInstanceStatus}"
```

DB インスタンスのステータスが available になったら、次のステップに進む準備が整いました。

ステップ 5: データをロードする

フラットファイルから行を読み取ってデータベーステーブルにロードするには、MySQL の LOAD DATA LOCAL INFILE ステートメントを使用します。

> **Note**
>
> フラットファイルが存在するのと同じ場所から mysql シェルを呼び出すか、LOAD DATA LOCAL INFILE の実行時にファイルの絶対パスを使用する必要があります。

次の例は、sales.txt という名前のファイルからデータベース内の Sales という名前のテーブルにデータをロードする方法を示しています。

```
mysql> LOAD DATA LOCAL INFILE 'sales.txt' INTO TABLE Sales FIELDS TERMINATED BY ' '
 ENCLOSED BY '' ESCAPED BY '\\';
Query OK, 1 row affected (0.01 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

LOAD DATA ステートメントの詳細については、「MySQL ドキュメント」の「LOAD DATA Statement」を参照してください。

ステップ 6: Amazon RDS 自動バックアップを再度オンにする

ステップ 4 で Amazon RDS 自動バックアップをオフにした場合は、ロードが完了したら、バックアップ保持期間をロード前の値に戻して、自動バックアップをオンにします。ステップ 4 で説明したように、Amazon RDS は DB インスタンスを再起動するため、短時間の停止に備えてください。

次の例では、AWS CLI の modify-db-instance コマンドを実行して、AcmeRDS DB インスタンスの自動バックアップをオンにします。保持期間は 1 日に設定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --backup-retention-period 1 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --backup-retention-period 1 ^
    --apply-immediately
```

Amazon RDS での MySQL のレプリケーションの使用

リードレプリカは通常、Amazon RDS の DB インスタンス間でレプリケーションを設定するために使用します。リードレプリカの概要については、「DB インスタンスのリードレプリカの操作」を参照してください。Amazon RDS for MariaDB でリードレプリカを操作する具体的な方法については、MySQL リードレプリカの使用 を参照してください。

RDS for MySQL のレプリケーションにはグローバルトランザクション ID (GTID) を使用できます。詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

RDS for MySQL DB インスタンスと Amazon RDS の外部にある MariaDB や MySQL インスタンスとの間でレプリケーションを設定することもできます。外部出典とのレプリケーションの設定については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

これらのどのレプリケーションオプションでも、行ベース、ステートメントベース、または混合レプリケーションが使用できます。行ベースのレプリケーションは、SQL ステートメントの結果として変更された行のみをレプリケートします。ステートメントベースのレプリケーションは、SQL ステートメント全体をレプリケートします。混合レプリケーションは、可能な場合にはステートメントレプリケーションを使用しますが、ステートメントベースのレプリケーションに対して安全でない SQL ステートメントが実行されると、行ベースのレプリケーションに切り替えます。ほとんどの場合には、混合レプリケーションをお勧めします。DB インスタンスのバイナリログ形式は、レプリケーションが行ベース、ステートメントベース、混合のいずれであるかを判断します。バイナリログ形式の設定については、「インスタンスデプロイの RDS for MySQL バイナリログの設定」を参照してください。

> **Note**
>
> Amazon RDS の外部にある MariaDB または MySQL インスタンスからデータベースをインポートしたり、これらのインスタンスにデータベースをエクスポートするようレプリケーションを設定できます。詳細については、「ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする」および「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

スナップショットから DB インスタンスを復元した後、またはポイントインタイムリカバリを実行した後で、ソースデータベースから最後に復元したバイナリログの位置を RDS コンソールで確認でき

ます。[ログとイベント] で、「binlog」と入力します。バイナリログの位置は [システムノート] の下に表示されます。

トピック

- MySQL リードレプリカの使用
- GTID ベースレプリケーションを使用する
- 外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定
- Amazon RDS for MySQL のマルチソースレプリケーションの設定

MySQL リードレプリカの使用

以下では、 MySQL の RDS でのリードレプリカの操作に関する特定の情報を確認することができます。リードレプリカと使用手順の概要については、「DB インスタンスのリードレプリカの操作」を参照してください。

MySQL リードレプリカの詳細については、以下のトピックを参照してください。

- MySQL でのレプリケーションフィルターの設定
- MySQL での遅延レプリケーションの設定
- MySQL でのリードレプリカの更新
- MySQL でのマルチ AZ リードレプリカのデプロイの使用
- RDS for MySQL でのカスケードリードレプリカの使用
- MySQL リードレプリカのレプリケーションラグのモニタリング
- MySQL リードレプリカでのレプリケーションのスタートと停止
- MySQL リードレプリカに関する問題のトラブルシューティング

MySQL でのリードレプリカの設定

MySQL DB インスタンスがレプリケーションのソースとして機能するには、ソースの DB インスタンスで自動バックアップを有効にする必要があります。これを行うには、バックアップ保持期間の値を 0 以外の値に設定します。この要件は、別のリードレプリカのソース DB インスタンスであるリードレプリカにも適用されます。自動バックアップは、MySQL の任意のバージョンを実行するリードレプリカでサポートされています。レプリケーションは、MySQL DB インスタンスのバイナリログ座標に基づいて設定できます。

次のバージョンで、グローバルなトランザクション識別子 (GTID) を使用したレプリケーションを設定できます。

- RDS for MySQL バージョン 5.7.44 以上の 5.7 バージョン
- RDS for MySQL バージョン 8.0.28 以上の 8.0 バージョン
- RDS for MySQL バージョン 8.4.3 以上の 8.4 バージョン

詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

同一リージョン内の 1 つの DB インスタンスから、最大 15 個のリードレプリカを作成できます。レプリケーションを効率的に実行するには、各リードレプリカにソース DB インスタンスと同程度のコンピューティングリソースとストレージリソースが必要です。ソースの DB インスタンスをスケールした場合は、リードレプリカもスケールする必要があります。

RDS for MySQL では、リードレプリカのカスケードをサポートしています。リードレプリカのカスケードを設定する方法については、「RDS for MySQL でのカスケードリードレプリカの使用」を参照してください。

同じソースの DB インスタンスを参照する複数のリードレプリカの作成や削除の操作は同時に実行できます。その操作を実行するには、ソースインスタンスごとに作成できるリードレプリカを 15 個に制限します。

MySQL DB インスタンスのリードレプリカは、ソース DB インスタンスよりも低い DB エンジンバージョンを使用できません。

MyISAM を使用する MySQL DB インスタンスを準備する

MySQL DB インスタンスで MyISAM などの非トランザクションエンジンを使用する場合は、以下のステップに従ってリードレプリカを設定する必要があります。このステップは、リードレプリカとデータの整合性を保つために必要です。すべてのテーブルが InnoDB などのトランザクションエンジンを使用している場合には、このステップは必要ありません。

1. ソース DB インスタンスの非トランザクションテーブルのすべてのデータ操作言語 (DML) とデータ定義言語 (DDL) の操作を停止します。SELECT 記述で実行を続けます。
2. ソース DB インスタンスでテーブルをフラッシュおよびロックします。
3. 以下のセクションのいずれかの方法を使用してリードレプリカを作成します。
4. `DescribeDBInstances` API オペレーションなどを使用して、リードレプリカの作成の進捗状況を確認します。リードレプリカが使用可能になったら、ソース DB インスタンスのテーブルのロックを解除し、通常のデータベース操作を再開します。

MySQL でのレプリケーションフィルターの設定

レプリケーションフィルターを使用して、リードレプリカでレプリケートするデータベースとテーブルを指定できます。レプリケーションフィルターは、データベースとテーブルをレプリケーションに含めることも、レプリケーションから除外することもできます。

レプリケーションフィルターの使用例は以下のとおりです。

- リードレプリカのサイズを縮小します。レプリケーションフィルタリングを使用すると、リードレプリカで必要のないデータベースとテーブルを除外できます。
- セキュリティ上の理由から、データベースとテーブルをリードレプリカから除外するため。
- 異なるリードレプリカで、特定のユースケースごとにさまざまなデータベースとテーブルを複製するため。例えば、分析やシャーディングに特定のリードレプリカを使用できます。
- 異なる AWS リージョン にリードレプリカがある DB インスタンスで、異なる AWS リージョンに異なるデータベースまたはテーブルを複製する場合。

> **Note**
>
> また、レプリケーションフィルターを使用して、インバウンドのレプリケーショントポロジでレプリカとして設定されているプライマリ MySQL DB インスタンスでレプリケートするデータベースとテーブルを指定することもできます。この設定の詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

トピック

- RDS for MySQL のレプリケーションフィルターパラメータの設定
- RDS for MySQL のレプリケーションフィルターの制限
- RDS for MySQL のレプリケーションフィルターの例
- リードレプリカのレプリケーションフィルターを表示する

RDS for MySQL のレプリケーションフィルターパラメータの設定

レプリケーションフィルターを構成するには、リードレプリカに次のレプリケーションフィルターのパラメータを設定します。

- replicate-do-db - 指定したデータベースに変更を複製します。リードレプリカに対してこのパラメータを設定すると、パラメータで指定されたデータベースのみが複製されます。
- replicate-ignore-db - 指定したデータベースに変更を複製しないでください。リードレプリカに replicate-do-db パラメータが設定されている場合、このパラメータは評価されません。
- replicate-do-table -指定されたテーブルに変更を複製します。このパラメータをリードレプリカに設定した場合、パラメータで指定したテーブルのみが複製されます。また、replicate-do-db または replicate-ignore-db パラメータが設定されている場合は、指定されたテーブルを含むデータベースを、必ずリードレプリカのレプリケーションに含めます。
- replicate-ignore-table - 指定したテーブルに変更を複製しないでください。リードレプリカに replicate-do-table パラメータが設定されている場合、このパラメータは評価されません。
- replicate-wild-do-table - 指定したデータベースおよびテーブル名のパターンに基づいてテーブルを複製します。% および _ ワイルドカードの文字がサポート対象となります。replicate-do-db または replicate-ignore-db パラメータが設定されている場合は、リードレプリカを使用して、指定したテーブルを含むデータベースをレプリケーションに含めるようにしてください。
- replicate-wild-ignore-table - 指定したデータベースおよびテーブル名のパターンに基づいてテーブルを複製しないでください。% および _ ワイルドカードの文字がサポート対象となります。リードレプリカに replicate-do-table または replicate-wild-do-table パラメータが設定されている場合、このパラメータは評価されません。

パラメータは、記載されている順序に沿って評価されます。これらのパラメータの詳細な仕組みについては、MySQL のドキュメントを参照してください。

- 一般的な情報については、Replica Server Options and Variables を参照してください。
- データベースレプリケーションのフィルターパラメータを評価する方法については、Evaluation of Database-Level Replication and Binary Logging Options を参照してください。
- テーブルレプリケーションのフィルターパラメータを評価する方法については、Evaluation of Table-Level Replication Options を参照してください。

デフォルトでは、これらの各パラメータの値は空です。各リードレプリカで、これらのパラメータを使用してレプリケーションフィルターを設定、変更、削除することができます。これらのパラメータの 1 つを設定する場合は、各フィルターを他のフィルターとコンマで区切ります。

% および _ パラメータで replicate-wild-do-table および replicate-wild-ignore-table ワイルドカードの文字を使用できます。% ワイルドカードは任意の文字数と一致し、_ ワイルドカードは 1 文字のみと一致します。

ソース DB インスタンスのバイナリログ形式は、データ変更のレコードを決定するため、レプリケーションでは重要です。binlog_format パラメータの設定により、レプリケーションが行ベースかステートメントベースかが決まります。詳細については、「インスタンスデプロイの RDS for MySQL バイナリログの設定」を参照してください。

> **Note**
>
> ソース DB インスタンスの binlog_format 設定に関係なく、すべてのデータ定義言語 (DDL) ステートメントはステートメントとして複製されます。

RDS for MySQL のレプリケーションフィルターの制限

RDS for MySQL のレプリケーションフィルターには、次の制限が適用されます。

- 各レプリケーションフィルターのパラメータには、2,000 文字といった制限があります。
- レプリケーションフィルターでは、パラメータ値としてカンマはサポートされていません。パラメータのリストでは、カンマは値の区切り文字としてのみ使用できます。例えば、ParameterValue='`a,b`'| はサポートされていませんが、ParameterValue='a,b' はサポートされています。
- MySQL --binlog-do-db とバイナリログフィルターの --binlog-ignore-db オプションはサポートされていません。
- レプリケーションフィルタリングは、XA トランザクションをサポートしていません。

  詳細については、MySQL ドキュメントの「Restrictions on XA Transactions」を参照してください。

RDS for MySQL のレプリケーションフィルターの例

リードレプリカのレプリケーションフィルタリングを構成するには、リードレプリカに関連付けられているパラメータグループのレプリケーションフィルタリングパラメータを変更します。

> **Note**
>
> デフォルトのパラメータグループを変更することはできません。リードレプリカがデフォルトのパラメータグループを使用している場合は、新しいパラメータグループを作成してリードレプリカに関連付けます。DB パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、パラメータグループのパラメータを設定できます。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。パラメータグループにパラメータを設定すると、そのパラメータグループに関連付けられているすべての DB インスタンスでパラメータ設定を使用します。パラメータグループにレプリケーションフィルターのパラメータを設定する場合は、パラメータグループがリードレプリカにのみ関連付けられていることを確認してください。ソース DB インスタンスのレプリケーションフィルターのパラメータは空のままにします。

次の例では、AWS CLI を使用してパラメータを設定します。これらの例では、CLI コマンドが完了した直後にパラメータの変更が行われるように `ApplyMethod` を `immediate` に設定しています。リードレプリカの再起動後に保留中の変更を適用する場合は、`ApplyMethod` を `pending-reboot` に設定します。

以下の例では、レプリケーションフィルターを設定します。

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Exampleレプリケーションにデータベースを含める

次の例では、レプリケーションに `mydb1` データベースと `mydb2` データベースを含めています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
```

```
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Exampleレプリケーションにテーブルを含める

次の例では、データベース mydb1 の table1 テーブルと table2 テーブルをレプリケーションに含めています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Exampleワイルドカードの文字を使用してレプリケーションにテーブルを含める

次の例では、データベース mydb の order および return で始まる名前のテーブルをレプリケーションに含めています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
```

```
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Exampleレプリケーションからデータベースを除外する

次の例では、mydb5 データベースと mydb6 データベースをレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Exampleレプリケーションからテーブルを除外する

次の例では、データベース mydb5 のテーブル table1 とデータベース mydb6 の table2 をレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Exampleワイルドカードの文字を使用したレプリケーションからテーブルを除外する

次の例では、データベース mydb7 の order および return で始まる名前のテーブルをレプリケーションから除外しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

リードレプリカのレプリケーションフィルターを表示する

リードレプリカのレプリケーションフィルターは、次の方法で表示できます。

- リードレプリカに関連付けられているパラメータグループのレプリケーションフィルタリングパラメータの設定を確認してください。

  手順については、「Amazon RDS のDB パラメータグループのパラメータ値の表示」を参照してください。
- MySQL クライアントで、リードレプリカに接続し、SHOW REPLICA STATUS ステートメントを実行します。

  出力の次のフィールドには、リードレプリカのレプリケーションフィルターが表示されます。
  - Replicate_Do_DB
  - Replicate_Ignore_DB

- `Replicate_Do_Table`
- `Replicate_Ignore_Table`
- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

これらのフィールドの詳細については、MySQL のドキュメントの Checking Replication Status を参照してください。

MySQL での遅延レプリケーションの設定

遅延レプリケーションは、災害対策用の戦略として使用できます。遅延レプリケーションでは、ソースからリードレプリカへのレプリケーションを遅延させる最小時間を秒数で指定します。障害発生時 (意図しないテーブルの削除など) には、以下のステップを実行して障害から早急に復旧します。

- 障害を起こした変更がリードレプリカに送られる前に、リードレプリカへのレプリケーションを停止します。

  レプリケーションを停止するには、mysql.rds_stop_replication ストアドプロシージャを使用します。
- レプリケーションをスタートし、レプリケーションがログファイルの場所で自動的に停止するように指定します。

  障害発生の直前の場所を指定するには、mysql.rds_start_replication_until ストアドプロシージャを使用します。
- 「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順を使用してリードレプリカを新しい出典の DB インスタンスに昇格させます。

> **Note**
>
> - RDS for MySQL 8.4 の場合、遅延レプリケーションは MySQL 8.4.3 以降でサポートされています。RDS for MySQL 8.0 の場合、遅延レプリケーションは MySQL 8.0.28 以降でサポートされています。RDS for MySQL 5.7 の場合、遅延レプリケーションは MySQL 5.7.44 以降でサポートされています。
> - 遅延レプリケーションを設定するには、ストアドプロシージャを使用します。遅延レプリケーションを AWS マネジメントコンソール、AWS CLI、または Amazon RDS API で設定することはできません。

- 次のバージョンの遅延レプリケーションの設定で、グローバルなトランザクション識別子 (GTID) に基づくレプリケーションを使用できます。
  - RDS for MySQL バージョン 5.7.44 以上の 5.7 バージョン
  - RDS for MySQL バージョン 8.0.28 以上の 8.0 バージョン
  - RDS for MySQL バージョン 8.4.3 以上の 8.4 バージョン

  GTID ベースのレプリケーションを使用する場合は、mysql.rds_start_replication_until_gtid ストアドプロシージャの代わりに、mysql.rds_start_replication_until ストアドプロシージャを実行します。GTID ベースのレプリケーションの詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

トピック

- リードレプリカ作成時の遅延レプリケーションの設定
- 既存のリードレプリカの遅延レプリケーションの変更
- リードレプリカへのレプリケーションを停止する場所の設定
- リードレプリカの昇格

リードレプリカ作成時の遅延レプリケーションの設定

DB インスタンスから今後作成するリードレプリカの遅延レプリケーションを設定するには、mysql.rds_set_configuration パラメータを指定して `target delay` ストアドプロシージャを実行します。

リードレプリカの作成時に遅延レプリケーションを設定するには

1. MySQL クライアントを使用し、マスターユーザーとして、リードレプリカのソースとなる MySQL DB インスタンスに接続します。
2. mysql.rds_set_configuration パラメータを指定して `target delay` ストアドプロシージャを実行します。

   例えば、現在の DB インスタンスから作成されるリードレプリカへのレプリケーションを少なくとも 1 時間 (3600 秒) 遅延させるように指定するには、次のストアドプロシージャを実行します。

   ```
   call mysql.rds_set_configuration('target delay', 3600);
   ```

> **Note**
>
> このストアドプロシージャを実行すると、AWS CLI または Amazon RDS API を使用して作成したリードレプリカには、指定した秒数で遅延するレプリケーションが設定されます。

既存のリードレプリカの遅延レプリケーションの変更

既存のリードレプリカの遅延レプリケーションを変更するには、mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降) ストアドプロシージャを実行します。

既存のリードレプリカの遅延レプリケーションを変更するには

1. MySQL クライアントを使用して、マスターユーザーとしてリードレプリカに接続します。
2. レプリケーションを停止するには、mysql.rds_stop_replication ストアドプロシージャを使用します。
3. mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降) ストアドプロシージャを実行します。

   例えば、リードレプリカへのレプリケーションを少なくとも 1 時間 (3600 秒) 遅延させるように指定するには、次のストアドプロシージャを実行します。

   ```
   call mysql.rds_set_source_delay(3600);
   ```
4. mysql.rds_start_replication ストアドプロシージャを使用してレプリケーションをスタートします。

リードレプリカへのレプリケーションを停止する場所の設定

リードレプリカへのレプリケーションが停止したら、mysql.rds_start_replication_until ストアドプロシージャを使用してレプリケーションをスタートしてバイナリログファイルの指定した位置で停止できます。

リードレプリカへのレプリケーションをスタートして指定の位置でレプリケーションを停止するには

1. MySQL クライアントを使用して、マスターユーザーとしてソース MySQL DB インスタンスに接続します。

2. mysql.rds_start_replication_until ストアドプロシージャを実行します。

   次の例では、レプリケーションをスタートし、120 バイナリログファイルの場所 mysql-bin-changelog.000777 に達するまで変更をレプリケートします。災害対策シナリオでは、場所 120 は災害発生直前の時点として想定されます。

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

停止ポイントに達すると、レプリケーションは自動的に停止します。RDS イベントとして、Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure が生成されます。

リードレプリカの昇格

レプリケーションが停止したら、災害対策シナリオでリードレプリカを新しいソース DB インスタンスに昇格させます。リードレプリカの昇格については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

MySQL でのリードレプリカの更新

リードレプリカは読み取りクエリをサポートするように設計されていますが、ときどき更新が必要になることがあります。例えば、インデックスを追加して、レプリカにアクセスする特定のタイプのクエリを最適化する必要が生じることがあります。

ただし、リードレプリカの DB パラメータグループで read_only パラメータを 0 に設定することにより、更新を有効にすることはできますが、リードレプリカとソースの DB インスタンスの互換性がなくなると問題が発生する可能性があるため、それは推奨できません。メンテナンス作業には、ブルー/グリーンデプロイを使用することをおすすめします。詳細については、「データベースの更新にブルー/グリーンデプロイを使用する」を参照してください。

リードレプリカで読み取り専用を無効にする場合は、できるだけ早く read_only パラメータの値を 1 に戻してください。

MySQL でのマルチ AZ リードレプリカのデプロイの使用

リードレプリカは、シングル AZ DB インスタンス配置からもマルチ AZ DB インスタンス配置からも作成できます。重要なデータの耐久性の高いと可用性を高めるにはマルチ AZ 配置を使用します

が、読み取り専用クエリを処理するためにマルチ AZ セカンダリを使用することはできません。代わりに、トラフィックの多いマルチ AZ DB インスタンスのリードレプリカを作成して、読み取り専用クエリをオフロードできます。マルチ AZ 配置のソースのインスタンスがセカンダリにフェイルオーバーすると、関連付けられているすべてのリードレプリカが (プライマリから) セカンダリをレプリケーションのソースとして使用するように自動的に切り替わります。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

リードレプリカは、マルチ AZ DB インスタンスとして作成できます。Amazon RDS では、レプリカのフェイルオーバーをサポートするため、別のアベイラビリティーゾーンにレプリカのスタンバイを作成します。リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。

RDS for MySQL でのカスケードリードレプリカの使用

RDS for MySQL では、リードレプリカのカスケードをサポートしています。カスケードリードレプリカにより、ソースの RDS for MySQL DB インスタンスにオーバーヘッドを追加せずに読み取りをスケーリングできます。

カスケードリードレプリカを使用すると、RDS for MySQL DB インスタンスは、チェーン内の最初のリードレプリカにデータを送信します。その後、そのリードレプリカは、チェーン内の 2 番目のレプリカにデータを送信し、その動作が順に続いていきます。その結果、チェーン内のすべてのリードレプリカに RDS for MySQL DB インスタンスの更新が送信されますが、ソース DB インスタンスでのオーバーヘッドは発生しません。

ソースの RDS for MySQL DB インスタンスから、チェーン内にリードレプリカを 3 層まで作成できます。例えば、RDS for MySQL DB インスタンス、`mysql-main` があるとします。以下の操作を行うことができます。

- `mysql-main` で開始し、チェーン内に最初のリードレプリカ、`read-replica-1` を作成します。
- 次に、`read-replica-1` で、チェーン内に次のリードレプリカ、`read-replica-2` を作成します。
- 最後に、`read-replica-2` で、チェーン内に 3 番目のリードレプリカ、`read-replica-3` を作成します。

`mysql-main` の層では、この 3 番目のカスケードリードレプリカに続く、別のリードレプリカを作成することはできません。一連の完全なインスタンス (RDS for MySQL のソース DB インスタンスから、この層の最後のカスケードリードレプリカまで) は、最大 4 つの DB インスタンスで構成できます。

リードレプリカのカスケードを設定するには、RDS for MySQL DB インスタンスで自動バックアップを有効にします。リードレプリカで自動バックアップを有効にするには、まずリードレプリカを作成し、次に自動バックアップを有効にするようにリードレプリカを変更します。詳細については、「リードレプリカの作成」を参照してください。

他のリードレプリカと同様に、カスケードの一部となっているリードレプリカを昇格できます。リードレプリカのチェーン内でリードレプリカを昇格させると、そのレプリカはチェーンから削除されます。例えば、mysql-main DB インスタンスのワークロードの一部を、経理部のみが使用する新しいインスタンスに移動するとします。この例では、3 つのリードレプリカから成るチェーンがあると仮定し、read-replica-2 を昇格させることにします。チェーンは以下のような影響を受けます。

- 昇格する read-replica-2 は、レプリケーションチェーンから削除されます。
  - このリードレプリカは、完全な読み取り/書き込み DB インスタンスになります。
  - 昇格前と同じように、read-replica-3 へのレプリケーションを継続します。
- mysql-main は、read-replica-1 へのレプリケーションを継続します。

リードレプリカの昇格についての詳細は、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

MySQL リードレプリカのレプリケーションラグのモニタリング

MySQL のリードレプリカでは、Amazon CloudWatch で Amazon RDS の ReplicaLag メトリクスを確認することでレプリケーションの遅延をモニタリングできます。ReplicaLag メトリクスには、Seconds_Behind_Master コマンドの SHOW REPLICA STATUS フィールドの値が報告されます。

MySQL のレプリケーション遅延の一般的な原因は以下のとおりです。

- ネットワークが停止している。
- リードレプリカとインデックスが異なるテーブルに書き込んでいる。リードレプリカで、read_only パラメータが 0 に設定されている場合、リードレプリカとソースの DB インスタンスの互換性がなくなると、レプリケーションが中断する可能性があります。リードレプリカのメンテナンスタスクを実行したら、read_only パラメータは 1 に戻すことをお勧めします。
- MyISAM などの非トランザクションストレージエンジンを使用している。レプリケーションは、MySQL 上の InnoDB ストレージエンジンでのみサポートされます。

ReplicaLag メトリックが 0 に達すると、レプリカがソース DB インスタンスに追いついています。ReplicaLag メトリクスにより -1 が返された場合、レプリケーションは現在アクティブではありません。ReplicaLag = -1 は Seconds_Behind_Master = NULL と同等です。

MySQL リードレプリカでのレプリケーションのスタートと停止

システムのストアドプロシージャ mysql.rds_stop_replication および mysql.rds_start_replication を呼び出すことにより、Amazon RDS DB インスタンスでレプリケーションプロセスを停止して再スタートすることができます。これは、大きいインデックスの作成など、長時間実行されている操作の 2 つの Amazon RDS インスタンス間でレプリケーションするときに実行できます。レプリケーションは、データベースをインポートまたはエクスポートするときに停止してスタートする必要もあります。詳細については、「ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする」および「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

レプリケーションを手動で停止するかレプリケーションエラーで停止してから連続して 30 日を超えると、Amazon RDS はソースの DB インスタンスとすべてのリードレプリカの間のレプリケーションを終了します。これは、ソース DB インスタンスでの所要ストレージの増大と長期間のフェイルオーバーを防ぐためです。リードレプリカの DB インスタンスは引き続き使用できます。ただし、レプリケーションが終了されるとリードレプリカに必要なバイナリログがソースの DB インスタンスから削除されるため、レプリケーションを再開することはできません。レプリケーションを再度行うには、ソースの DB インスタンスの新しいリードレプリカを作成します。

MySQL リードレプリカに関する問題のトラブルシューティング

MySQL DB インスタンスでは、リードレプリカとソース DB インスタンスの間のレプリケーションエラーやデータの不整合 (またはその両方) がリードレプリカで発生する場合があります。この問題は、リードレプリカまたはソースの DB インスタンスの障害時に、一部のバイナリログ (binlog) イベントまたは InnoDB redo ログがフラッシュされない場合に発生します。その場合、リードレプリカを手動で削除して作成し直します。次のパラメータ値 (sync_binlog=1、innodb_flush_log_at_trx_commit=1) を設定することで、これが発生する可能性を減らすことができます。これらの設定によりパフォーマンスが低下することがあるため、本稼働環境で変更を実装する前に影響をテストしてください。

> **Warning**
>
> ソース DB インスタンスに関連付けられたパラメータグループでは、パラメータ値 sync_binlog=1 および innodb_flush_log_at_trx_commit=1 を保持することをお勧めします。これらのパラメータは動的です。これらの設定を使用しない場合、ソース DB イ

ンスタンスで再起動する可能性のある操作を実行する前に、これらの値を一時的に設定することをお勧めします。該当する操作には、再起動、フェイルオーバーによる再起動、データベースバージョンのアップグレード、DB インスタンスクラスまたはそのストレージの変更が含まれますが、これらに限定されません。同じ推奨事項が、ソース DB インスタンスの新しいリードレプリカを作成する場合にも適用されます。
このガイダンスに従わない場合、リードレプリカとソースの DB インスタンス間のレプリケーションエラーやデータの不整合 (またはその両方) がリードレプリカで発生するリスクが高くなります。

MySQL のレプリケーションテクノロジーは非同期です。非同期であるため、ソースの DB インスタンスの `BinLogDiskUsage` やリードレプリカの `ReplicaLag` が増加する場合があります。例えば、ソース DB インスタンスへの大量の書き込みオペレーションは並行して実行できます。一方、リードレプリカへの書き込みオペレーションは単一の I/O スレッドでシリアルで行われるため、ソースのインスタンスとリードレプリカの間で遅延が発生する場合があります。読み取り専用レプリカの詳細については、MySQL ドキュメントの「レプリケーション実装の詳細」を参照してください。

ソースの DB インスタンスに対する更新とそれに続くリードレプリカに対する更新の間の遅延を低減するには、次のいくつかの方法があります。

- ストレージサイズと DB インスタンスクラスがソース DB インスタンスと同程度となるようにリードレプリカのサイズを決定します。
- ソース DB インスタンスとリードレプリカにより使用される DB パラメータグループのパラメータ設定に互換性を確保します。詳細と例については、このセクションの後方にある `max_allowed_packet` パラメータの説明を参照してください。

Amazon RDS は、リードレプリカのレプリケーションの状態をモニタリングし、何らかの理由でレプリケーションが停止した場合はリードレプリカのインスタンスの `Replication State` フィールドを `Error` に更新します。これには、リードレプリカで実行された DML クエリがソースの DB インスタンスで行われた更新と競合した場合などがあります。

`Replication Error`フィールドを参照することで、MySQL エンジンによりスローされた関連するエラーの詳細を確認できます。リードレプリカのステータスを示すイベントが生成されます (RDS-EVENT-0045、RDS-EVENT-0046、RDS-EVENT-0047 など)。イベントについてとイベントへのサブスクライブの詳細については、「Amazon RDS イベント通知の操作」を参照してください。MySQL のエラーメッセージが返された場合は、MySQL のエラーメッセージのドキュメントでエラー番号を確認してください。

レプリケーションエラーを引き起こす一般的な問題は、リードレプリカの max_allowed_packet パラメータの値がソース DB インスタンスの max_allowed_packet パラメータより小さいことです。max_allowed_packet パラメータは、DB パラメータグループで設定できるカスタムパラメータです。max_allowed_packet は、データベースで実行できる DML コードの最大サイズを指定するために使用します。場合によっては、リードレプリカに関連付けらている DB パラメータグループの max_allowed_packet の値が、ソース DB インスタンスに関連付けられている DB パラメータグループの max_allowed_packet の値より小さいことがあります。このような場合、レプリケーションプロセスは Packet bigger than 'max_allowed_packet' bytes エラーをスローし、レプリケーションを停止する可能性があります。このエラーを修正するには、ソース DB インスタンスとリードレプリカの DB パラメータグループで max_allowed_packet パラメータに同じ値を使用します。

レプリケーションエラーを引き起こす可能性があります他の一般的な状況は次のとおりです。

- リードレプリカのテーブルに書き込んでいる。場合によっては、リードレプリカにソースの DB インスタンスのインデックスとは異なるインデックスを作成することがあります。その場合は、read_only パラメータを 0 に設定してインデックスを作成してください。リードレプリカのテーブルに書き込む場合、リードレプリカとソースの DB インスタンスの互換性がなくなると、レプリケーションが中断する可能性があります。リードレプリカのメンテナンスタスクを実行したら、read_only パラメータは 1 に戻すことをお勧めします。
- MyISAM などの非トランザクションストレージエンジンを使用している。リードレプリカにはトランザクションストレージエンジンが必要です。レプリケーションは、MySQL 上の InnoDB ストレージエンジンでのみサポートされます。
- SYSDATE() など、安全でない非決定的クエリを使用している。詳細については、「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください。

エラーを安全にスキップできると判断した場合、「RDS for MySQL の現在のレプリケーションエラーのスキップ」セクションで説明されているステップに従うことができます。そうでない場合は、まずリードレプリカを削除します。その上で、同じ DB インスタンス識別子を使用してインスタンスを作成することで、エンドポイントを前のリードレプリカと同じままにすることができます。レプリケーションエラーが解決すると、[Replication State] は [レプリケーション中] に変化します。

レコードの欠落によるレプリケーション障害のトラブルシューティング

バイナリログレプリケーションで Amazon RDS for MySQL または MariaDB を使用する場合、リードレプリカのレコードがないためにプロセスが停止するレプリケーションエラーが発生する可能性があります。このセクションでは、これらの問題を診断して解決する方法について説明します。

一般的な原因

ソースとレプリカ間のデータの不整合は、次のいずれかのシナリオが原因で発生する可能性があります。

- レプリカでの手動データ変更
- sql_replica_skip_counter (MySQL 8.0.26 より前では sql_slave_skip_counter) を使用してスキップされたトランザクション
- 移行ツールを使用した初期データ同期中にレプリカで行が欠落

問題の特定

この問題の一般的な指標は、Can't find record メッセージを伴う Error 1032 です。次のエラーメッセージがレプリカのエラーログに表示されます。

```
2025-11-04T21:24:11.038899Z 566 [ERROR] [MY-010584] [Repl] Replica SQL for channel '':
Worker 1 failed executing transaction 'ANONYMOUS' at source log
mysql-bin-changelog.031523, end_log_pos 899; Could not execute Update_rows event on
table test_replication.users; Can't find record in 'users', Error_code: 1032; handler
error HA_ERR_KEY_NOT_FOUND; the event's source log mysql-bin-changelog.031523,
end_log_pos 899, Error_code: MY-001032
```

> **Note**
>
> エラーログの形式は、MySQL エンジンと MariaDB エンジンで異なります。ログを取得する方法については、「データベースログファイルの表示とリスト化」を参照してください。

問題の診断

レプリケーションの失敗を診断するには、次の手順を実行します。

- レプリカのエラーログを確認して、レプリケーションが停止した場所を特定します。検索キーワードとして MY-010584 を使用できます。
- レプリカで SHOW REPLICA STATUS\G (MySQL 8.0.22 より前では SHOW SLAVE STATUS\G) を実行し、Last_Error 列を確認します。

```
Last_Error: Coordinator stopped because there were error(s) in the worker(s).
```

```
The most recent failure being: Worker 1 failed executing transaction 'ANONYMOUS'
at source log mysql-bin-changelog.031523, end_log_pos 899. See error log and/or
performance_schema.replication_applier_status_by_worker table for more details
about this failure or others, if any.
```

- バイナリログファイルの名前と、レプリカインスタンスでレプリケーションプロセスが停止した場所を確認します。この情報は、レプリカインスタンスのエラーログにあります。

バイナリログの分析

特定のトランザクションでのレプリケーションの失敗を調査するには、mysqlbinlog ユーティリティを使用してバイナリログファイルを調べることができます。例えば、バイナリログファイル mysql-bin-changelog.031523 の end_log_pos 位置 899 でレプリケーションが失敗した場合、この特定の場所を分析して失敗の原因を特定できます。

> **Note**
>
> end_log_pos 値は、失敗したイベントの終了位置を示します。デコードされたバイナリログ出力で、end_log_pos がこの値と一致するイベントを探します。

mysqlbinlog ユーティリティを使用して、関連するバイナリログをダウンロードして調べます。プライマリインスタンスのエンドポイントを使用します。

```
$ mysqlbinlog --read-from-remote-server \
--host=<rds-endpoint> \
--port=3306 \
--user=<username> \
--password=<password> \
--base64-output=DECODE-ROWS -vv \
mysql-bin-changelog.031523 > decoded-binlog-file-name
```

> **Note**
>
> mysqlbinlog ユーティリティは、バイナリログの内容の読み取りと解釈に役立つネイティブ MySQL ツールです。Amazon RDS のバイナリログへのアクセス方法の詳細については、「MySQL バイナリログにアクセスする」を参照してください。

ライターインスタンスのバイナリログには、ライターインスタンスが test_replication.users テーブル内の id=1 のレコードを更新しようとしたことが示されています。

```
#251104 21:13:13 server id 680788197  end_log_pos 899 CRC32 0x89b1e255     Update_rows:
 table id 101 flags: STMT_END_F
### UPDATE `test_replication`.`users`
### WHERE
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='Test User' /* VARSTRING(400) meta=400 nullable=1 is_null=0 */
###   @3='updated4@example.com' /* VARSTRING(400) meta=400 nullable=1 is_null=0 */
###   @4=1762288260 /* TIMESTAMP(0) meta=0 nullable=1 is_null=0 */
### SET
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='Test User' /* VARSTRING(400) meta=400 nullable=1 is_null=0 */
###   @3='updated5@example.com' /* VARSTRING(400) meta=400 nullable=1 is_null=0 */
###   @4=1762288260 /* TIMESTAMP(0) meta=0 nullable=1 is_null=0 */
# at 899
```

レプリカのエラーログには、ターゲットレコードが存在しないため、レプリカが UPDATE ステートメントを実行できなかったことが示されています。

```
Can't find record in 'users', Error_code: 1032; handler error HA_ERR_KEY_NOT_FOUND
```

行が欠落していることを確認するには、レプリカに特定の行が存在するかどうかを確認します。

```
SELECT 1 FROM test_replication.users WHERE id=1;
Empty set (0.00 sec)
```

この出力は、ターゲット行がレプリカに存在しないため、ソース DB インスタンスからの UPDATE オペレーションがレプリケートに失敗したことを確認します。

レプリケーションの失敗を回避する

レプリケーションの不整合を避けるため、次のベストプラクティスをお勧めします。

- GTID ベースのレプリケーションを使用します。詳細については、「GTID ベースレプリケーションを使用する」を参照してください。
- プライマリインスタンスでは、binlog_format に ROW を使用します。binlog_format を MIXED にすると、レプリケーションの不整合が検出されないまま見過ごされる可能性があります。

- 意図しない行の変更を防ぐために、レプリカインスタンスで `read_only` を 1 に設定します。
- レプリカインスタンスで `innodb_flush_log_at_trx_commit` を 1 に設定します。
- プライマリインスタンスで `sync_binlog` を 1 に設定します。
- プライマリとレプリカ間のテーブルスキーマの同等性を確認します。
- レプリケートされたテーブルにプライマリキーがあることを確認します。

GTID ベースレプリケーションを使用する

以下では、複数の Amazon RDS for MySQL DB インスタンス間において、バイナリログ (binlog) レプリケーションでグローバルトランザクション ID (GTID) を使用する方法について説明します。

binlog レプリケーションを使用する際に MySQL での GTID ベースのレプリケーションに慣れていない場合は、MySQL ドキュメントの「Replication with global transaction identifiers」を参照してください。

GTID ベースのレプリケーションは、次のバージョンでサポートされています。

- すべての RDS for MySQL 8.4 バージョン
- すべての RDS for MySQL 8.0 バージョン
- すべての RDS for MySQL 5.7 バージョン

レプリケーション設定のすべての MySQL DB インスタンスがこのバージョン要件を満たしている必要があります。

トピック

- グローバルトランザクション ID (GTID) の概要
- GTID ベースレプリケーションのパラメータ
- RDS for MySQL の新しいリードレプリカの GTID ベースのレプリケーションを有効にする
- RDS for MySQL の既存のリードレプリカの GTID ベースのレプリケーションを有効にする
- リードレプリカを持つ MySQL DB インスタンスの GTID ベースレプリケーションを無効にする

グローバルトランザクション ID (GTID) の概要

グローバルトランザクション ID (GTID) はコミットされた MySQL トランザクションに対して生成される一意の ID です。GTID を使用することで、簡単に binlog をレプリケーションおよびトラブルシューティングできるようになります。

MySQL では、binlog レプリケーションに 2 種類のトランザクションを使用します。

- GTID トランザクション - GTID によって識別されるトランザクション。
- 匿名トランザクション - GTID が割り当てられていないトランザクション。

レプリケーション設定では、GTID はすべての DB インスタンスで一意です。GTID を使用すると、ログファイルの位置を参照する必要がないため、GTID はレプリケーション設定を簡素化します。GTID はまた、レプリケートされたトランザクションを追跡し、出典インスタンスとレプリカが一致しているかどうかの判断を容易にします。

GTID ベースのレプリケーションを使用して、RDS for MySQL リードレプリカでデータをレプリケートできます。新しいリードレプリカの作成時に GTID ベースのレプリケーションを設定するか、GTID ベースのレプリケーションを使用するように既存のリードレプリカを変換することができます。

また、RDS for MySQL を使用し、遅延レプリケーション設定で GTID ベースのレプリケーションを使用することもできます。詳細については、「MySQL での遅延レプリケーションの設定」を参照してください。

GTID ベースレプリケーションのパラメータ

以下のパラメータを使用して、GTID ベースレプリケーションを設定します。

Parameter	有効な値	説明
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF` は新しいトランザクションが匿名トランザクション (つまり GTID を持たない) であることを指定し、トランザクションは匿名でレプリケートされる必要があります。 `OFF_PERMISSIVE` は新しいトランザクションが匿名トランザクションであることを指定しま

Parameter	有効な値	説明
		すが、すべてのトランザクションをレプリケートできます。 ON_PERMISSIVE は新しいトランザクションが GTID トランザクションであることを指定しますが、すべてのトランザクションをレプリケートできます。 ON は新しいトランザクションが GTID トランザクションであることを指定し、トランザクションは複製される GTID トランザクションでなければなりません。
enforce_gtid_consistency	OFF, ON, WARN	OFF はトランザクションが GTID の整合性に違反することを許可します。 ON はトランザクションが GTID の整合性に違反することを防ぎます。 WARN は、トランザクションが GTID の整合性に違反することを許可しますが、違反が発生すると警告を生成します。

> **Note**
>
> AWS マネジメントコンソール では、gtid_mode パラメータは gtid-mode のように表示されます。

GTID ベースのレプリケーションでは、DB インスタンスまたはリードレプリカのパラメータグループでこれらの設定を使用します。

- ON と ON_PERMISSIVE は、RDS DB インスタンスからの送信レプリケーションにのみ適用されます。いずれの値でも、RDS DB インスタンスは、レプリケートされるトランザクションに GTID を使用します。ON の場合は、ターゲットデータベースも GTID ベースのレプリケーションを使用し

ます。ON_PERMISSIVE の場合、GTID ベースのレプリケーションは、ターゲットデータベースではオプションになります。

- OFF_PERMISSIVE が設定された場合、これは、RDS DB インスタンスがソースデータベースからの受信レプリケーションを受け入れることができることを意味します。これは、ソースデータベースで GTID ベースのレプリケーションが使用されているかどうかにかかわらず実行できます。
- OFF が設定された場合、これは、RDS DB インスタンスが、GTID ベースのレプリケーションを使用しないソースデータベースからの受信レプリケーションのみを受け入れることができることを意味します。

パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

RDS for MySQL の新しいリードレプリカの GTID ベースのレプリケーションを有効にする

RDS for MySQL DB インスタンスで GTID ベースのレプリケーションが有効になっている場合、GTID ベースのレプリケーションは DB インスタンスのリードレプリカに対して自動的に設定されます。

新しいリードレプリカの GTID ベースのレプリケーションを有効にするには

1. DB インスタンスに関連付けられたパラメータグループに次のパラメータ設定があることを確認します。

   - gtid_mode - ON または ON_PERMISSIVE
   - enforce_gtid_consistency – ON

   パラメータグループを使用して設定パラメータの設定の詳細については、「Amazon RDS のパラメータグループ」を参照してください。
2. DB インスタンスのパラメータグループを変更した場合は、DB インスタンスを再起動します。方法の詳細については、 DB インスタンスの再起動 を参照してください。
3. DB インスタンスの 1 つまたは複数のリードレプリカを作成します。方法の詳細については、リードレプリカの作成 を参照してください。

Amazon RDS は、MASTER_AUTO_POSITION を使用して MySQL DB インスタンスとリードレプリカ間で GTID ベースのレプリケーションの接続の確立を試みます。試行が失敗した場

合、Amazon RDS はリードレプリカを使用してレプリケーションにログファイルの位置を使用します。MASTER_AUTO_POSITION の詳細については、MySQL ドキュメントの「GTID 自動配置」を参照してください。

RDS for MySQL の既存のリードレプリカの GTID ベースのレプリケーションを有効にする

GTID ベースのレプリケーションを使用しないリードレプリカを持つ既存の MySQL DB インスタンスでは、DB インスタンスとリードレプリカ間で GTID ベースのレプリケーションを設定できます。

既存のリードレプリカの GTID ベースレプリケーションを有効にするには

1. DB インスタンスまたはリードレプリカで RDS for MySQL バージョン 8.0.26 以下の 8.0 バージョンを使用している場合は、DB インスタンスまたはリードレプリカを8.0.26 以降の MySQL 8.0 バージョンにアップグレードします。すべての RDS for MySQL 8.4 バージョンおよび 5.7 バージョンで GTID ベースのレプリケーションをサポートしています。

   詳細については、「RDS for MySQL DB エンジンのアップグレード」を参照してください。
2. (オプション) GTID パラメータをリセットし、DB インスタンスとリードレプリカの動作をテストします。

   a. DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループで enforce_gtid_consistency パラメータが WARN に設定されていることを確認します。

      パラメータグループを使用して設定パラメータの設定の詳細については、「Amazon RDS のパラメータグループ」を参照してください。

   b. DB インスタンスのパラメータグループを変更した場合は、DB インスタンスを再起動します。リードレプリカのパラメータグループを変更した場合は、リードレプリカを再起動します。

      詳細については、「 DB インスタンスの再起動」を参照してください。

   c. DB インスタンスとリードレプリカを通常のワークロードで実行し、ログファイルをモニタリングします。

      GTID と互換性のないトランザクションに関する警告が表示された場合は、GTID 互換の機能のみを使用するようアプリケーションを調整してください。以下のステップに進む前に、DB インスタンスが GTID と互換性のないトランザクションに関する警告を生成していないことを確認してください。

3. リードレプリカがすべてを処理するまで匿名トランザクションを許可する GTID ベースのレプリケーションの GTID パラメータをリセットします。

   a. DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループに、以下のパラメータ設定が含まれていることを確認します。

      - `gtid_mode` – `ON_PERMISSIVE`
      - `enforce_gtid_consistency` – `ON`

   b. DB インスタンスのパラメータグループを変更した場合は、DB インスタンスを再起動します。リードレプリカのパラメータグループを変更した場合は、リードレプリカを再起動します。

4. すべての匿名トランザクションがレプリケートされるまで待ちます。これらがレプリケートされていることを確認するには、次の作業を行います。

   a. 出典 DB インスタンスで、次のステートメントを実行します。

      MySQL 8.4

      ```
      SHOW BINARY LOG STATUS;
      ```

      MySQL 5.7 および 8.0

      ```
      SHOW MASTER STATUS;
      ```

      [File] および [Position] 列の値に注意してください。

   b. 各リードレプリカで前述のステップの出典インスタンスからのファイルと位置情報を使用し、次のクエリを実行します。

      ```
      SELECT MASTER_POS_WAIT('file', position);
      ```

      例えば、ファイル名が `mysql-bin-changelog.000031` で、場所が `107` の場合、以下のステートメントを実行します。

      ```
      SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
      ```

リードレプリカが指定された位置を過ぎている場合、クエリは直ちに返ります。変更しなかった場合、その関数は待ちます。すべてのリードレプリカに対してクエリが返るときは、以下のステップに進みます。

5. GTID ベースのレプリケーションの GTID パラメータのみをリセットします。

   a. DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループに、以下のパラメータ設定が含まれていることを確認します。

      - `gtid_mode` – `ON`
      - `enforce_gtid_consistency` – `ON`

   b. DB インスタンスと各リードレプリカを再起動します。

6. 各リードレプリカで、以下の手順を実行します。

   MySQL 8.4 以降のメジャーバージョン

   ```
   CALL mysql.rds_set_source_auto_position(1);
   ```

   MySQL 8.0 以前のメジャーバージョン

   ```
   CALL mysql.rds_set_master_auto_position(1);
   ```

リードレプリカを持つ MySQL DB インスタンスの GTID ベースレプリケーションを無効にする

リードレプリカを含む MySQL DB インスタンスです。

リードレプリカを含む MySQL DB インスタンスに対して の GTID ベースのレプリケーションを無効にするには

1. 各リードレプリカで、次の手順を実行します。

   MySQL 8.4 以降のメジャーバージョン

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   MySQL 8.0 以前のメジャーバージョン

```
CALL mysql.rds_set_master_auto_position(0);
```

2. gtid_mode を ON_PERMISSIVE にリセットします。

   a. MySQL DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループで gtid_mode が ON_PERMISSIVE になっていることを確認します。

      パラメータグループを使用して設定パラメータの設定の詳細については、「Amazon RDS のパラメータグループ」を参照してください。

   b. MySQL DB インスタンスと各リードレプリカを再起動します。再起動の詳細については、「DB インスタンスの再起動」を参照してください。

3. gtid_mode を OFF_PERMISSIVE にリセットします。

   a. MySQL DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループで gtid_mode が OFF_PERMISSIVE になっていることを確認します。

   b. MySQL DB インスタンスと各リードレプリカを再起動します。

4. すべての GTID トランザクションがすべてのリードレプリカに適用されるまで待ちます。適用されたことを確認するには、次の手順を実行します。

   a. MySQL DB インスタンスで、次のコマンドを実行します。

      MySQL 8.4

      ```
      SHOW BINARY LOG STATUS
      ```

      MySQL 5.7 および 8.0

      ```
      SHOW MASTER STATUS
      ```

      出力は次のようになります。

      ```
      File                          Position
      ------------------------------------
      mysql-bin-changelog.000031      107
      ------------------------------------
      ```

      出力のファイルと位置に注意してください。

b. リードレプリカごとに、前のステップで得たソースインスタンスのファイルと位置の情報を使用して、次のクエリを実行します。

MySQL 8.4 および MySQL 8.0.26 以降の MySQL 8.0 バージョン

```
SELECT SOURCE_POS_WAIT('file', position);
```

MySQL 5.7*

```
SELECT MASTER_POS_WAIT('file', position);
```

例えば、ファイル名が mysql-bin-changelog.000031 で、場所が 107 の場合は、次のステートメントを実行します。

MySQL 8.4 および MySQL 8.0.26 以降の MySQL 8.0 バージョン

```
SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
```

MySQL 5.7*

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

5. GTID パラメータをリセットして、GTID ベースのレプリケーションを無効にします。

a. MySQL DB インスタンスおよび各リードレプリカに関連付けられたパラメータグループに、以下のパラメータ設定が含まれていることを確認します。

- gtid_mode – OFF
- enforce_gtid_consistency – OFF

b. MySQL DB インスタンスと各リードレプリカを再起動します。

外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

バイナリログファイルのレプリケーションを使用して、RDS for MySQLまたは MariaDB DB インスタンスと Amazon RDS の外部にある MySQL または MariaDB インスタンスとの間でレプリケーションを設定できます。

トピック

- 開始する前に
- 外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

開始する前に

レプリケートされたトランザクションのバイナリログファイルの位置を使用して、レプリケーションを設定できます。

Amazon RDS DB インスタンスでレプリケーションをスタートするために必要なアクセス権限は限定されており、Amazon RDS マスターユーザーは利用できません。そのため、Amazon RDS の mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) または mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)、および mysql.rds_start_replication コマンドを使用して、ライブデータベースと Amazon RDS のデータベースのレプリケーションを設定する必要があります。

MySQL または MariaDB データベースにバイナリログ形式を設定するには、`binlog_format` パラメータを更新します。DB インスタンスがデフォルト DB インスタンスパラメータグループを使用している場合、新しい DB パラメータグループを作成して `binlog_format` パラメータを変更します。MariaDB および MySQL 8.0 以前のバージョンでは、`binlog_format` のデフォルトは `MIXED` です。ただし、特定バイナリログ (binlog) 形式が必要な場合は `binlog_format` を `ROW` または `STATEMENT` に設定する必要もあります。変更を適用するには、DB インスタンスを再起動します。MySQL 8.4 以降のバージョンでは、`binlog_format` のデフォルトは `ROW` です。

`binlog_format` パラメータの設定については、インスタンスデプロイの RDS for MySQL バイナリログの設定 を参照してください。さまざまな MySQL レプリケーションタイプの詳細については、MySQL ドキュメントの「ステートメントベースおよび行ベースレプリケーションのメリットとデメリット」を参照してください。

外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定

Amazon RDS で外部ソースインスタンスとレプリカをセットアップする場合は、次のガイドラインに従ってください。

- レプリカである Amazon RDS DB インスタンスのフェイルオーバーのイベントをモニタリングします。フェイルオーバーが発生すると、レプリカである DB インスタンスが、新しいホスト上に別

のネットワークアドレスで再作成されます。フェイルオーバーイベントをモニタリングする方法については、「Amazon RDS イベント通知の操作」を参照してください。

- ソースインスタンスのバイナリログがレプリカに適用されたことを確認するまで、これらのバイナリログを保持します。このメンテナンスによって、障害発生時にソースインスタンスを復元できます。
- Amazon RDS DB インスタンスにある自動バックアップを有効にします。自動バックアップを有効にすると、ソースインスタンスとレプリカを再同期する必要がある場合に、特定の時点にレプリカを復元できます。バックアップと特定の時点への復元の詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

外部ソースインスタンスを使用してバイナリログファイルのレプリケーションを設定するには

1. ソース MySQL または MariaDB インスタンスを読み取り専用にします。

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. ソース MySQL または MariaDB インスタンスで SHOW MASTER STATUS コマンドを実行して、binlog の場所を特定します。

   次の例のような出力を受け取ります。

```
File                          Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. mysqldump を使用して、外部のインスタンスから Amazon RDS DB インスタンスにデータベースをコピーします。非常に大きなデータベースでは、「ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする」の手順を使用することが必要になる場合があります。

   Linux、macOS、Unix の場合:

```
mysqldump --databases database_name \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
```

```
    -plocal_password | mysql \
        --host=hostname \
        --port=3306 \
        -u RDS_user_name \
        -pRDS_password
```

Windows の場合:

```
mysqldump --databases database_name ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -plocal_password | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u RDS_user_name ^
        -pRDS_password
```

> **Note**
>
> -p オプションと入力するパスワードの間にスペースがないことを確認します。

Amazon RDS DB インスタンスに接続するためのホスト名、ユーザー名、ポート、およびパスワードを指定するには、--host コマンドで --user (-u)、--port、-p および mysql オプションを使用します。ホスト名は、Amazon RDS DB インスタンスのエンドポイントのドメインネームサービス (DNS) 名 (例: myinstance.123456789012.us-east-1.rds.amazonaws.com) です。エンドポイントの値は、AWS マネジメントコンソールのインスタンスの詳細で確認できます。

4. もう一度ソース MySQL または MariaDB インスタンスを書き込み可能にします。

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

レプリケーションで使用するバックアップの作成の詳細については、MySQL ドキュメントを参照してください。

5. AWS マネジメントコンソール で、外部のデータベースをホストするサーバーの IP アドレスを、Amazon RDS DB インスタンスの仮想プライベートクラウド (VPC) のセキュリティグループに追加します。VPC セキュリティグループの変更方法の詳細については、Amazon Virtual Private Cloudユーザーガイドの「VPC のセキュリティグループ」を参照してください。

   以下の条件が満たされると、IP アドレスが変更される場合があります。

   - 外部ソースインスタンスと DB インスタンス間の通信にパブリック IP アドレスを使用している。
   - 外部ソースインスタンスが停止して再起動した。

   これらの条件が満たされている場合は、追加する前に IP アドレスを確認してください。

   Amazon RDS の DB インスタンスの IP アドレスからの接続を許可するようにローカルネットワークを設定することも必要になる場合があります。これは、ローカルネットワークから外部の MySQL または MariaDB インスタンスと通信できるようにするためです。Amazon RDS DB インスタンスの IP アドレスを確認するには、`host` コマンドを使用します。

   ```
   host db_instance_endpoint
   ```

   ホスト名は Amazon RDS DB インスタンスのエンドポイントの DNS 名です。

6. 選択したクライアントを使用して、外部のインスタンスに接続し、レプリケーションに使用される ユーザーを作成します。このアカウントをレプリケーション専用に使用し、セキュリティを強化するためドメインに制限します。以下に例を示します。

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

7. 外部の インスタンスについて、`REPLICATION CLIENT` と `REPLICATION SLAVE` の特権をレプリケーションユーザーに付与します。例えば、すべてのデータベースに対する `REPLICATION CLIENT` および `REPLICATION SLAVE` 権限を "`repl_user`" ユーザーに付与するには、以下のコマンドを実行します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

8. Amazon RDS DB インスタンスをレプリカにします。これを行うには、まず、マスターユーザーとして Amazon RDS の DB インスタンスに接続します。次に、mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降) または mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) のコマンドを使用して、外部の MySQL または MariaDB データベースをソースインスタンスとして指定します。ステップ 2 で特定したマスターログファイル名とマスターログの場所を使用します。次のコマンドはその例です。

   MySQL 8.4

   ```
   CALL mysql.rds_set_external_source ('mysourceserver.mydomain.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   MariaDB と MySQL 8.0 および 5.7

   ```
   CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1);
   ```

   > **Note**
   >
   > RDS for MySQL では、代わりに mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降) または mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) のストアドプロシージャを実行することで、遅延レプリケーションを使用するよう選択できます。RDS for MySQL で遅延レプリケーションを使用する 1 つの理由は、mysql.rds_start_replication_until ストアドプロシージャで災害対策を有効にするためです。現在、RDS for MariaDB では遅延レプリケーションはサポートされていますが、`mysql.rds_start_replication_until` プロシージャはサポートされていません。

9. Amazon RDS DB インスタンスで、mysql.rds_start_replication コマンドを実行してレプリケーションをスタートします。

   ```
   CALL mysql.rds_start_replication;
   ```

Amazon RDS for MySQL のマルチソースレプリケーションの設定

マルチソースレプリケーションを使用すると、Amazon RDS for MySQL DB インスタンスを、複数の RDS for MySQL ソース DB インスタンスからバイナリログイベントを受信するレプリカとして設定できます。マルチソースレプリケーションは、次のエンジンバージョンを実行している RDS for MySQL DB インスタンスでサポートされています。

- すべての MySQL 8.4 バージョン
- 8.0.35 以降のマイナーバージョン
- 5.7.44 以降のマイナーバージョン

MySQL マルチソースレプリケーションの詳細については、MySQL ドキュメントの「MySQL マルチソースレプリケーション」を参照してください。MySQL ドキュメントにはこの機能に関する詳細情報が含まれていますが、このトピックでは RDS for MySQL DB インスタンスでマルチソースレプリケーションチャネルを設定および管理する方法を説明します。

マルチソースレプリケーションのユースケース

以下のケースは、RDS for MySQL でマルチソースレプリケーションを使用するのに適しています。

- 個別の DB インスタンス上の複数のシャードを 1 つのシャードにマージまたは結合する必要があるアプリケーション。
- 複数のソースから統合されたデータからレポートを生成する必要があるアプリケーション。
- 複数の RDS for MySQL DB インスタンスに分散されるデータの統合された長期バックアップを作成するための要件。

マルチソースレプリケーションの前提条件

マルチソースレプリケーションを設定する前に、以下の前提条件を完了してください。

- 各ソース RDS for MySQL DB インスタンスで自動バックアップが有効になっていることを確認します。自動バックアップを有効にすると、バイナリログ記録が有効になります。自動バックアップを有効にする方法については、「the section called “自動バックアップの有効化”」を参照してください。
- レプリケーションエラーを回避するには、ソース DB インスタンスへの書き込み操作をブロックすることをお勧めします。そのためには、RDS for MySQL ソース DB インスタンスにアタッチされ

たカスタムパラメータグループで、read-only パラメータを ON に設定します。AWS マネジメントコンソール または AWS CLI を使用して、新しいカスタムパラメータグループを作成するか、既存のものを変更できます。詳細については、「the section called “DB パラメータグループを作成する”」および「the section called “DB パラメータグループのパラメータの変更”」を参照してください。

- ソース DB インスタンスごとに、インスタンスの IP アドレスをマルチソース DB インスタンスの Amazon 仮想プライベートクラウド (VPC) セキュリティグループに追加します。ソース DB インスタンスの IP アドレスを識別するには、 コマンド dig *RDS Endpoint* を実行します。送信先のマルチソース DB インスタンスと同じ VPC 内の Amazon EC2 インスタンスから コマンドを実行します。
- 次の例のように、ソース DB インスタンスごとに、クライアントを使用して DB インスタンスに接続し、レプリケーションに必要な権限を持つデータベースユーザーを作成します。

```
CREATE USER 'repl_user' IDENTIFIED BY 'password';
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user';
```

> **Note**
>
> MySQL 8.4 以降、REPLICATION SLAVE 権限は廃止され、REPLICATION REPLICA に置き換えられました。MySQL 8.4 以降のバージョンでは、次の構文を代わりに使用します。
>
> ```
> CREATE USER 'repl_user' IDENTIFIED BY 'password';
> GRANT REPLICATION CLIENT, REPLICATION REPLICA ON *.* TO 'repl_user';
> ```

RDS for MySQL DB インスタンスでのマルチソースレプリケーションチャネルの設定

マルチソースレプリケーションチャネルの設定は、シングルソースレプリケーションの設定と同様です。マルチソースレプリケーションの場合、まず、ソースインスタンスのバイナリログ記録を有効にします。次に、ソースからマルチソースレプリカにデータをインポートします。次に、バイナリログ座標を使用するか、GTID 自動配置を使用して、各ソースからレプリケーションを開始します。

RDS for MySQL DB インスタンスを 2 つ以上の RDS for MySQL DB インスタンスのマルチソースレプリカとして設定するには、以下のステップを実行します。

トピック

- ステップ 1: ソース DB インスタンスからマルチソースレプリカにデータをインポートする

- ステップ 2: ソース DB インスタンスからマルチソースレプリカへのレプリケーションを開始する

ステップ 1: ソース DB インスタンスからマルチソースレプリカにデータをインポートする

各ソース DB インスタンスで以下のステップを実行します。

ソースからマルチソースレプリカにデータをインポートする前に、 SHOW MASTER STATUS コマンドを実行して、現在のバイナリログファイルと場所を決定します。次のステップで使用するために、これらの詳細を書き留めておきます。この出力例では、ファイルは mysql-bin-changelog.000031 であり、場所は 107 です。

> **Note**
>
> MySQL 8.4 以降、SHOW MASTER STATUS コマンドは廃止され、SHOW BINARY LOG STATUS に置き換えられました。MySQL 8.4 以降のバージョンでは、代わりに SHOW BINARY LOG STATUS を使用します。

```
File                           Position
-----------------------------------
mysql-bin-changelog.000031      107
-----------------------------------
```

次に、次の例のように、mysqldump を使用して、ソース DB インスタンスからマルチソースレプリカにデータベースをコピーします。

```
mysqldump --databases database_name \
 --single-transaction \
 --compress \
 --order-by-primary \
 -u RDS_user_name \
 -p RDS_password \
 --host=RDS Endpoint | mysql \
 --host=RDS Endpoint \
 --port=3306 \
 -u RDS_user_name \
-p RDS_password
```

データベースをコピーしたら、ソース DB インスタンスの読み取り専用パラメータを OFF に設定できます。

ステップ 2: ソース DB インスタンスからマルチソースレプリカへのレプリケーションを開始する

ソース DB インスタンスごとに、管理者ユーザーの認証情報を使用してインスタンスに接続し、次の 2 つのストアドプロシージャを実行します。これらのストアドプロシージャは、チャネルでレプリケーションを設定し、レプリケーションを開始します。この例では、前のステップの出力例のバイナリログファイルの名前と場所を使用します。

```
CALL mysql.rds_set_external_source_for_channel('mysourcehost.example.com', 3306,
 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 1, 'channel_1');
CALL mysql.rds_start_replication_for_channel('channel_1');
```

これらのストアドプロシージャやその他を使用してレプリケーションチャネルを設定および管理する方法の詳細については、「the section called “マルチソースレプリケーションの管理”」を参照してください。

マルチソースレプリケーションでのフィルターの使用

レプリケーションフィルターを使用して、マルチソースレプリカでレプリケートするデータベースとテーブルを指定できます。レプリケーションフィルターは、データベースとテーブルをレプリケーションに含めることも、レプリケーションから除外することもできます。レプリケーションフィルターの詳細については、「the section called “レプリケーションフィルターの設定”」を参照してください。

マルチソースレプリケーションでは、レプリケーションフィルターをグローバルまたはチャネルレベルで設定できます。チャネルレベルのフィルタリングは、 バージョン 8.0 または 8.4 を実行するサポートされている DB インスタンスでのみ使用できます。次の例は、グローバルまたはチャネルレベルでフィルターを設定する方法を示しています。

マルチソースレプリケーションでのフィルタリングでは、次の要件と動作に注意してください。

- チャネル名をバッククォート (``) で囲む必要があります。
- パラメータグループのレプリケーションフィルターを変更すると、更新を含むすべてのチャネルのマルチソースレプリカの sql_thread が再起動され、変更が動的に適用されます。更新にグローバルフィルターが含まれる場合、実行状態のすべてのレプリケーションチャネルが再起動されます。
- すべてのグローバルフィルターは、チャネル固有のフィルターの前に適用されます。
- フィルタがグローバルに適用され、チャネルレベルで適用されている場合、チャネルレベルのフィルタのみが適用されます。例えば、フィルターが

replicate_ignore_db="db1,`channel_22`:db2" の場合、db1 に設定された replicate_ignore_db が channel_22 以外のすべてのチャネルに適用された場合、channel_22 のみは db2 からの変更を無視します。

例 1: グローバルフィルターの設定

次の例では、temp_data データベースはすべてのチャネルのレプリケーションから除外されます。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name myparametergroup \
--parameters "ParameterName=replicate-ignore-
db,ParameterValue='temp_data',ApplyMethod=immediate"
```

例 2: チャネルレベルフィルターの設定

次の例では、sample22 データベース からの変更はチャネル channel_22 にのみ含まれます。同様に、sample99 データベース からの変更は、チャンネル channel_99 にのみ含まれます。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
--db-parameter-group-name myparametergroup \
--parameters "ParameterName=replicate-do-db,ParameterValue='\`channel_22\`:sample22,
\`channel_99\`:sample99',ApplyMethod=immediate"
```

マルチソースレプリケーションチャネルのモニタリング

次の方法を使用して、マルチソースレプリカの個々のチャネルをモニタリングできます。

- すべてのチャネルまたは特定のチャネルのステータスをモニタリングするには、マルチソースレプリカに接続し、SHOW REPLICA STATUS または SHOW REPLICA STATUS FOR CHANNEL '*channel_name*' コマンドを実行します。詳細については、MySQL ドキュメントの「レプリケーションステータスのチェック」を参照してください。
- レプリケーションチャネルが開始、停止、または削除されたときに通知を受け取るには、RDS イベント通知を使用します。詳細については、「the section called “Amazon RDS イベント通知の操作”」を参照してください。

- 特定のチャネルの遅延をモニタリングするには、そのチャネルの ReplicationChannelLagメトリクスを確認します。このメトリクスのデータポイントは期間が 60 秒 (1 分) であり、15 日間使用できます。チャネルのレプリケーションチャネルラグを特定するには、インスタンス識別子とレプリケーションチャネル名を使用します。この遅延が特定のしきい値を超えたときに通知を受け取るには、CloudWatch アラームを設定できます。詳細については、「the section called "CloudWatch を使用した RDS のモニタリング"」を参照してください。

マルチソースレプリケーションの考慮事項とベストプラクティス

RDS for MySQL でマルチソースレプリケーションを使用する前に、以下の考慮事項とベストプラクティスを確認してください。

- マルチソースレプリカとして設定された DB インスタンスに、複数のソースインスタンスからのワークロードを処理するのに十分なリソース (スループット、メモリ、CPU、IOPS など) があることを確認します。
- マルチソースレプリカのリソース使用率を定期的にモニタリングし、リソースに負荷をかけずにワークロードを処理するようにストレージまたはインスタンスの設定を調整します。
- システム変数 `replica_parallel_workers` を `0` より大きい値に設定することで、マルチソースレプリカでマルチスレッドレプリケーションを設定できます。この場合、各チャネルに割り当てられるスレッド数は、この変数の値に、適用スレッドを管理する 1 つのコーディネータースレッドを足した数です。
- 競合を避けるために、レプリケーションフィルターを適切に設定します。データベース全体をレプリカの別のデータベースにレプリケートするには、`--replicate-rewrite-db` オプションを使用できます。例えば、データベース A のすべてのテーブルをレプリカインスタンスのデータベース B にレプリケートできます。このアプローチは、すべてのソースインスタンスが同じスキーマ命名規則を使用している場合に役立ちます。`--replicate-rewrite-db` オプションの詳細については、MySQL ドキュメントで「Replica Server Options and Variables」を参照してください。
- レプリケーションエラーを回避するには、レプリカに書き込まないでください。マルチソースレプリカで `read_only` パラメータを有効にして、書き込み操作をブロックすることをお勧めします。これにより、書き込み操作の競合によるレプリケーションの問題を排除できます。
- マルチソースレプリカで実行されるソートや高負荷結合などの読み取り操作のパフォーマンスを向上させるには、RDS Optimized Reads の使用を検討してください。この機能は、大きな一時テーブルまたはソートファイルに依存するクエリに役立ちます。詳細については、「the section called "RDS Optimized Reads によるクエリパフォーマンスの向上"」を参照してください。
- レプリケーションの遅延を最小限に抑え、マルチソースレプリカのパフォーマンスを向上させるには、最適化された書き込みを有効にすることを検討してください。詳細については、「the section

called “RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上”」を参照してください。

- 一度に 1 つのチャネルで管理操作 (設定の変更など) を実行し、複数の接続から複数のチャンネルに変更を実行しないようにします。これらのプラクティスにより、レプリケーション操作で競合が発生する可能性があります。例えば、複数の接続から `rds_skip_repl_error_for_channel` と `rds_start_replication_for_channel` プロシージャを同時に実行すると、意図したのとは異なるチャネルでイベントがスキップされる可能性があります。
- マルチソースレプリケーションインスタンスでバックアップを有効にし、そのインスタンスから Amazon S3 バケットにデータをエクスポートして、長期間保存できます。ただし、個々のソースインスタンスで適切な保持期間を持つバックアップを設定することも重要です。Amazon S3 へのスナップショットデータのエクスポートの詳細については、「the section called “Amazon S3 への DB スナップショットデータのエクスポート”」を参照してください。
- マルチソースレプリカへの読み取りワークロードを分散するには、マルチソースレプリカからリードレプリカを作成できます。これらのリードレプリカは、アプリケーションの要件に応じて異なる AWS リージョン に配置できます。リードレプリカの詳細については、「the section called “MySQL リードレプリカ”」を参照してください。

RDS for MySQL のマルチソースレプリケーションの制限事項

RDS for MySQL のマルチソースレプリケーションには、以下の制限が適用されます。

- 現在、RDS for MySQL では、マルチソースレプリカに最大 15 個のチャネルを設定できます。
- リードレプリカインスタンスをマルチソースレプリカとして設定することはできません。
- エンジンバージョン 5.7 を実行している RDS for MySQL でマルチソースレプリケーションを設定するには、レプリカインスタンスで Performance Schema を有効にする必要があります。エンジンバージョン 8.0 または 8.4 を実行している RDS for MySQL では、Performance Schema の有効化はオプションです。
- エンジンバージョン 5.7 を実行している RDS for MySQL の場合、レプリケーションフィルターはすべてのレプリケーションチャネルに適用されます。エンジンバージョン 8.0 または 8.4 を実行している RDS for MySQL では、すべてのレプリケーションチャネルまたは個々のチャネルに適用されるフィルターを設定できます。
- RDS スナップショットを復元するか、ポイントインタイムリストア (PITR) を実行しても、マルチソースのレプリカチャネル設定は復元されません。
- マルチソースレプリカのリードレプリカを作成すると、マルチソースインスタンスからのデータのみがレプリケートされます。チャネル設定は復元されません。

- MySQL は、チャネルごとに異なる数の並列ワーカーの設定をサポートしていません。すべてのチャンネルが、`replica_parallel_workers` 値に基づいて同じ数の並列ワーカーを取得します。

マルチソースレプリケーションターゲットがマルチ AZ DB クラスターの場合、以下の追加制限が適用されます。

- ソース RDS for MySQL インスタンスへの書き込みが発生する前に、そのインスタンスにチャネルを設定する必要があります。
- 各ソース RDS for MySQL インスタンスでは、GTID ベースのレプリケーションが有効になっている必要があります。
- DB クラスターのフェイルオーバーイベントにより、マルチソースレプリケーション設定が削除されます。その設定を復元するには、設定ステップを繰り返す必要があります。

RDS for MySQL のアクティブ/アクティブクラスターの設定

Amazon RDS のアクティブ/アクティブクラスターは、複数のノードが読み取りおよび書き込みオペレーションをアクティブに処理し、可用性とスケーラビリティを向上させるためにインスタンス間でワークロードを分散するデータベース設定です。クラスター内の各ノードが同期され、データ整合性が維持されるため、ノード障害発生時の高可用性と迅速なフェイルオーバーが可能になります。

MySQL グループレプリケーションプラグインを使用して、RDS for MySQL のアクティブ/アクティブクラスターを設定できます。グループレプリケーションプラグインは、次のバージョンを実行している RDS for MySQL DB インスタンスでサポートされています。

- すべての MySQL 8.4 バージョン
- MySQL 8.0.35 以降のマイナーバージョン

MySQL グループレプリケーションの詳細については、MySQL ドキュメントの「グループレプリケーション」を参照してください。MySQL ドキュメントにはこの機能に関する詳細情報が含まれていますが、このトピックでは RDS for MySQL DB インスタンスでプラグインを設定および管理する方法を説明します。

> **Note**
>
> 簡潔にするために、このトピックの「アクティブ/アクティブ」クラスターに関するすべての言及は、MySQL グループレプリケーションプラグインを使用するアクティブ/アクティブクラスターを指します。

アクティブ/アクティブクラスターのユースケース

アクティブ/アクティブクラスターの使用には、次のケースが適しています。

- 書き込み操作をサポートするために、クラスター内のすべての DB インスタンスを必要とするアプリケーション。グループレプリケーションプラグインは、アクティブ/アクティブクラスター内の各 DB インスタンスでデータの整合性を維持します。この機能の詳細については、MySQL ドキュメントの「グループレプリケーション」を参照してください。
- データベースの継続的な可用性を必要とするアプリケーション。アクティブ/アクティブクラスターでは、データはクラスター内のすべての DB インスタンスに保持されます。1 つの DB インスタンスに障害が発生した場合、アプリケーションはクラスター内の別の DB インスタンスにトラフィックを再ルーティングできます。

- 負荷分散のために、クラスター内の複数の DB インスタンス間で読み取り操作と書き込み操作を分割する必要があるアプリケーション。アクティブ/アクティブクラスターを使用すると、アプリケーションは特定の DB インスタンスに読み取りトラフィックを送信し、他のインスタンスに書き込みトラフィックを送信できます。また、読み取りまたは書き込みの送信先となる DB インスタンスをいつでも切り替えることができます。

トピック

- アクティブ/アクティブクラスターの制限と考慮事項
- クロス VPC アクティブ/アクティブクラスターの準備
- アクティブ/アクティブクラスターに必要なパラメータ設定
- 既存の DB インスタンスをアクティブ/アクティブクラスターに変換する
- 新しい DB インスタンスを使用したアクティブ/アクティブクラスターのセットアップ
- DB インスタンスをアクティブ/アクティブクラスターに追加する
- アクティブ/アクティブクラスターのモニタリング
- アクティブ/アクティブクラスター内の DB インスタンスでのグループレプリケーションを停止する
- アクティブ/アクティブクラスター内の DB インスタンスの名前を変更する
- アクティブ/アクティブクラスターから DB インスタンスを削除する

アクティブ/アクティブクラスターの制限と考慮事項

Amazon RDS のアクティブ/アクティブクラスターは、複数のインスタンスにワークロードを分散することで、可用性とスケーラビリティを向上させます。ただし、このアーキテクチャを使用する際に留意すべき重要な制限と考慮事項があります。

以下のセクションでは、レプリケーションの遅延、競合解決、リソースの割り当て、フェイルオーバー動作などの主要な要因の概要を説明します。これらの考慮事項を理解することは、アクティブ/アクティブクラスターデプロイで最適なパフォーマンスと信頼性を確保するのに役立ちます。

トピック

- RDS for MySQL アクティブ/アクティブクラスターの制限事項
- RDS for MySQL アクティブ/アクティブクラスターに関する考慮事項とベストプラクティス

RDS for MySQL アクティブ/アクティブクラスターの制限事項

RDS for MySQL のアクティブ/アクティブクラスターには、以下の制限事項が適用されます。

- アクティブ/アクティブクラスター内の DB インスタンスについては、マスターユーザー名を `rdsgrprepladmin` にすることはできません。このユーザー名は、グループレプリケーション接続用に予約されています。
- アクティブ/アクティブクラスター内のリードレプリカ付き DB インスタンスの場合、`Replicating` 以外のレプリケーションステータスが長くなると、ログファイルがストレージ制限を超える可能性があります。リードレプリカのステータスについては、「リードレプリケーションのモニタリング」を参照してください。
- ブルー/グリーンデプロイは、アクティブ/アクティブクラスター内の DB インスタンスではサポートされません。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。
- Kerberos 認証は、アクティブ/アクティブクラスター内の DB インスタンスではサポートされません。詳細については、「Amazon RDS for MySQL での Kerberos 認証の使用」を参照してください。
- マルチ AZ DB クラスターの DB インスタンスをアクティブ/アクティブクラスターに追加することはできません。ただし、マルチ AZ DB インスタンスデプロイの DB インスタンスをアクティブ/アクティブクラスターに追加することはできます。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。
- プライマリキーを持たないテーブルは、グループレプリケーションプラグインによって書き込みが拒否されるため、アクティブ/アクティブクラスターにレプリケートされません。
- InnoDB 以外のテーブルは、アクティブ/アクティブクラスターにレプリケートされません。
- アクティブ/アクティブクラスターは、クラスター内の異なる DB インスタンスでの DML ステートメントと DDL ステートメントの同時実行をサポートしていません。
- グループのレプリケーションモードとしてシングルプライマリモードを使用するようにアクティブ/アクティブクラスターを設定することはできません。この設定については、代わりにマルチ AZ DB クラスターを使用することをお勧めします。詳細については、「Amazon RDS のマルチ AZ DB クラスターデプロイ」を参照してください。
- マルチソースレプリケーションは、アクティブ/アクティブクラスターの DB インスタンスではサポートされていません。
- クロスリージョンのアクティブ/アクティブクラスターは、グループレプリケーション接続に認証機関 (CA) 検証を適用できません。

RDS for MySQL アクティブ/アクティブクラスターに関する考慮事項とベストプラクティス

RDS for MySQL アクティブ/アクティブクラスターを使用する前に、以下の考慮事項とベストプラクティスを確認してください。

- アクティブ/アクティブクラスターには、10 個以上の DB インスタンスを含めることはできません。
- グループレプリケーションプラグインを使用すると、アクティブ/アクティブクラスターのトランザクション整合性保証を制御できます。詳細については、MySQL ドキュメントの「トランザクション整合性保証」を参照してください。
- 異なる DB インスタンスがアクティブ/アクティブクラスター内の同じ行を更新すると、競合が発生する可能性があります。競合と競合の解決については、MySQL ドキュメントの「グループレプリケーション」を参照してください。
- 耐障害性を確保するために、アクティブ/アクティブクラスターに少なくとも 3 つの DB インスタンスを含めます。アクティブ/アクティブクラスターは、1 つまたは 2 つの DB インスタンスのみで設定できますが、クラスターは耐障害性がありません。耐障害性の詳細については、MySQL ドキュメントの「耐障害性」を参照してください。
- DB インスタンスが既存のアクティブ/アクティブクラスターに参加し、クラスター内の最も低いエンジンバージョンと同じエンジンバージョンを実行している場合、DB インスタンスは読み取り/書き込みモードで参加します。
- DB インスタンスが既存のアクティブ/アクティブクラスターに参加し、クラスター内の最も低いエンジンバージョンより高いエンジンバージョンを実行している場合、DB インスタンスは読み取り専用モードにとどまる必要があります。
- DB パラメータグループの `rds.group_replication_enabled` パラメータを 1 に設定することによって DB インスタンスのグループレプリケーションを有効にしても、レプリケーションが開始されていないか、開始に失敗した場合、データの不整合を防ぐために DB インスタンスはスーパー読み取り専用モードになります。スーパー読み取り専用モードの詳細については、「MySQL ドキュメント」を参照してください。
- アクティブ/アクティブクラスター内の DB インスタンスをアップグレードできますが、アクティブ/アクティブクラスター内の他のすべての DB インスタンスが同じエンジンバージョンまたはそれ以上のエンジンバージョンにアップグレードされるまで、DB インスタンスは読み取り専用です。DB インスタンスをアップグレードすると、アップグレードが完了したとき、DB インスタンスは同じアクティブ/アクティブクラスターに自動的に参加します。DB インスタンスの意図しない読み取り専用モードへの切り替えを回避するには、自動マイナーバージョンアップグレードを無効

にします。MySQL DB インスタンスのアップグレードの詳細については、「RDS for MySQL DB エンジンのアップグレード」を参照してください。

- マルチ AZ DB インスタンスデプロイの DB インスタンスを既存のアクティブ/アクティブクラスターに追加できます。アクティブ/アクティブクラスターのシングル AZ DB インスタンスをマルチ AZ DB インスタンスデプロイに変換することもできます。マルチ AZ 配置のプライマリ DB インスタンスに障害が発生した場合、そのプライマリインスタンスはスタンバイインスタンスにフェイルオーバーします。フェイルオーバーが完了すると、新しいプライマリ DB インスタンスは同じクラスターに自動的に参加します。マルチ AZ DB インスタンスデプロイの詳細については、「Amazon RDS のマルチ AZ DB インスタンスデプロイ」を参照してください。
- アクティブ/アクティブクラスターの DB インスタンスでは、メンテナンスウィンドウの時間範囲を変更することをお勧めします。この方法により、クラスター内の複数の DB インスタンスがメンテナンスのために同時にオフラインになるのを避けることができます。詳細については、「Amazon RDS メンテナンスウィンドウ」を参照してください。
- アクティブ/アクティブクラスターは、DB インスタンス間の接続に SSL を使用できます。SSL 接続を設定するには、group_replication_recovery_use_ssl および group_replication_ssl_mode パラメータを設定します。これらのパラメータの値は、アクティブ/アクティブクラスター内のすべての DB インスタンスで一致する必要があります。

  現在、アクティブ/アクティブクラスターは、AWS リージョン 間の接続の認証機関 (CA) 検証をサポートしていません。そのため、group_replication_ssl_mode パラメータは、クロスリージョンクラスターの場合は `DISABLED` (デフォルト) または `REQUIRED` に設定する必要があります。
- RDS for MySQL のアクティブ/アクティブクラスターは、マルチプライマリモードで実行します。group_replication_enforce_update_everywhere_checks のデフォルト値は `ON` であり、パラメータは静的です。このパラメータが `ON` に設定されると、外部キーのカスケード制約があるテーブルにアプリケーションを挿入できません。
- RDS for MySQL アクティブ/アクティブクラスターは、XCOM の代わりに MySQL 通信スタックを使用して接続セキュリティを確保します。詳細については、MySQL ドキュメントの「接続セキュリティ管理のための通信スタック」を参照してください。
- DB パラメータグループがアクティブ/アクティブクラスター内の DB インスタンスに関連付けられている場合は、この DB パラメータグループをクラスター内の他の DB インスタンスにのみ関連付けることをお勧めします。
- アクティブ/アクティブクラスターは RDS for MySQL DB インスタンスのみをサポートします。これらの DB インスタンスは、サポートされているバージョンの DB エンジンを実行している必要があります。

- アクティブ/アクティブクラスター内の DB インスタンスに予期しない障害が発生した場合、RDS は DB インスタンスの回復を自動的に開始します。DB インスタンスが回復しない場合は、クラスター内の正常な DB インスタンスでポイントインタイムリカバリを実行して、新しい DB インスタンスに置き換えることをお勧めします。手順については、「ポイントインタイムリカバリを使用してアクティブ/アクティブクラスターに DB インスタンスを追加する」を参照してください。
- クラスター内の他の DB インスタンスに影響を与えることなく、アクティブ/アクティブクラスター内の DB インスタンスを削除できます。DB インスタンスの削除については、「DB インスタンスを削除する」を参照してください。
- DB インスタンスが意図せずにアクティブ/アクティブクラスターを離れると、デフォルトで `group_replication_exit_state_action`パラメータが `OFFLINE_MODE` に変わります。この状態では、DB インスタンスにアクセスできず、DB インスタンスを再起動してオンラインに戻し、クラスターを再結合する必要があります。この動作を変更するには、カスタムパラメータグループの `group_replication_exit_state_action` パラメータを変更します。パラメータを `READ_ONLY` に設定することで、DB インスタンスが意図せずにクラスターを離れたときに、オフラインではなくスーパー読み取り専用状態になります。

クロス VPC アクティブ/アクティブクラスターの準備

複数の VPC の Amazon RDS for MySQL DB インスタンスを使用して、アクティブ/アクティブクラスターを設定できます。VPC は同じ AWS リージョン にあっても、異なる AWS リージョン にあってもかまいません。

> **Note**
>
> 複数の AWS リージョン 間でトラフィックを送信すると、追加コストが発生する可能性があります。詳細については、「一般的なアーキテクチャのデータ転送コストの概要」を参照してください。

単一の VPC でアクティブ/アクティブクラスターを設定する場合は、これらのステップをスキップして 新しい DB インスタンスを使用したアクティブ/アクティブクラスターのセットアップ に進むことができます。

複数の VPC の DB インスタンスを使用して、アクティブ/アクティブクラスターを準備するには

1. CIDR ブロックの IPv4 アドレス範囲が次の要件を満たしていることを確認します。

- VPC の CIDR ブロック内の IPv4 アドレス範囲は重複できません。
- CIDR ブロック内のすべての IPv4 アドレス範囲は、128.0.0.0/*subnet_mask* より低いか、128.0.0.0/*subnet_mask* より高い必要があります。

以下の範囲は、これらの要件を示しています。

- 1 つの VPC の 10.1.0.0/16 と、他の VPC の 10.2.0.0/16 がサポートされています。
- 1 つの VPC の 172.1.0.0/16 と、他の VPC の 172.2.0.0/16 がサポートされています。
- 範囲が重複するため、1 つの VPC の 10.1.0.0/16 と他の VPC の 10.1.0.0/16 はサポートされません。
- 1 つの VPC の 10.1.0.0/16 と、他の VPC の 172.1.0.0/16 は、一方が 128.0.0.0/*subnet_mask* より低く、もう一方が 128.0.0.0/*subnet_mask* より高いため、サポートされません。

CIDR ブロックの詳細については、「Amazon VPC ユーザーガイド」の「VPC CIDR ブロック」を参照してください。

2. 各 VPC で、DNS 解決と DNS ホスト名の両方が有効になっていることを確認します。

   詳細については、「Amazon VPC ユーザーガイド」の「VPC の DNS 属性の表示と更新」を参照してください。

3. 次のいずれかの方法でトラフィックをルーティングできるように、 VPC を設定します。

   - VPC 間の VPC ピアリング接続を確立します。

     手順については、「Amazon VPC ピアリングガイド」の「VPC ピアリング接続を作成する」を参照してください。各 VPC で、ピアリング接続された VPC のセキュリティグループを参照するセキュリティグループのインバウンドルールがあることを確認します。これにより、トラフィックはピアリング接続された VPC の参照セキュリティグループに関連付けられたインスタンスに出入りできます。手順については、「Amazon VPC ピアリングガイド」の「ピアリングセキュリティグループを参照するようにセキュリティグループを更新する」を参照してください。

   - VPC 間にトランジットゲートウェイを作成します。

     手順については、「Amazon VPC Transit Gateway」の「トランジットゲートウェイの開始方法」を参照してください。各 VPC で、他の VPC の CIDR を指定するインバウンドルールな

ど、他の VPC からのトラフィックを許可するセキュリティグループのインバウンドルールがあることを確認します。これにより、トラフィックはアクティブ/アクティブクラスター内の参照セキュリティグループに関連付けられたインスタンスに出入りできます。詳細については、「Amazon VPC ユーザーガイド」の「セキュリティグループを使用して AWS リソースへのトラフィックを制御する」を参照してください。

アクティブ/アクティブクラスターに必要なパラメータ設定

Amazon RDS for MySQL でアクティブ/アクティブクラスターのパラメータを設定することは、一貫したパフォーマンスと運用上の安定性を維持するために不可欠です。この表では、レプリケーション、競合解決、ワークロード分散を制御する主要なパラメータについて詳しく説明します。正しい設定により、ノード間の効率的な同期を確保し、レプリケーションの遅延を最小限に抑え、分散環境またはトラフィックの多い環境でのリソース使用率を最適化できます。

パラメータ	説明	必須の設定
`binlog_format`	バイナリログ形式を設定します。RDS for MySQL バージョン 8.0 以前の場合、デフォルト値は `MIXED` になります。RDS for MySQL 8.4 のデフォルト値は `ROW` です。詳細については、MySQL ドキュメントを参照してください。	`ROW`
`enforce_gtid_consistency`	ステートメント実行に GTID 整合性を適用します。RDS for MySQL のデフォルト値は `OFF` です。詳細については、MySQL ドキュメントを参照してください。	`ON`
`group_replication_group_name`	グループレプリケーション名を UUID に設定します。UUID の形式は `11111111-2222-3333-4444-555555555555` です。MySQL	MySQL UUID

パラメータ	説明	必須の設定
	DB インスタンスに接続して SELECT UUID() を実行することで、MySQL UUID を生成できます。値は、アクティブ/アクティブクラスター内のすべての DB インスタンスで同じである必要があります。詳細については、MySQL ドキュメントを参照してください。	
gtid-mode	GTID ベースのログ記録を制御します。RDS for MySQL のデフォルト値は OFF_PERMISSIVE です。詳細については、MySQL ドキュメントを参照してください。	ON
rds.custom_dns_resolution	VPC 内の Amazon DNS サーバーからの DNS 解決を許可するかどうかを指定します。DNS 解決は、グループレプリケーションが rds.group_replication_enabled パラメータで有効になっているときに有効である必要があります。グループレプリケーションが rds.group_replication_enabled パラメータで無効になっているときに DNS 解決を有効にすることはできません。詳細については、「Amazon VPC ユーザーガイド」の「Amazon DNS サーバー」を参照してください。	1

パラメータ	説明	必須の設定
rds.group_replication_enabled	DB インスタンスでグループレプリケーションを有効にするかどうかを指定します。グループレプリケーションは、アクティブ/アクティブクラスターの DB インスタンスで有効にする必要があります。	1
replica_preserve_commit_order (RDS for MySQL 8.4 以降のバージョン) または slave_preserve_commit_order (RDS for MySQL 8.0 バージョン)	レプリカでトランザクションをコミットする順序を制御します。RDS for MySQL のデフォルト値は ON です。詳細については、MySQL ドキュメントを参照してください。	ON

既存の DB インスタンスをアクティブ/アクティブクラスターに変換する

アクティブ/アクティブクラスターに移行する DB インスタンスの DB エンジンバージョンは、次のいずれかである必要があります。

- すべての MySQL 8.4 バージョン
- MySQL 8.0.35 以降のマイナーバージョン

エンジンバージョンをアップグレードする必要がある場合は、「RDS for MySQL DB エンジンのアップグレード」を参照してください。

複数の VPC に DB インスタンスを持つアクティブ/アクティブクラスターを設定する場合は、「クロス VPC アクティブ/アクティブクラスターの準備」の前提条件を満たしていることを確認してください。

既存の DB インスタンスを RDS for MySQL のアクティブ/アクティブクラスターに移行するには、以下のステップを実行します。

トピック

- ステップ 1: 1 つ以上のカスタムパラメータグループでアクティブ/アクティブクラスターパラメータを設定する
- ステップ 2: 必要なグループレプリケーションパラメータが設定されている DB パラメータグループに DB インスタンスを関連付ける
- ステップ 3: アクティブ/アクティブクラスターを作成する
- ステップ 4: アクティブ/アクティブクラスター用の追加の RDS for MySQL DB インスタンスを作成する
- ステップ 5: 変換する DB インスタンスでグループを初期化する
- ステップ 6: アクティブ/アクティブクラスター内の他の DB インスタンスでレプリケーションを開始する
- ステップ 7: (推奨) アクティブ/アクティブクラスターのステータスを確認する

ステップ 1: 1 つ以上のカスタムパラメータグループでアクティブ/アクティブクラスターパラメータを設定する

アクティブ/アクティブクラスターの RDS for MySQL DB インスタンスは、必須パラメータに正しい設定を持つカスタムパラメータグループに関連付けられている必要があります。パラメータとそれぞれの必須パラメータの詳細については、「アクティブ/アクティブクラスターに必要なパラメータ設定」を参照してください。

これらのパラメータは、新しいパラメータグループまたは既存のパラメータグループで設定できます。ただし、アクティブ/アクティブクラスターに含まれていない DB インスタンスに誤って影響しないように、新しいカスタムパラメータグループを作成することを強くお勧めします。アクティブ/アクティブクラスター内の DB インスタンスは、同じ DB パラメータグループまたは異なる DB パラメータグループに関連付けることができます。

AWS マネジメントコンソール または AWS CLI を使用して、新しいカスタムパラメータグループを作成できます。詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。次の例では、create-db-parameter-group AWS CLI コマンドを実行して、RDS for MySQL 8.0 で *myactivepg* という名前のカスタム DB パラメータグループを作成します。

Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --db-parameter-group-family mysql8.0 \
  --description "Parameter group for active-active clusters"
```

Windows の場合:

```
aws rds create-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --db-parameter-group-family mysql8.0 ^
  --description "Parameter group for active-active clusters"
```

AWS マネジメントコンソール または AWS CLI を使用して、カスタムパラメータグループのパラメータを設定することもできます。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

次の例では、modify-db-parameter-group AWS CLI コマンドを実行して、RDS for MySQL 8.0 でパラメータを設定します。RDS for MySQL 8.4 でこの例を使用するには、`slave_preserve_commit_order` を `replica_preserve_commit_order` に変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" \
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" \

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" \

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 \

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
```

```
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" ^
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" ^

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" ^

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 ^

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

ステップ 2: 必要なグループレプリケーションパラメータが設定されている DB パラメータグループに DB インスタンスを関連付ける

前のステップで作成または変更したパラメータグループに DB インスタンスを関連付けます。手順については、「Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける」を参照してください。

DB インスタンスを再起動して、新しいパラメータ設定を有効にします。手順については、「DB インスタンスの再起動」を参照してください。

ステップ 3: アクティブ/アクティブクラスターを作成する

DB インスタンスに関連付けられている DB パラメータグループで、group_replication_group_seeds パラメータを、変換する DB インスタンスのエンドポイントに設定します。

AWS マネジメントコンソール または AWS CLI を使用して、パラメータを設定できます。このパラメータを設定した後に DB インスタンスを再起動する必要はありません。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

次の例では、modify-db-parameter-group AWS CLI コマンドを実行して、パラメータを設定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

ステップ 4: アクティブ/アクティブクラスター用の追加の RDS for MySQL DB インスタンスを作成する

アクティブ/アクティブクラスター用の追加の DB インスタンスを作成するには、変換する DB インスタンスでポイントインタイムリカバリを実行します。手順については、「ポイントインタイムリカバリを使用してアクティブ/アクティブクラスターに DB インスタンスを追加する」を参照してください。

アクティブ/アクティブクラスターには、最大 9 つの DB インスタンスを含めることができます。クラスターに必要な DB インスタンスの数に達するまで、DB インスタンスでポイントインタイムリカバリを実行します。ポイントインリカバリを実行するときは、追加する DB インスタンスを、`rds.group_replication_enabled` が 1 に設定されている DB パラメータグループに関連付ける必要があります。そうしないと、新しく追加された DB インスタンスでグループレプリケーションが開始されません。

ステップ 5: 変換する DB インスタンスでグループを初期化する

グループを初期化し、レプリケーションを開始します。

1. SQL クライアントで変換する DB インスタンスに接続します。RDS for Oracle DB インスタンスへの接続の詳細については、「MySQL DB インスタンスへの接続」を参照してください。
2. SQL クライアントで、次のストアドプロシージャを実行し、*group_replication_user_password* を `rdsgrprepladmin` ユーザーのパスワードに

置き換えます。rdsgrprepladmin ユーザーは、アクティブ/アクティブクラスターのグループレプリケーション接続用に予約されています。このユーザーのパスワードは、アクティブ/アクティブクラスター内のすべての DB インスタンスで同じである必要があります。

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(1);
```

この例では、binlog retention hours 値を 168 に設定します。つまり、バイナリログファイルは DB インスタンスで 7 日間保持されます。この値は、要件に合わせて調整できます。

この例では、mysql.rds_group_replication_start ストアドプロシージャで 1 を指定し、現在の DB インスタンスで新しいグループを初期化します。

この例で呼び出されるストアドプロシージャの詳細については、「アクティブ/アクティブクラスターの管理」を参照してください。

ステップ 6: アクティブ/アクティブクラスター内の他の DB インスタンスでレプリケーションを開始する

アクティブ/アクティブクラスター内の各 DB インスタンスについて、SQL クライアントを使用してインスタンスに接続し、次のストアドプロシージャを実行します。*group_replication_user_password* を rdsgrprepladmin ユーザーのパスワードに置き換えます。

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(0);
```

この例では、binlog retention hours 値を 168 に設定します。つまり、バイナリログファイルは DB インスタンスで 7 日間保持されます。この値は、要件に合わせて調整できます。

この例では、mysql.rds_group_replication_start ストアドプロシージャで 0 を指定して、現在の DB インスタンスを既存のグループに結合します。

> **Tip**
>
> これらのストアドプロシージャは、アクティブ/アクティブクラスター内の他のすべての DB インスタンスで実行してください。

ステップ 7: (推奨) アクティブ/アクティブクラスターのステータスを確認する

クラスターの各メンバーが正しく設定されていることを確認するには、アクティブ/アクティブクラスター内の DB インスタンスに接続し、次の SQL コマンドを実行して、クラスターのステータスを確認します。

```
SELECT * FROM performance_schema.replication_group_members;
```

出力には、次のサンプル出力のように、各 DB インスタンスの MEMBER_STATE の ONLINE が表示されます。

```
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+----------------------------+
3 rows in set (0.00 sec)
```

可能な MEMBER_STATE 値の詳細については、MySQL ドキュメントの「グループレプリケーションサーバーの状態」を参照してください。

新しい DB インスタンスを使用したアクティブ/アクティブクラスターのセットアップ

新しい Amazon RDS for MySQL DB インスタンスを使用してアクティブ/アクティブクラスターを設定するには、以下のステップを実行します。

複数の VPC に DB インスタンスを持つアクティブ/アクティブクラスターを設定する場合は、「クロス VPC アクティブ/アクティブクラスターの準備」の前提条件を満たしていることを確認してください。

トピック

- ステップ 1: 1 つ以上のカスタムパラメータグループでアクティブ/アクティブクラスターパラメータを設定する
- ステップ 2: アクティブ/アクティブクラスターの新しい RDS for MySQL DB インスタンスを作成する
- ステップ 3: アクティブ/アクティブクラスターの DB インスタンスを指定する
- ステップ 4: DB インスタンスでグループを初期化し、レプリケーションを開始する
- ステップ 5: アクティブ/アクティブクラスター内の他の DB インスタンスでレプリケーションを開始する
- ステップ 6: (推奨) アクティブ/アクティブクラスターのステータスを確認する
- ステップ 7: (オプション) アクティブ/アクティブクラスターの DB インスタンスにデータをインポートする

ステップ 1: 1 つ以上のカスタムパラメータグループでアクティブ/アクティブクラスターパラメータを設定する

アクティブ/アクティブクラスターの RDS for MySQL DB インスタンスは、必須パラメータに正しい設定を持つカスタムパラメータグループに関連付けられている必要があります。パラメータとそれぞれの必須パラメータの詳細については、「アクティブ/アクティブクラスターに必要なパラメータ設定」を参照してください。

これらのパラメータは、新しいパラメータグループまたは既存のパラメータグループで設定できます。ただし、アクティブ/アクティブクラスターに含まれていない DB インスタンスに誤って影響しないように、新しいカスタムパラメータグループを作成することを強くお勧めします。アクティブ/アクティブクラスター内の DB インスタンスは、同じ DB パラメータグループまたは異なる DB パラメータグループに関連付けることができます。

AWS マネジメントコンソール または AWS CLI を使用して、新しいカスタムパラメータグループを作成できます。詳細については、「Amazon RDS での DB パラメータグループの作成」を参照してください。次の例では、create-db-parameter-group AWS CLI コマンドを実行して、RDS for MySQL 8.0 で *myactivepg* という名前のカスタム DB パラメータグループを作成します。

Linux、macOS、Unix の場合:

```
aws rds create-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --db-parameter-group-family mysql8.0 \
  --description "Parameter group for active-active clusters"
```

Windows の場合:

```
aws rds create-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --db-parameter-group-family mysql8.0 ^
  --description "Parameter group for active-active clusters"
```

AWS マネジメントコンソール または AWS CLI を使用して、カスタムパラメータグループのパラメータを設定することもできます。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

次の例では、modify-db-parameter-group AWS CLI コマンドを実行して、RDS for MySQL 8.0 でパラメータを設定します。RDS for MySQL 8.4 でこの例を使用するには、`slave_preserve_commit_order` を `replica_preserve_commit_order` に変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" \

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" \
```

```
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" \

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" \

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 \

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='rds.group_replication_enabled',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='rds.custom_dns_resolution',ParameterValue='1',ApplyMethod=pending-
reboot" ^

 "ParameterName='enforce_gtid_consistency',ParameterValue='ON',ApplyMethod=pending-
reboot" ^
               "ParameterName='gtid-mode',ParameterValue='ON',ApplyMethod=pending-
reboot" ^

 "ParameterName='binlog_format',ParameterValue='ROW',ApplyMethod=immediate" ^

 "ParameterName='slave_preserve_commit_order',ParameterValue='ON',ApplyMethod=immediate"
 ^

 "ParameterName='group_replication_group_name',ParameterValue='11111111-2222-3333-4444-55555555
reboot"
```

ステップ 2: アクティブ/アクティブクラスターの新しい RDS for MySQL DB インスタンスを作成する

アクティブ/アクティブクラスターは、次のバージョンの RDS for MySQL DB インスタンスでサポートされています。

- すべての MySQL 8.4 バージョン

- MySQL バージョン 8.0.35 以降のマイナーバージョン

クラスターには最大 9 つの新しい DB インスタンスを作成できます。

AWS マネジメントコンソール または AWS CLI を使用して、新しい DB インスタンスを作成できます。DB インスタンスの作成の詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。DB インスタンスを作成するとき、前のステップで作成または変更した DB パラメータグループに関連付けます。

ステップ 3: アクティブ/アクティブクラスターの DB インスタンスを指定する

各 DB インスタンスに関連付けられている DB パラメータグループで、group_replication_group_seeds パラメータを、クラスターに含める DB インスタンスのエンドポイントに設定します。

AWS マネジメントコンソール または AWS CLI を使用して、パラメータを設定できます。このパラメータを設定した後に DB インスタンスを再起動する必要はありません。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

次の例では、modify-db-parameter-group AWS CLI コマンドを実行して、パラメータを設定します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name myactivepg \
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb2.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb3.123456789012.us-
east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
  --db-parameter-group-name myactivepg ^
  --parameters
 "ParameterName='group_replication_group_seeds',ParameterValue='myactivedb1.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb2.123456789012.us-
east-1.rds.amazonaws.com:3306,myactivedb3.123456789012.us-
east-1.rds.amazonaws.com:3306',ApplyMethod=immediate"
```

> **Tip**
>
> アクティブ/アクティブクラスターの DB インスタンスに関連付けられている各 DB パラメータグループで group_replication_group_seeds パラメータが設定されていることを確認します。

ステップ 4: DB インスタンスでグループを初期化し、レプリケーションを開始する

新しい DB を選択してグループを初期化し、レプリケーションを開始できます。そのためには、以下のステップを実行します。

1. アクティブ/アクティブクラスター内の DB インスタンスを選択し、SQL クライアントでその DB インスタンスに接続します。RDS for Oracle DB インスタンスへの接続の詳細については、「MySQL DB インスタンスへの接続」を参照してください。
2. SQL クライアントで、次のストアドプロシージャを実行し、*group_replication_user_password* を rdsgrprepladmin ユーザーのパスワードに置き換えます。rdsgrprepladmin ユーザーは、アクティブ/アクティブクラスターのグループレプリケーション接続用に予約されています。このユーザーのパスワードは、アクティブ/アクティブクラスター内のすべての DB インスタンスで同じである必要があります。

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(1);
```

この例では、binlog retention hours 値を 168 に設定します。つまり、バイナリログファイルは DB インスタンスで 7 日間保持されます。この値は、要件に合わせて調整できます。

この例では、mysql.rds_group_replication_start ストアドプロシージャで 1 を指定し、現在の DB インスタンスで新しいグループを初期化します。

この例で呼び出されるストアドプロシージャの詳細については、「アクティブ/アクティブクラスターの管理」を参照してください。

ステップ 5: アクティブ/アクティブクラスター内の他の DB インスタンスでレプリケーションを開始する

アクティブ/アクティブクラスター内の各 DB インスタンスについて、SQL クライアントを使用してインスタンスに接続し、次のストアドプロシージャを実行します。*group_replication_user_password* を rdsgrprepladmin ユーザーのパスワードに置き換えます。

```
call mysql.rds_set_configuration('binlog retention hours', 168); -- 7 days binlog
call mysql.rds_group_replication_create_user('group_replication_user_password');
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
call mysql.rds_group_replication_start(0);
```

この例では、binlog retention hours 値を 168 に設定します。つまり、バイナリログファイルは DB インスタンスで 7 日間保持されます。この値は、要件に合わせて調整できます。

この例では、mysql.rds_group_replication_start ストアドプロシージャで 0 を指定して、現在の DB インスタンスを既存のグループに結合します。

> **Tip**
>
> これらのストアドプロシージャは、アクティブ/アクティブクラスター内の他のすべての DB インスタンスで実行してください。

ステップ 6: (推奨) アクティブ/アクティブクラスターのステータスを確認する

クラスターの各メンバーが正しく設定されていることを確認するには、アクティブ/アクティブクラスター内の DB インスタンスに接続し、次の SQL コマンドを実行して、クラスターのステータスを確認します。

```
SELECT * FROM performance_schema.replication_group_members;
```

出力には、次のサンプル出力のように、各 DB インスタンスの MEMBER_STATE の ONLINE が表示されます。

```
+---------------------------+--------------------------------------
+----------------+-------------+--------------+-------------+----------------
+----------------------------+
```

```
| CHANNEL_NAME               | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+---------------
+----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+---------------
+----------------------------+
3 rows in set (0.00 sec)
```

可能な MEMBER_STATE 値の詳細については、MySQL ドキュメントの「グループレプリケーションサーバーの状態」を参照してください。

ステップ 7: (オプション) アクティブ/アクティブクラスターの DB インスタンスにデータをインポートする

MySQL データベースからアクティブ/アクティブクラスターの DB インスタンスにデータをインポートできます。データがインポートされると、グループレプリケーションはクラスター内の他の DB インスタンスにレプリケートします。

データのインポートの詳細については、「ダウンタイムを短縮して Amazon RDS for MySQL データベースにデータをインポートする」を参照してください。

DB インスタンスをアクティブ/アクティブクラスターに追加する

DB インスタンスを Amazon RDS for MySQL アクティブ/アクティブクラスターに追加するには、DB スナップショットを復元するか、DB インスタンスを特定の時点に復元します。アクティブ/アクティブクラスターには、最大 9 つの DB インスタンスを含めることができます。

DB インスタンスを特定の時点に復元する場合、通常、DB スナップショットから復元された DB インスタンスよりも最近のトランザクションが含まれます。DB インスタンスに最新のトランザクションがある場合、レプリケーションを開始するときに適用するトランザクションの数を減らす必要があります。したがって、ポイントインタイムリカバリを使用して DB インスタンスをクラスターに追加する方が、通常、DB スナップショットから復元するよりも高速です。

トピック

- ポイントインタイムリカバリを使用してアクティブ/アクティブクラスターに DB インスタンスを追加する
- DB インスタンスをアクティブ/アクティブクラスターに追加する

ポイントインタイムリカバリを使用してアクティブ/アクティブクラスターに DB インスタンスを追加する

アクティブ/アクティブクラスターに DB インスタンスを追加するには、クラスター内の DB インスタンスでポイントインタイムリカバリを実行します。

DB インスタンスを別の AWS リージョン の特定時点に復旧する方法については、「別の AWS リージョン への自動バックアップのレプリケーション」を参照してください。

ポイントインタイムリカバリを使用してアクティブ/アクティブクラスターに DB インスタンスを追加するには

1. アクティブ/アクティブクラスター内の DB インスタンスでポイントインタイムリカバリを実行することによって、新しい DB インスタンスを作成します。

   クラスター内の任意の DB インスタンスに対してポイントインタイムリカバリを実行して、新しい DB インスタンスを作成できます。手順については、Amazon RDS の DB インスタンスを特定の時点に復元する を参照してください。

   > **Important**
   >
   > ポイントインタイムリカバリ中に、新しい DB インスタンスを、アクティブ/アクティブクラスターパラメータが設定されている DB パラメータグループに関連付けます。そうしないと、新しい DB インスタンスでグループレプリケーションが開始されません。パラメータとそれぞれの必須パラメータの詳細については、「アクティブ/アクティブクラスターに必要なパラメータ設定」を参照してください。

> **Tip**
>
> ポイントインタイムリカバリを開始する前に DB インスタンスのスナップショットを作成すると、新しい DB インスタンスにトランザクションを適用するのに必要な時間を短縮できる場合があります。

2. 新しい DB インスタンスに関連付けた DB パラメータグループを含め、アクティブ/アクティブクラスター内の DB インスタンスに関連付けられた各 DB パラメータグループの `group_replication_group_seeds` パラメータに DB インスタンスを追加します。

   パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

3. SQL クライアントで、新しい DB インスタンスに接続し、mysql.rds_group_replication_set_recovery_channel ストアドプロシージャを呼び出します。*`group_replication_user_password`* を `rdsgrprepladmin` ユーザーのパスワードに置き換えます。

   ```
   call
    mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
   ```

4. SQL クライアントを使用して、mysql.rds_group_replication_start ストアドプロシージャを呼び出してレプリケーションを開始します。

   ```
   call mysql.rds_group_replication_start(0);
   ```

DB インスタンスをアクティブ/アクティブクラスターに追加する

DB インスタンスをアクティブ/アクティブクラスターに追加するには、クラスター内の DB インスタンスの DB スナップショットを作成し、DB スナップショットを復元します。

スナップショットを別の AWS リージョン にコピーする詳細については、「the section called “リージョン間のコピー”」を参照してください。

DB スナップショットを使用して、DB インスタンスをアクティブ/アクティブクラスターに追加するには

1. アクティブ/アクティブクラスターに DB インスタンスの DB スナップショットを作成します。

クラスター内の任意の DB インスタンスの DB スナップショットを作成することができます。手順については、Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成を参照してください。

2. DB スナップショットから DB インスタンスを復元します。

   スナップショット復元操作時に、新しい DB インスタンスを、アクティブ/アクティブクラスターパラメータが設定されている DB パラメータグループに関連付けます。パラメータとそれぞれの必須パラメータの詳細については、「アクティブ/アクティブクラスターに必要なパラメータ設定」を参照してください。

   DB スナップショットから DB インスタンスを復元する方法については、「DB インスタンスへの復元」を参照してください。

3. 新しい DB インスタンスに関連付けた DB パラメータグループを含め、アクティブ/アクティブクラスター内の DB インスタンスに関連付けられた各 DB パラメータグループの `group_replication_group_seeds` パラメータに DB インスタンスを追加します。

   パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

4. SQL クライアントで、新しい DB インスタンスに接続し、mysql.rds_group_replication_set_recovery_channel ストアドプロシージャを呼び出します。*`group_replication_user_password`* を `rdsgrprepladmin` ユーザーのパスワードに置き換えます。

```
call
 mysql.rds_group_replication_set_recovery_channel('group_replication_user_password');
```

5. SQL クライアントを使用して、mysql.rds_group_replication_start ストアドプロシージャを呼び出してレプリケーションを開始します。

```
call mysql.rds_group_replication_start(0);
```

アクティブ/アクティブクラスターのモニタリング

Amazon RDS for MySQL のアクティブ/アクティブクラスターのモニタリングは、パフォーマンス、レプリケーションの整合性、ノード同期を追跡するために不可欠です。アクティブ/アクティブクラ

スターをモニタリングするには、クラスター内の DB インスタンスに接続し、次の SQL コマンドを実行します。

```
SELECT * FROM performance_schema.replication_group_members;
```

出力には、次のサンプル出力のように、各 DB インスタンスの MEMBER_STATE の ONLINE が表示されます。

```
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+-----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST    |
 MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK
 |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+-----------------------------+
| group_replication_applier | 9854d4a2-5d7f-11ee-b8ec-0ec88c43c251 | ip-10-15-3-137 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | 9e2e9c28-5d7f-11ee-8039-0e5d58f05fef | ip-10-15-3-225 |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
| group_replication_applier | a6ba332d-5d7f-11ee-a025-0a5c6971197d | ip-10-15-1-83  |
      3306 | ONLINE       | PRIMARY     | 8.0.35         | MySQL                       |
+---------------------------+--------------------------------------
+---------------+-------------+--------------+-------------+----------------
+-----------------------------+
3 rows in set (0.00 sec)
```

可能な MEMBER_STATE 値の詳細については、MySQL ドキュメントの「グループレプリケーションサーバーの状態」を参照してください。

アクティブ/アクティブクラスター内の DB インスタンスでのグループレプリケーションを停止する

アクティブ/アクティブクラスター内の DB インスタンスでのグループレプリケーションを停止できます。グループレプリケーションを停止すると、レプリケーションが再開されるか、アクティブ/アクティブクラスターから DB インスタンスが削除されるまで、DB インスタンスはスーパー読み取り専用モードになります。スーパー読み取り専用モードの詳細については、「MySQL ドキュメント」を参照してください。

アクティブ/アクティブクラスターのグループレプリケーションを一時的に停止するには

1. SQL クライアントを使用して、アクティブ/アクティブクラスター内の DB インスタンスに接続します。

   RDS for Oracle DB インスタンスへの接続の詳細については、「MySQL DB インスタンスへの接続」を参照してください。

2. SQL クライアントで、mysql.rds_group_replication_stop ストアドプロシージャを呼び出します。

```
call mysql.rds_group_replication_stop();
```

アクティブ/アクティブクラスター内の DB インスタンスの名前を変更する

アクティブ/アクティブクラスター内の DB インスタンスの名前を変更できます。アクティブ/アクティブクラスター内の複数の DB インスタンスの名前を変更するには、一度に 1 つの DB インスタンスの名前を変更します。そのため、1 つの DB インスタンスの名前を変更して、クラスターに再結合してから、次の DB インスタンスの名前を変更します。

アクティブ/アクティブクラスター内の DB インスタンスの名前を変更するには

1. SQL クライアントで DB インスタンスに接続し、mysql.rds_group_replication_stop ストアドプロシージャを呼び出します。

```
call mysql.rds_group_replication_stop();
```

2. 「DB インスタンスの名前を変更する」の手順に従って、DB インスタンスの名前を変更します。
3. アクティブ/アクティブクラスター内の DB インスタンスに関連付けられている各 DB パラメータグループの group_replication_group_seeds パラメータを変更します。

   パラメータ設定で、古い DB インスタンスエンドポイントを新しい DB インスタンスエンドポイントに置き換えます。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

4. SQL クライアントで DB インスタンスに接続し、mysql.rds_group_replication_start ストアドプロシージャを呼び出します。

```
call mysql.rds_group_replication_start(0);
```

アクティブ/アクティブクラスターから DB インスタンスを削除する

アクティブ/アクティブクラスターから DB インスタンスを削除すると、スタンドアロンの DB インスタンスに戻ります。

アクティブ/アクティブクラスターから DB インスタンスを削除するには

1. SQL クライアントで DB インスタンスに接続し、mysql.rds_group_replication_stop ストアドプロシージャを呼び出します。

   ```
   call mysql.rds_group_replication_stop();
   ```

2. アクティブ/アクティブクラスターに残っている DB インスタンスの group_replication_group_seeds パラメータを変更します。

   group_replication_group_seeds パラメータで、アクティブ/アクティブクラスターから削除する DB インスタンスを削除します。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。
3. アクティブ/アクティブクラスターから削除する DB インスタンスのパラメータを変更して、クラスターに含まれないようにします。

   DB インスタンスを別のパラメータグループに関連付けるか、DB インスタンスに関連付けられた DB パラメータグループのパラメータを変更できます。変更するパラメータには、group_replication_group_name、rds.group_replication_enabled、および group_replication_group_seeds が含まれます。アクティブ/アクティブクラスターパラメータの詳細については、「アクティブ/アクティブクラスターに必要なパラメータ設定」を参照してください。

   DB パラメータグループのパラメータを変更する場合は、DB パラメータグループがアクティブ/アクティブクラスター内の他の DB インスタンスに関連付けられていないことを確認します。
4. アクティブ/アクティブクラスターから削除した DB インスタンスを再起動して、新しいパラメータ設定を有効にします。

   手順については、「 DB インスタンスの再起動」を参照してください。

レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート

MySQL DB インスタンスから Amazon RDS の外部で実行されている MySQL インスタンスにデータをエクスポートするには、レプリケーションを使用できます。このシナリオでは、MySQL DB インスタンスはソース MySQL DB インスタンスであり、Amazon RDS 外部で実行されている MySQL インスタンスは外部 MySQL データベースです。

外部 MySQL データベースは、データセンターでオンプレミスで実行することも、Amazon EC2 インスタンスで実行することもできます。外部 MySQL データベースは、ソース MySQL DB インスタンスと同じバージョン、またはそれ以降のバージョンを実行する必要があります。

外部 MySQL データベースへのレプリケーションは、ソース MySQL DB インスタンスからデータベースをエクスポートするのにかかる時間のみサポートされます。レプリケーションは、データがエクスポートされ、アプリケーションが外部 MySQL インスタンスへのアクセスを開始できるようになったら終了する必要があります。

このプロセスは以下のステップで構成されます。各ステップについては、後のセクションで詳しく説明します。

1. 外部 MySQL DB インスタンスを準備します。
2. レプリケーション用にソース MySQL DB インスタンスを準備します。
3. mysqldump ユーティリティを使用して、ソース MySQL DB インスタンスから外部 MySQL データベースにデータベースを転送します。
4. 外部 MySQL データベースへのレプリケーションを開始します。
5. エクスポートが完了したら、レプリケーションを停止します。

外部の MySQL データベースの準備

次のステップを実行して、外部 MySQL データベースを準備します。

外部の MySQL データベースを準備するには

1. 外部 MySQL データベースをインストールします。
2. マスターユーザーとして外部 MySQL データベースに接続します。次に、データベースにアクセスする管理者、アプリケーション、およびサービスのサポートに必要なユーザーを作成します。

3. MySQL ドキュメントの手順に従って、外部 MySQL データベースをレプリカとして準備します。詳細については、MySQL ドキュメントの「Setting the Replica Configuration」を参照してください。
4. エクスポート中にリードレプリカとして動作する外部 MySQL データベースの Egress ルールを設定します。Egress ルールにより、レプリケーション中に外部 MySQL データベースがソース MySQL DB インスタンスに接続できるようになります。ソース MySQL DB インスタンスのポートおよび IP アドレスへの Transmission Control Protocol (TCP) 接続を許可する Egress ルールを指定します。

   環境に適した Egress ルールを指定します。

   - 外部 MySQL データベースが、Amazon VPC サービスに基づく Virtual Private Cloud (VPC) の Amazon EC2 インスタンスで実行されている場合は、VPC セキュリティグループで Egress ルールを指定します。(詳しくは、「セキュリティグループによるアクセス制御」を参照してください。)
   - 外部 MySQL データベースがオンプレミスでインストールされている場合は、ファイアウォールで Egress ルールを指定します。
5. 外部 MySQL データベースが VPC で実行されている場合は、セキュリティグループの Egress ルールに加えて、VPC アクセスコントロールリスト (ACL) のルールを設定します。

   - ACL Ingress ルールを設定し、ソース MySQL DB インスタンスの IP アドレスからポート 1024–65535 への TCP トラフィックを許可します。
   - ACL Egress ルールを設定し、ソース MySQL DB インスタンスのポートおよび IP アドレスへのアウトバウンド TCP トラフィックを許可します。

   Amazon VPC ネットワーク ACL の詳細については、Amazon VPC ユーザーガイドの「ネットワーク ACL」を参照してください。
6. (オプション) レプリケーションエラーを回避するために、`max_allowed_packet` パラメータを最大サイズに設定します。この設定をお勧めします。

ソース MySQL DB インスタンスの準備

以下のステップを実行して、レプリケーションソースとしてソース MySQL DB インスタンスを準備します。

ソース MySQL DB インスタンスを準備するには

1. レプリケーションのセットアップ中にバイナリログを保存するのに十分なディスク容量がクライアントコンピュータにあることを確認します。
2. ソース MySQL DB インスタンスに接続し、MySQL ドキュメントの「Creating a User for Replication」の手順に従ってレプリケーションアカウントを作成します。
3. レプリケーション中に外部 MySQL データベースが接続できるように、ソース MySQL DB インスタンスを実行しているシステムで Ingress ルールを設定します。外部の MySQL データベースの IP アドレスからソース MySQL DB インスタンスによって使用されるポートへの TCP 接続を許可する Ingress ルールを指定します。
4. Egress ルールを指定します。

   - ソース MySQL DB インスタンスが VPC で実行されている場合、VPC セキュリティグループで Ingress ルールを指定します。(詳しくは、「セキュリティグループによるアクセス制御」を参照してください。)
5. ソース MySQL DB インスタンスが VPC で実行されている場合、セキュリティグループの Ingress ルールに加えて VPC ACL ルールを設定します。

   - ACL Ingress ルールを設定し、外部 MySQL データベースの IP アドレスから Amazon RDS インスタンスにより使用されるポートへの TCP 接続を許可します。
   - ACL Egress ルールを設定し、ポート 1024–65535 から外部 MySQL データベースの IP アドレスへの TCP 接続を許可します。

   Amazon VPC ネットワーク ACL の詳細については、Amazon VPC ユーザーガイドの「ネットワーク ACL」を参照してください。
6. バックアップ保持期間について、エクスポート中にバイナリログが消去されない十分な長さに設定されていることを確認します。エクスポートが完了する前にいずれかのログが消去された場合、レプリケーションを最初から再開しなければなりません。バックアップ保持期間の設定の詳細については、「バックアップの概要」を参照してください。
7. `mysql.rds_set_configuration` ストアドプロシージャを使用して、バイナリログの保持期間を、エクスポート中にバイナリログが消去されない十分な長さに設定します。詳細については、「MySQL バイナリログにアクセスする」を参照してください。
8. ソース MySQL DB インスタンスのバイナリログが消去されないことをさらに確実にするため、ソース MySQL DB インスタンスから Amazon RDS リードレプリカを作成します。詳細については、「リードレプリカの作成」を参照してください。

9. Amazon RDS リードレプリカが作成されたら、mysql.rds_stop_replication ストアドプロシージャを呼び出してレプリケーションプロセスを停止します。ソース MySQL DB インスタンスはバイナリログファイルを消去しなくなり、レプリケーションプロセスで使用できるようになります。
10. (オプション) レプリケーションエラーを回避するために、max_allowed_packet パラメータと slave_max_allowed_packet の両方を最大サイズに設定します。両方のパラメータの最大サイズは、1 GB です。両方のパラメータに対して、この設定をお勧めします。パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

データベースのコピー

データベースをコピーするには、次のステップを実行します。

データベースをコピーするには

1. ソース MySQL DB インスタンスの RDS リードレプリカに接続し、MySQL SHOW REPLICA STATUS\G ステートメントを実行します。次の値に注意してください。
   - Master_Host
   - Master_Port
   - Master_Log_File
   - Exec_Master_Log_Pos

   > **Note**
   >
   > MySQL の旧バージョンは、SHOW REPLICA STATUS ではなく SHOW SLAVE STATUS を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、SHOW SLAVE STATUS を使用します。

2. mysqldump ユーティリティを使用してスナップショットを作成します。Amazon RDS からローカルクライアントコンピュータにデータがコピーされます。レプリケートするデータベースにある mysqldump ファイルを保存するのに十分な容量がクライアントコンピュータにあることを確認します。非常に大きなデータベースでは、このプロセスに数時間かかる可能性があります。MySQL ドキュメントの「Creating a Data Snapshot Using mysqldump」の手順に従います。

次の例では、クライアントで mysqldump を実行し、ダンプをファイルに書き込みます。

Linux、macOS、Unix の場合:

```
mysqldump -h source_MySQL_DB_instance_endpoint \
    -u user \
    -ppassword \
    --port=3306 \
    --single-transaction \
    --routines \
    --triggers \
    --databases  database database2 > path/rds-dump.sql
```

Windows の場合:

```
mysqldump -h source_MySQL_DB_instance_endpoint ^
    -u user ^
    -ppassword ^
    --port=3306 ^
    --single-transaction ^
    --routines ^
    --triggers ^
    --databases  database database2 > path\rds-dump.sql
```

バックアップファイルを外部 MySQL データベースにロードすることができます。詳細については、MySQL ドキュメントの「SQL 形式のバックアップの再ロード」を参照してください。別のユーティリティを実行して、データを外部 MySQL データベースにロードします。

エクスポートの完了

エクスポートを完了するには、次のステップを実行します。

エクスポートを完了するには

1. MySQL CHANGE MASTER ステートメントを使用し、外部 MySQL データベースを設定します。REPLICATION SLAVE アクセス許可を付与されたユーザーの ID とパスワードを指定します。RDS リードレプリカで実行する MySQL Master_Host ステートメントから取得した Master_Port、Relay_Master_Log_File、Exec_Master_Log_Pos、SHOW REPLICA

STATUS\G の値を指定します。詳細については、MySQL ドキュメントの「CHANGE MASTER TO Statement」を参照してください。

> **Note**
>
> MySQL の旧バージョンは、SHOW REPLICA STATUS ではなく SHOW SLAVE STATUS を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、SHOW SLAVE STATUS を使用します。

2. MySQL START REPLICA コマンドを使用して、ソース MySQL DB インスタンスから外部 MySQL データベースへのレプリケーションを開始します。

   これにより、ソース MySQL DB インスタンスからレプリケーションが開始され、Amazon RDS リードレプリカからレプリケーションを停止した後に発生したすべてのソース変更がエクスポートされます。

   > **Note**
   >
   > MySQL の旧バージョンは、START REPLICA ではなく START SLAVE を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、START SLAVE を使用します。

3. 外部 MySQL データベースで MySQL SHOW REPLICA STATUS\G コマンドを実行して、リードレプリカとして動作していることを確認します。詳細については、MySQL ドキュメントの「SHOW SLAVE | REPLICA STATUS Statement」を参照してください。
4. 外部 MySQL データベースでのレプリケーションがソース MySQL DB インスタンスに追いついたら、MySQL STOP REPLICA コマンドを使用して、ソース MySQL DB インスタンスからのレプリケーションを停止します。

   > **Note**
   >
   > MySQL の旧バージョンは、STOP REPLICA ではなく STOP SLAVE を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、STOP SLAVE を使用します。

5. Amazon RDS リードレプリカで、mysql.rds_start_replication ストアドプロシージャを呼び出します。これにより、Amazon RDS がソース MySQL DB インスタンスからのバイナリログファイルの消去を開始できるようになります。

MySQL DB インスタンスのオプション

ここでは、MySQL DB エンジンを実行する Amazon RDS インスタンスで使用できるオプションまたは追加機能について説明します。これらのオプションを有効にするには、カスタムオプショングループにオプションを追加して、そのオプショングループを DB インスタンスに関連付けます。オプショングループの操作方法の詳細については、「オプショングループを使用する」を参照してください。

Amazon RDS では、以下の MySQL 用オプションがサポートされています。

オプション	オプション ID	エンジンバージョン
MySQL に対する MariaDB 監査プラグインのサポート	MARIADB_AUDIT_PLUGIN	すべての MySQL 8.4 バージョン MySQL のバージョン 8.0 (8.0.26 以降) すべての MySQL 5.7 バージョン
MySQL の memcached サポート	MEMCACHED	すべての MySQL 5.7 および 8.0 バージョン

MySQL に対する MariaDB 監査プラグインのサポート

Amazon RDS は、オープンソースの MariaDB 監査プラグインに基づいた MySQL データベースインスタンス用の監査プラグインを提供しています。詳細については、「MySQL サーバー GitHub リポジトリの監査プラグイン」を参照してください。

> **Note**
>
> MySQL の監査プラグインは MariaDB 監査プラグインに基づいています。この記事では、このプラグインを「MariaDB 監査プラグイン」と呼びます。

MariaDB 監査プラグインは、データベースへのユーザーのログオンやデータベースに対して実行されたクエリなどのデータベースアクティビティを記録します。データベースのアクティビティのレコードはログファイルに保存されます。

監査プラグインのオプション設定

Amazon RDS では、MariaDB 監査プラグインのオプションの次の設定がサポートされています。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	ログファイルの場所。ログファイルには、SERVER_AUDIT_EVENTS で指定されたアクティビティのレコードが含まれます。詳細については、「データベースログファイルの表示とリスト化」および「MySQL データベースのログファイル」を参照してください。
SERVER_AUDIT_FILE_ROTATE_SIZE	1-1000000000	1000000	このバイト数のサイズに達するとファイルがローテーションします。詳細については、「RDS for MySQL データベースログの概要」を参照してください。
SERVER_AUDIT_FILE_	0-100	9	server_audit_output_type=file 時に保存するログローテーション数。0 に設定すると、ログファイルはローテーションされませ

オプション設定	有効な値	デフォルト値	説明
ROTATIONS			ん。詳細については、「RDS for MySQL データベースログの概要」および「データベースログファイルのダウンロード」を参照してください。
SERVER_AUDIT_EVENTS	CONNECT, QUERY, QUERY_DDL , QUERY_DML , QUERY_DML_NO_SELECT , QUERY_DCL	CONNECT, QUERY	ログに記録するアクティビティのタイプ。MariaDB 監査プラグインのインストール自体も記録されます。 • CONNECT: データベースへ接続の成功と失敗、およびデータベースからの切断を記録します。 • QUERY: データベースに対して実行されたすべてのクエリのテキストを記録します。 • QUERY_DDL : QUERY イベントと同様ですが、返るのは、データ定義言語 (DDL) クエリ (CREATE、ALTER など) のみです。 • QUERY_DML : QUERY イベントと同様ですが、返るのは、データ操作言語 (DML) クエリ (INSERT、UPDATE、SELECT など) のみです。 • QUERY_DML_NO_SELECT : QUERY_DML イベントと類似していますが、SELECT クエリを口グ記録しません。 • QUERY_DCL : QUERY イベントと同様ですが、返るのは、データ制御言語 (DCL) クエリ (GRANT、REVOKE など) のみです。 MySQL では、TABLE はサポートされていません。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_INCL_USERS	複数のカンマ区切り値	なし	指定されたユーザーからのアクティビティのみを含めます。デフォルトでは、アクティビティはすべてのユーザーについて記録されます。SERVER_AUDIT_INCL_USERS と SERVER_AUDIT_EXCL_USERS は相互に排他的です。SERVER_AUDIT_INCL_USERS に値を追加する場合は、SERVER_AUDIT_EXCL_USERS に追加される値がないことを確認してください。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_EXCL_USERS	複数のカンマ区切り値	なし	指定されたユーザーからのアクティビティを除外します。デフォルトでは、アクティビティはすべてのユーザーについて記録されます。SERVER_AUDIT_INCL_USERS と SERVER_AUDIT_EXCL_USERS は相互に排他的です。SERVER_AUDIT_EXCL_USERS に値を追加する場合は、SERVER_AUDIT_INCL_USERS に追加される値がないことを確認してください。 rdsadmin ユーザーは 1 秒ごとにデータベースをクエリしてデータベースのヘルスチェックを行います。そのほかの設定によっては、このアクティビティによってログファイルのサイズが急激に増大する可能性があります。このアクティビティを記録する必要がない場合は、SERVER_AUDIT_EXCL_USERS リストにrdsadminユーザーを追加します。 **Note** CONNECT アクティビティは、ユーザーがこのオプション設定で指定されていても、すべてのユーザーについて常に記録されます。
SERVER_AUDIT_LOGGING	ON	ON	ログ記録がアクティブです。唯一の有効な値は ON です。Amazon RDS では、ログ記録の非アクティブ化はサポートしていません。ログ記録を非アクティブ化する場合は、MariaDB 監査プラグインを削除します。詳細については、「MariaDB 監査プラグインの削除」を参照してください。

オプション設定	有効な値	デフォルト値	説明
SERVER_AUDIT_QUERY_LOG_LIMIT	0-2147483647	1024	レコードのクエリ文字列の長さに対する制限。

MariaDB 監査プラグインの追加

MariaDB 監査プラグインを DB インスタンスに追加する一般的な手順は以下のとおりです。

- 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更する
- オプショングループにオプションを追加する
- オプショングループを DB インスタンスに関連付ける

MariaDB 監査プラグインを追加した後で、DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、直ちに監査がスタートされます。

> **Important**
>
> MariaDB 監査プラグインを DB インスタンスに追加すると、停止する可能性があります。MariaDB 監査プラグインは、メンテナンス期間中またはデータベースのワークロードが低い時間帯に追加することをお勧めします。

MariaDB 監査プラグインを追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。それ以外の場合は、カスタム DB オプショングループを作成します。[エンジン] で [MySQL] を選択し、[メジャーエンジンバージョン] で [5.7]、[8.0]、または [8.4] を選択します。詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [MARIADB_AUDIT_PLUGIN] オプションを追加し、オプションを設定します。オプションの追加方法の詳細については、「オプショングループにオプションを追加す

る」を参照してください。各設定の詳細については、「監査プラグインのオプション設定」を参照してください。

3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

監査ログの形式

ログファイルは、UTF-8 形式のカンマ区切り変数 (CSV) ファイルとして表されます。

> **Tip**
>
> ログファイルのエントリは、順番になっていません。エントリを順序付けするには、タイムスタンプ値を使用します。最新のイベントを表示するには、すべてのログファイルの確認が必要な場合があります。ログデータの並べ替えと検索をより柔軟に行うためには、監査ログを CloudWatch にアップロードするための設定を有効にし、CloudWatch インターフェイスを使用してそれらを表示します。
> より多くのタイプのフィールドを含み、JSON 形式で出力された監査データを表示するには、データベースのアクティビティストリーム機能を使用することもできます。詳細については、「データベースアクティビティストリームを使用した Amazon RDS のモニタリング」を参照してください。

監査ログファイルの行には、次のカンマ区切りの情報が指定された順序で含まれています。

フィールド	説明
timestamp	YYYYMMDD の後、ログに記録されたイベントの HH:MI:SS (24 時間制) が続きます。
serverhost	イベントが記録されているインスタンスの名前。
username	ユーザーの接続されたユーザー名。

フィールド	説明
host	ユーザーの接続元のホスト。
connectionid	記録されたオペレーションの接続 ID 番号。
queryid	クエリ ID 番号。リレーショナルテーブルイベントと関連するクエリの検索に使用できます。TABLE イベントの場合、複数の行が追加されます。
オペレーション	記録されたアクションの種類。指定できる値は CONNECT、QUERY、READ、WRITE、CREATE、ALTER、RENAME、DROP です。
データベース	USE コマンドにより設定されたアクティブなデータベース。
オブジェクト	QUERY イベントの場合、この値は、データベースが実行したクエリを示します。TABLE イベントの場合、テーブル名を示します。
retcode	記録されたオペレーションのリターンコード。
connection_type	サーバーへの接続のセキュリティ状態です。可能な値は以下のとおりです。 • Undefined • TCP/IP • Socket • Named pipe • SSL/TLS • Shared memory

MariaDB 監査プラグインのログの表示とダウンロード

MariaDB 監査プラグインを有効にした後は、他のテキストベースのログファイルと同様の方法でログファイル内の結果にアクセスします。監査ログファイルは /rdsdbdata/log/audit/ にあります。コンソールでログファイルを表示する方法の詳細については、「データベースログファイルの表示とリスト化」を参照してください。ログファイルのダウンロードについては、「データベースログファイルのダウンロード」を参照してください。

MariaDB 監査プラグインの設定の変更

MariaDB 監査プラグインを有効にした後、設定を変更できます。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。各設定の詳細については、「監査プラグインのオプション設定」を参照してください。

MariaDB 監査プラグインの削除

Amazon RDS では、MariaDB 監査プラグインの口グ記録の無効化はサポートされていません。ただし、DB インスタンスからプラグインを削除することはできます。MariaDB 監査プラグインを削除すると、DB インスタンスが自動的に再起動され、監査が停止します。

MariaDB 監査プラグインを DB インスタンスから削除するには、次のいずれかを実行します。

- MariaDB 監査プラグインが所属するオプショングループからプラグインを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、プラグインが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

MySQL の memcached サポート

Amazon RDS は、MySQL 5.6 で導入された InnoDB テーブルに対する memcached インターフェイスの使用をサポートしています。memcached API を使用すると、NoSQL キー/値データストアと似た方法でアプリケーションが InnoDB テーブルを使用することができます。

> **Note**
>
> memcached インターフェイスは、MySQL 8.4 では使用できなくなりました。DB インスタンスを MySQL 8.4 にアップグレードするときは、既存のオプショングループで memcached を無効にする必要があります。

memcached インターフェイスは、シンプルなキーベースのキャッシュです。アプリケーションは memcached を使用して、キャッシュにあるキーと値のデータペアの挿入、操作、取得を実行します。MySQL 5.6 では、memcached プロトコルによって InnoDB テーブルのデータを発行するデーモンサービスを実装するプラグインが導入されました。MySQL memcached プラグインの詳細については、「InnoDB と memcached の統合」を参照してください。

RDS for MySQL DB インスタンスの memcached サポートを有効にするには

1. memcached インターフェイスへのアクセスを制御するために使用するセキュリティグループを決定します。既に SQL インターフェイスを使用しているアプリケーションセットが memcached インターフェイスにアクセスするアプリケーションセットと同じ場合、SQL インターフェイスで使用されている既存の VPC セキュリティグループを使用できます。異なるアプリケーションセットが memcached インターフェイスにアクセスする場合は、新しい VPC または DB セキュリティグループを定義します。セキュリティグループの管理方法の詳細については、「セキュリティグループによるアクセス制御」を参照してください。
2. カスタム DB オプショングループを作成し、エンジンタイプとバージョンとして MySQL を選択します。オプショングループの作成方法の詳細については、「オプショングループを作成する」を参照してください。
3. オプショングループに [MEMCACHED] オプションを追加します。memcached インターフェイスで使用するポート、および、インターフェイスへのアクセスを制御するために使用するセキュリティグループを指定します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
4. 必要に応じて、オプション設定を変更し、memcached パラメータを設定します。オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。

5. インスタンスにオプショングループを適用します。オプショングループが適用された場合、Amazon RDS では、そのインスタンスの memcached サポートが有効になります。

   - インスタンスを起動するときにカスタムオプショングループを指定して、新しいインスタンスの memcached サポートを有効にします。MySQL インスタンスの起動方法の詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - インスタンスを変更するときにカスタムオプショングループを指定して、既存のインスタンスの memcached サポートを有効にします。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

6. memcached インターフェイスを介して MySQL テーブル内のアクセス可能な列を指定します。memcached プラグインは、containers という専用データベースに innodb_memcache というカタログテーブルを作成します。containers テーブルに行を挿入して InnoDB テーブルをマッピングし、memcached を介してアクセスします。memcached キー値を格納する InnoDB テーブルの列を 1 つ指定し、キーに関連付けられたデータ値を格納する列を 1 つ以上指定します。また、memcached アプリケーションがその列セットを参照するときに使用する名前も指定します。containers テーブルに行を挿入する方法の詳細については、「InnoDB memcached プラグインの内部構造」を参照してください。InnoDB テーブルをマッピングし、memcached を介してアクセスする例については、「InnoDB memcached プラグインの書き込みアプリケーション」を参照してください。

7. memcached インターフェイスにアクセスするアプリケーションが、SQL インターフェイスを使用するアプリケーションとは異なるコンピュータまたは EC2 インスタンス上にある場合は、MySQL インスタンスに関連付けられた VPC セキュリティグループに、そのコンピュータの接続情報を追加します。セキュリティグループの管理方法の詳細については、「セキュリティグループによるアクセス制御」を参照してください。

インスタンスの memcached サポートを無効にするには、インスタンスを変更して MySQL バージョンのデフォルトオプショングループを指定します。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

MySQL memcached のセキュリティ上の考慮事項

memcached プロトコルはユーザー認証をサポートしていません。MySQL memcached セキュリティ上の考慮事項の詳細については、MySQL ドキュメントの「InnoDB memcached プラグインのセキュリティ上の考慮事項」を参照してください。

以下の対策をとると、memcached インターフェイスのセキュリティを高めることができます。

- MEMCACHED オプションをオプショングループに追加するときに、デフォルトの 11211 とは異なるポートを指定します。
- 既知の信頼されたクライアントアドレスおよび EC2 インスタンスのみにアクセスを制限する VPC セキュリティグループに、memcached インターフェイスを関連付けます。セキュリティグループの管理方法の詳細については、「セキュリティグループによるアクセス制御」を参照してください。

MySQL memcached の接続情報

memcached インターフェイスにアクセスするには、アプリケーションで Amazon RDS インスタンスの DNS 名と memcached ポート番号の両方を指定する必要があります。例えば、インスタンスの DNS 名が my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com で、memcached インターフェイスがポート 11212 を使用している場合、PHP で指定する接続情報は次のようになります。

```
<?php

$cache = new Memcache;
$cache->connect('my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com',11212);
?>
```

MySQL DB インスタンスの DNS 名と memcached ポートを確認するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. AWS マネジメントコンソールで、DB インスタンスを含むリージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. MySQL DB インスタンスの名前を選択して詳細を表示します。
5. [Connect] セクションで、[Endpoint] フィールドの値を書き留めます。DNS 名はエンドポイントと同じです。また、[Connect] セクションのポートは memcached インターフェイスへのアクセスには使用されないことにご注意ください。
6. [Details] セクションで、[Option Group] フィールドにリストされた名前を書き留めます。
7. ナビゲーションペインで、[オプショングループ] を選択します。

8. MySQL DB インスタンスで使用するオプショングループの名前を選択して、オプショングループの詳細を表示します。[Options] セクションで、[MEMCACHED] オプションの [Port] 設定の値を書き留めます。

MySQL memcached のオプション設定

Amazon RDS は、MySQL memcached パラメータを Amazon RDS MEMCACHED オプションのオプション設定として公開します。

MySQL memcached のパラメータ

- DAEMON_MEMCACHED_R_BATCH_SIZE - COMMIT を実行して新しいトランザクションをスタートする前に実行する memcached 読み取りオペレーション (get) の回数を指定する整数値。指定できる値は 1〜4294967295 で、デフォルトは 1 です。このオプションは、インスタンスが再開されるまで有効になりません。
- DAEMON_MEMCACHED_W_BATCH_SIZE - COMMIT を実行して新しいトランザクションをスタートする前に実行する memcached 書き込み操作 (add、set、incr など) の回数を指定する整数値。指定できる値は 1〜4294967295 で、デフォルトは 1 です。このオプションは、インスタンスが再開されるまで有効になりません。
- INNODB_API_BK_COMMIT_INTERVAL - InnoDB memcached インターフェイスを使用するアイドル状態の接続を自動コミットする頻度を指定する整数値。指定できる値は 1〜1073741824 で、デフォルトは 5 です。このオプションは即座に反映され、インスタンスを再開する必要はありません。
- INNODB_API_DISABLE_ROWLOCK - InnoDB memcached インターフェイスを使用しているときに行ロックの使用を無効 (1=true) または有効 (0=false) にするブール値。デフォルトは 0 (false) です。このオプションは、インスタンスが再開されるまで有効になりません。
- INNODB_API_ENABLE_MDL - ブール値。0 (false) に設定すると、InnoDB memcached プラグインで使用するテーブルがロックされ、SQL インターフェイスを介して DDL によってそのテーブルを削除または変更できなくなります。デフォルトは 0 (false) です。このオプションは、インスタンスが再開されるまで有効になりません。
- INNODB_API_TRX_LEVEL - memcached インターフェイスで処理されるクエリのトランザクション分離レベルを指定する整数値。指定できる値は 0〜3 です。デフォルトは 0 です。このオプションは、インスタンスが再開されるまで有効になりません。

次の MySQL memcached のパラメータは Amazon RDS によって設定され、ユーザーが変更することはできません:

DAEMON_MEMCACHED_LIB_NAME、DAEMON_MEMCACHED_LIB_PATH、INNODB_API_ENABLE_BINLOG。M
管理者が daemon_memcached_options を使用して設定したパラメータは、Amazon RDS の個々の MEMCACHED オプション設定として使用できます。

MySQL daemon_memcached_options のパラメータ

- BINDING_PROTOCOL - 使用するバインディングプロトコルを指定する文字列。指定できる値は、auto、ascii、または binary です。デフォルトは auto で、サーバーが自動的にクライアントとプロトコルを交渉します。このオプションは、インスタンスが再開されるまで有効になりません。
- BACKLOG_QUEUE_LIMIT - による処理待ちが可能なネットワーク接続の数を指定する整数値。memcachedこの値を増やすと、クライアントが memcached インスタンスに接続できないという エラーの発生回数が減る可能性がありますが、サーバーのパフォーマンスは向上しません。指定できる値は 1〜2048 で、デフォルトは 1024 です。このオプションは、インスタンスが再開されるまで有効になりません。
- CAS_DISABLED - 比較と交換 (CAS: compare and swap) の使用を有効 (1=true) または無効 (0=false) にするブール値。項目ごとのサイズが 8 バイトずつ小さくなります。デフォルトは 0 (false) です。このオプションは、インスタンスが再開されるまで有効になりません。
- CHUNK_SIZE - 最小項目のキー、値、およびフラグに割り当てる最小のチャンクサイズ (バイト単位) を指定する整数値。指定できる値は 1〜48 です。デフォルトは 48 で、値を小さくするとメモリ効率が大幅に向上します。このオプションは、インスタンスが再開されるまで有効になりません。
- CHUNK_SIZE_GROWTH_FACTOR - 新しいチャンクのサイズを制御する浮動小数点値。新しいチャンクのサイズは、前のチャンクのサイズに CHUNK_SIZE_GROWTH_FACTOR を掛けた値です。指定できる値は 1〜2 で、デフォルトは 1.25 です。このオプションは、インスタンスが再開されるまで有効になりません。
- ERROR_ON_MEMORY_EXHAUSTED - ブール値。1 (true) に設定すると、項目を格納するメモリが不足した場合、memcached は項目を削除するのではなく、エラーを返します。0 (false) に設定した場合、メモリ不足になると memcached は項目を削除します。デフォルトは 0 (false) です。このオプションは、インスタンスが再開されるまで有効になりません。
- MAX_SIMULTANEOUS_CONNECTIONS - 同時接続の最大数を指定する整数値。この値を 10 未満に設定すると、MySQL は起動できなくなります。指定できる値は 10〜1024 で、デフォルトは 1024 です。このオプションは、インスタンスが再開されるまで有効になりません。
- VERBOSITY - memcached サービスが MySQL エラーログに記録する情報のレベルを指定する文字列。デフォルトは「v」です。このオプションは、インスタンスが再開されるまで有効になりません。指定できる値は次のとおりです。

- v - メインイベントループの実行中に発生したエラーと警告を記録します。
- vv - v で記録する情報に加えて、各クライアントのコマンドとレスポンスも記録します。
- vvv - vv で記録する情報に加えて、内部の状態移行も記録します。

これらの MySQL DAEMON_MEMCACHED_OPTIONS のパラメータは
Amazon RDS によって設定され、ユーザーが変更することはできません:
DAEMON_PROCESS、LARGE_MEMORY_PAGES、MAXIMUM_CORE_FILE_LIMIT、MAX_ITEM_SIZE、LOCK_D

MySQL のパラメータ

デフォルトでは、MySQL DB インスタンスは MySQL データベースに固有の DB パラメータグループを使用します。このパラメータグループには、MySQL データベースエンジンのパラメータが含まれています。パラメータグループの操作とパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

RDS for MySQL パラメータは、選択したストレージエンジンのデフォルト値に設定されます。MySQL パラメータの詳細については、MySQL のドキュメントを参照してください。MySQL ストレージエンジンの詳細については、「RDS for MySQL のサポートされているストレージエンジン」を参照してください。

RDS コンソールまたは AWS CLI を使用して、特定の RDS for MySQL バージョンで使用可能なパラメータを表示できます。RDS コンソールで MySQL パラメータグループ内のパラメータを表示する方法については、「Amazon RDS のDB パラメータグループのパラメータ値の表示」を参照してください。

AWS CLI を使用して `describe-engine-default-parameters` コマンドを実行すると、RDS for MySQL バージョンのパラメータを表示できます。`--db-parameter-group-family` オプションには、次の値のうち 1 つを指定します。

- `mysql8.4`
- `mysql8.0`
- `mysql5.7`

例えば RDS for MySQL バージョン 8.0 のパラメータを表示するには、次のコマンドを実行します。

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0
```

出力は次のようになります。

```
{
    "EngineDefaults": {
        "Parameters": [
            {
                "ParameterName": "activate_all_roles_on_login",
                "ParameterValue": "0",
                "Description": "Automatically set all granted roles as active after the
 user has authenticated successfully.",
```

```
                "Source": "engine-default",
                "ApplyType": "dynamic",
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": true
            },
            {
                "ParameterName": "allow-suspicious-udfs",
                "Description": "Controls whether user-defined functions that have only
 an xxx symbol for the main function can be loaded",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": false
            },
            {
                "ParameterName": "auto_generate_certs",
                "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
                "Source": "engine-default",
                "ApplyType": "static",
                "DataType": "boolean",
                "AllowedValues": "0,1",
                "IsModifiable": false
            },
        ...
```

RDS for MySQL バージョン 8.0 の変更可能なパラメータのみを一覧表示するには、次のコマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0 \
   --query 'EngineDefaults.Parameters[?IsModifiable==`true`]'
```

Windows の場合:

```
aws rds describe-engine-default-parameters --db-parameter-group-family mysql8.0 ^
   --query "EngineDefaults.Parameters[?IsModifiable==`true`]"
```

MySQL DB インスタンスの一般的な DBA タスク

以下では、MySQL データベースエンジンを実行する DB インスタンスに関する一般的な DBA タスクの Amazon RDS 固有の実装について説明します。マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、アドバンストの特権を必要とする特定のシステムの手順やテーブルへのアクセスも制限されいます。

Amazon RDS での MySQL ログファイルの操作に関する詳細は、「MySQL データベースのログファイル」を参照してください。

定義済みユーザーとは

Amazon RDS は、新しい RDS for MySQL DB インスタンスを使用して、複数の事前定義済みのユーザーを自動的に作成します。事前定義済みのロールとそのロールの権限は変更できません。このような事前定義済みのロールについて、権限の削除、権限名の変更、権限の変更を行うことはできません。上記の操作を実行しようとすると、エラーが発生します。

- rdsadmin – `superuser` 権限を持つ管理者がスタンドアロンの MySQL データベースで実行する管理タスクの多くを処理するために作成されるロールです。このユーザーは、多くの管理タスクのために RDS for MySQL によって内部的に使用されます。
- rdsrepladmin – RDS for MySQL DB インスタンスおよびクラスターでのレプリケーションアクティビティをサポートするために Amazon RDS によって内部的に使用されるユーザーです。

その他の一般的な DBA タスクの詳細については、以下のトピックを参照してください。

トピック

- RDS for MySQL のロールベースの権限モデル
- RDS for MySQL の動的権限
- RDS for MySQL のセッションまたはクエリの終了
- RDS for MySQL の現在のレプリケーションエラーのスキップ
- InnoDB テーブルスペースの操作による RDS for MySQL のクラッシュリカバリ時間の短縮
- RDS for MySQL のグローバルステータス履歴の管理
- MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定

RDS for MySQL のロールベースの権限モデル

RDS for MySQL バージョン 8.0.36 以降では、mysql データベースのテーブルを直接変更することはできません。特に、grant テーブルに対してデータ操作言語 (DML) オペレーションを実行してデータベースユーザーを作成することはできません。代わりに、MySQL アカウント管理ステートメント (CREATE USER、GRANT、REVOKE など) を使用してロールベースの権限をユーザーに付与します。また、mysql データベースでストアドプロシージャなど、他の種類のオブジェクトを作成することはできません。mysql テーブルにクエリを実行することはできます。バイナリログのレプリケーションを使用する場合、ソース DB インスタンスの mysql テーブルに直接行った変更は、ターゲットクラスターにレプリケートされません。

場合によっては、mysql テーブルに挿入することで、アプリケーションがショートカットを使用して、ユーザーやその他のオブジェクトを作成する場合があります。その場合は、アプリケーションコードを変更して、CREATE USER などの対応したステートメントを使用します。

外部 MySQL データベースからの移行中にデータベースユーザーのメタデータをエクスポートするには、以下のいずれかの方法を使用します。

- MySQL Shell のインスタンスダンプユーティリティを、ユーザー、ロール、権限を除外するフィルターと共に使用します。次の例は、使用するコマンド構文を示しています。outputUrl が空であることを確認します。

  ```
  mysqlsh user@host -- util.dumpInstance(outputUrl,{excludeSchemas:['mysql'],users:
   true})
  ```

  詳細については、MySQL リファレンスマニュアルの「インスタンスダンプユーティリティ、スキーマダンプユーティリティ、およびテーブルダンプユーティリティ」を参照してください。
- mysqlpump クライアントユーティリティを使用します。この例には、mysql システムデータベース内のテーブルを除くすべてのテーブルが含まれています。これには移行されたデータベース内のすべての MySQL ユーザーを再現する CREATE USER や GRANT ステートメントも含まれます。

  ```
  mysqlpump --exclude-databases=mysql --users
  ```

  mysqlpump クライアントユーティリティは、MySQL 8.4 では使用できなくなりました。代わりに mysqldump を使用してください。

多くのユーザーまたはアプリケーションの権限の管理を簡素化するには、CREATE ROLE ステートメントを使用して、一連の権限を持つロールを作成します。その後、GRANT および SET ROLE ステートメントと current_role 関数を使用して、ユーザーまたはアプリケーションにロールを割り当てたり、現在のロールを切り替えたり、有効なロールをチェックしたりできます。MySQL 8.0 でのロールベースのアクセス許可システムの詳細については、MySQL リファレンスマニュアルのロールの使用を参照してください。

> **Important**
>
> アプリケーションではマスターユーザーを直接使用しないことを強くお勧めします。代わりに、アプリケーションに必要な最小の特権で作成されたデータベースユーザーを使用するというベストプラクティスに従ってください。

RDS for MySQL バージョン 8.0.36 以降には、以下のすべての権限を持つ特別なロールが含まれています。ロールの名前は rds_superuser_role です。各 DB インスタンスのプライマリ管理ユーザーには、このロールが既に付与されています。rds_superuser_role ロールには、すべてのデータベースオブジェクトに対する次の権限が含まれます。

- ALTER
- APPLICATION_PASSWORD_ADMIN
- ALTER ROUTINE
- CREATE
- CREATE ROLE
- CREATE ROUTINE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- DROP ROLE
- EVENT
- EXECUTE
- INDEX

- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

ロール定義には WITH GRANT OPTION が含まれるため、管理ユーザーはそのロールを他のユーザーに付与することができます。特に、管理者は MySQL クラスターをターゲットとするバイナリログレプリケーションの実行に必要な権限を付与する必要があります。

> **Tip**
>
> アクセス許可の詳細全体を表示するには、次のステートメントを使用します。
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> ```

RDS for MySQL バージョン 8.0.36 以降でロールを使用してアクセスを許可する場合は、SET ROLE *role_name* または SET ROLE ALL ステートメントを使用してロールも有効にします。以下の例のように指定します。CUSTOM_ROLE 用に適切なロール名を置き換えます。

```
# Grant role to user
mysql> GRANT CUSTOM_ROLE TO 'user'@'domain-or-ip-address'
```

```
# Check the current roles for your user. In this case, the CUSTOM_ROLE role has not
 been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-------------------------------------------------+
| CURRENT_ROLE()                                  |
+-------------------------------------------------+
| `CUSTOM_ROLE`@`%`,`rds_superuser_role`@`%` |
+-------------------------------------------------+
```

RDS for MySQL の動的権限

動的権限は、GRANT ステートメントを使用して明示的に付与できる MySQL 権限です。RDS for MySQL のバージョンに応じて、RDS では特定の動的権限のみを付与できます。RDS では、レプリケーションやバックアップなどの特定のデータベース操作を妨げる可能性があるため、これらの権限の一部は許可されません。

次の表は、さまざまな MySQL バージョンに対して付与できるこれらの権限を示しています。8.0.36 より前の MySQL バージョンからバージョン 8.0.36 以降にアップグレードする場合は、特定の権限の付与が許可されなくなった場合にアプリケーションコードを更新する必要がある場合があります。

特権	MySQL 8.0.35 以前	MySQL 8.0.36 以降のマイナーバージョン	MySQL 8.4.3 以降
ALLOW_NON EXISTENT_DEFINER	利用不可	利用不可	許可されていません

特権	MySQL 8.0.35 以前	MySQL 8.0.36 以降のマイナーバージョン	MySQL 8.4.3 以降
APPLICATION_PASSWORD_ADMIN	許可	許可	許可
AUDIT_ABORT_EXEMPT	許可	許可されていません	許可されていません
AUDIT_ADMIN	許可されていません	許可されていません	許可されていません
AUTHENTICATION_POLICY_ADMIN	許可	許可されていません	許可されていません
BACKUP_ADMIN	許可	許可されていません	許可されていません
BINLOG_ADMIN	許可	許可されていません	許可されていません
BINLOG_ENCRYPTION_ADMIN	許可されていません	許可されていません	許可されていません
CLONE_ADMIN	許可されていません	許可されていません	許可されていません
CONNECTION_ADMIN	許可	許可されていません	許可されていません
ENCRYPTION_KEY_ADMIN	許可されていません	許可されていません	許可されていません
FIREWALL_ADMIN	許可されていません	許可されていません	許可されていません
FIREWALL_EXEMPT	許可	許可されていません	許可されていません
FIREWALL_USER	許可されていません	許可されていません	許可されていません
FLUSH_OPTIMIZER_COSTS	許可	許可	許可

特権	MySQL 8.0.35 以前	MySQL 8.0.36 以降のマイナーバージョン	MySQL 8.4.3 以降
FLUSH_PRIVILEGES	利用不可	利用不可	許可
FLUSH_STATUS	許可	許可	許可
FLUSH_TABLES	許可	許可	許可
FLUSH_USER_RESOURCES	許可	許可	許可
GROUP_REPLICATION_ADMIN	許可されていません	許可されていません	許可されていません
GROUP_REPLICATION_STREAM	許可されていません	許可されていません	許可されていません
INNODB_REDO_LOG_ARCHIVE	許可されていません	許可されていません	許可されていません
INNODB_REDO_LOG_ENABLE	許可されていません	許可されていません	許可されていません
MASKING_DICTIONARIES_ADMIN	許可されていません	許可されていません	許可されていません
NDB_STORED_USER	許可されていません	許可されていません	許可されていません
OPTIMIZE_LOCAL_TABLE	利用不可	利用不可	許可されていません
PASSWORDLESS_USER_ADMIN	許可されていません	許可されていません	許可されていません
PERSIST_RO_VARIABLES_ADMIN	許可されていません	許可されていません	許可されていません

特権	MySQL 8.0.35 以前	MySQL 8.0.36 以降のマイナーバージョン	MySQL 8.4.3 以降
REPLICATION_APPLIER	許可	許可されていません	許可されていません
REPLICATION_SLAVE_ADMIN	許可されていません	許可されていません	許可されていません
RESOURCE_GROUP_ADMIN	許可	許可されていません	許可されていません
RESOURCE_GROUP_USER	許可	許可されていません	許可されていません
ROLE_ADMIN	許可	許可	許可
SENSITIVE_VARIABLES_OBSERVER	許可	許可	許可
SERVICE_CONNECTION_ADMIN	許可	許可されていません	許可されていません
SESSION_VARIABLES_ADMIN	許可	許可	許可
SET_ANY_DEFINER	利用不可	利用不可	許可
SET_USER_ID	許可	許可	利用不可
SHOW_ROUTINE	許可	許可	許可
SKIP_QUERY_REWRITE	許可されていません	許可されていません	許可されていません
SYSTEM_USER	許可されていません	許可されていません	許可されていません
SYSTEM_VARIABLES_ADMIN	許可されていません	許可されていません	許可されていません

特権	MySQL 8.0.35 以前	MySQL 8.0.36 以降のマイナーバージョン	MySQL 8.4.3 以降
TABLE_ENCRYPTION_ADMIN	許可されていません	許可されていません	許可されていません
TELEMETRY_LOG_ADMIN	許可	許可されていません	許可されていません
TP_CONNECTION_ADMIN	許可されていません	許可されていません	許可されていません
TRANSACTION_GTID_TAG	利用不可	利用不可	許可されていません
VERSION_TOKEN_ADMIN	許可されていません	許可されていません	許可されていません
XA_RECOVER_ADMIN	許可	許可	許可

RDS for MySQL のセッションまたはクエリの終了

DB インスタンス上のユーザーセッションまたはクエリは、rds_kill コマンドおよび rds_kill_query コマンドを使用して終了することができます。まず MySQL DB インスタンスに接続し、次に以下に示す適切なコマンドを発行します。詳細については、「MySQL DB インスタンスへの接続」を参照してください。

```
CALL mysql.rds_kill(thread-ID)
CALL mysql.rds_kill_query(thread-ID)
```

例えば、スレッド 99 で実行中のセッションを終了するには、次のように入力します。

```
CALL mysql.rds_kill(99);
```

スレッド 99 で実行中のクエリを終了するには、次のように入力します。

```
CALL mysql.rds_kill_query(99);
```

RDS for MySQL の現在のレプリケーションエラーのスキップ

リードレプリカが反応を停止する原因となるエラーがデータの整合性に影響しない場合、リードレプリカでそのエラーをスキップすることができます。

> **Note**
>
> まず、そのエラーを安全にスキップできることを確認します。MySQL ユーティリティで、リードレプリカに接続して以下の MySQL コマンドを実行します。
>
> ```
> SHOW REPLICA STATUS\G
> ```
>
> 返される値の詳細については、MySQL ドキュメントを参照してください。
> MySQL の旧バージョンは SHOW SLAVE STATUS ではなく SHOW REPLICA STATUS を使用していました。8.0.23 より前の MySQL バージョンを使用している場合は、SHOW SLAVE STATUS を使用します。

リードレプリカのエラーは、次の方法でスキップできます。

トピック

- mysql.rds_skip_repl_error の手順を呼び出します。
- slave_skip_errors パラメータの設定

mysql.rds_skip_repl_error の手順を呼び出します。

Amazon RDS では、リードレプリカのエラーをスキップするために呼び出すことができるストアドプロシージャを提供しています。まず、リードレプリカに接続してから以下のように適切なコマンドを発行します。詳細については、「MySQL DB インスタンスへの接続」を参照してください。

エラーをスキップするには、次のコマンドを実行します。

```
CALL mysql.rds_skip_repl_error;
```

このコマンドは、出典 DB インスタンス、またはレプリケーションエラーが発生していないリードレプリカでは実行しても効果はありません。

mysql.rds_skip_repl_error をサポートする MySQL のバージョンなどの詳細については、「mysql.rds_skip_repl_error」を参照してください。

> **Important**
>
> mysql.rds_skip_repl_errorを呼び出そうとして次のエラーが発生した場合、ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist、MySQL DB インスタンスを最新のマイナーバージョンまたはmysql.rds_skip_repl_errorにリストされている最小のマイナーバージョンの1つにアップグレードします。

slave_skip_errors パラメータの設定

1 つ以上のエラーをスキップするには、slave_skip_errorsリードレプリカに静的パラメータを設定します。このパラメータでは、1 つ以上の特定のレプリケーションエラーコードをスキップするように設定できます。現在、このパラメータは RDS for MySQL 5.7 DB インスタンスに対してのみ設定できます。パラメータの設定変更後に新しい設定を有効にするには、DB インスタンスを必ず再起動してください。これらのパラメータ設定の詳細については、MySQL のドキュメントを参照してください。

このパラメータは別の DB パラメータグループに設定することをお勧めします。この DB パラメータグループは、エラーをスキップする必要があるリードレプリカにのみ関連付けることができます。このベストプラクティスに従うことで、他の DB インスタンスやリードレプリカに与える潜在的な影響が軽減されます。

> **Important**
>
> このパラメータにデフォルト以外の値を設定すると、レプリケーションの不整合につながることがあります。問題を解決するための他のオプションを使い果たし、リードレプリカのデータに潜在的な影響が確実である場合にのみ、このパラメータをデフォルト以外の値に設定してください。

InnoDB テーブルスペースの操作による RDS for MySQL のクラッシュリカバリ時間の短縮

MySQL のすべてのテーブルは、テーブル定義、データ、およびインデックスから構成されます。MySQL のストレージエンジン InnoDB は、tablespace にテーブルのデータとインデックスを格納します。InnoDB は、データディクショナリおよびその他の関連メタデータを含むグローバル共有データスペースを作成し、テーブルのデータとインデックスを含めることができます。また、InnoDB は、テーブルおよびパーティションごとに個別のテーブルスペースを作成することもできます。これらの個別のテーブルスペースは、.ibd の拡張子を持つファイルに格納され、各テーブルスペースのヘッダーには、それを一意に識別する数値が含まれます。

Amazon RDS は `innodb_file_per_table` と呼ばれる MySQL パラメータグループにパラメータを提供します。このパラメータは、InnoDB が、新しいテーブルデータとインデックスを共有テーブルスペースに追加するか (パラメータ値を 0 に設定)、または個別のテーブルスペースに追加するか (パラメータ値を 1 に設定) を制御します。Amazon RDS は、`innodb_file_per_table` パラメータのデフォルト値を 1 に設定します。この設定により、個別に InnoDB テーブルを削除し、それらのテーブルで使用していたストレージを、DB インスタンス用として再要求することができます。ほとんどの場合、`innodb_file_per_table` パラメータは 1 に設定することをお勧めします。

多数のテーブルがある場合 (スタンダード (磁気) または汎用 SSD ストレージを使用するときは 1,000 以上のテーブル、プロビジョンド IOPS ストレージを使用するときは 10,000 以上のテーブル) は、`innodb_file_per_table` パラメータを 0 に設定する必要があります。このパラメータを 0 に設定すると、個別のテーブルスペースは作成されないため、データベースのクラッシュ回復時間を短縮できます。

MySQL は、クラッシュ回復サイクル中に個々のメタデータファイルを処理して、そこにテーブルスペースを格納します。MySQL が共有テーブルスペース内のメタデータ情報を処理するために必要な時間は、複数のテーブルスペースが存在するときに数千のテーブルスペースファイルを処理するために必要な時間と比べるとごく僅かです。テーブルスペース番号は、各ファイルのヘッダーに保存されているため、すべてのテーブルスペースファイルを読み込むための時間を集計すると、数時間かかる可能性があります。例えば、クラッシュ回復サイクル中に、スタンダードストレージの 100 万の InnoDB テーブルスペースを処理するには、5 ~ 8 時間かかる可能性があります。場合によっては、InnoDB がクラッシュ回復サイクル後に、追加クリーンアップが必要であると判断し、再度クラッシュ回復サイクルをスタートすることがあり、それによって回復時間が長くなります。クラッシュ回復サイクルには、テーブルスペース情報の処理以外に、トランザクションのロールバック、損傷したページの修復、およびその他の処理も必要であることに注意してください。

innodb_file_per_table パラメータはパラメータグループ内にあるため、DB インスタンスを再起動しなくても、DB インスタンスが使用するパラメータグループを編集することで、このパラメータ値を変更できます。例えば、設定が 1 (個別のテーブルを作成する) から 0 (共有テーブルスペースを使用する) に変更されると、その後、新しい InnoDB テーブルが共有テーブルスペースに追加されますが、既存のテーブルには個別のテーブルスペースがそのまま残ります。InnoDB テーブルを共有テーブルスペースに移動するには、ALTER TABLE コマンドを使用する必要があります。

複数のテーブルスペースを共有テーブルスペースに移行する

InnoDB テーブルのメタデータを固有のテーブルスペースから共有テーブルスペースに移動することができます。このコマンドは、innodb_file_per_table パラメータ設定に従って、テーブルメタデータを再構築します。まず MySQL DB インスタンスに接続し、次に以下に示す適切なコマンドを発行します。詳細については、「MySQL DB インスタンスへの接続」を参照してください。

```
ALTER TABLE table_name ENGINE = InnoDB, ALGORITHM=COPY;
```

例えば、次のクエリは共有テーブルスペースにない InnoDB テーブルごとに ALTER TABLE ステートメントを返します。

MySQL 5.7 DB インスタンスの場合:

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(LEFT(NAME , INSTR((NAME), '/') - 1), '`', '``'), '`.`',
REPLACE(SUBSTR(NAME FROM INSTR(NAME, '/') + 1), '`', '``'), '` ENGINE=InnoDB,
 ALGORITHM=COPY;') AS Query
FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES
WHERE SPACE <> 0 AND LEFT(NAME, INSTR((NAME), '/') - 1) NOT IN ('mysql','');
```

MySQL 8.4 および 8.0 DB インスタンスの場合:

```
SELECT CONCAT('ALTER TABLE `',
REPLACE(LEFT(NAME , INSTR((NAME), '/') - 1), '`', '``'), '`.`',
REPLACE(SUBSTR(NAME FROM INSTR(NAME, '/') + 1), '`', '``'), '` ENGINE=InnoDB,
 ALGORITHM=COPY;') AS Query
FROM INFORMATION_SCHEMA.INNODB_TABLES
WHERE SPACE <> 0 AND LEFT(NAME, INSTR((NAME), '/') - 1) NOT IN ('mysql','');
```

テーブルのメタデータを共有テーブルスペースに移動するために MySQL テーブルを再構築するには、一時的にテーブルの再構築のための追加ストレージ領域が必要になるため、DB インスタンスに

は使用可能なストレージ領域が必要です。再構築中は、テーブルがロックされ、クエリのアクセスが制限されます。小さなテーブルや、頻繁にアクセスされないテーブルの場合、これは問題にはならないことがあります。大きなテーブルや過酷な同時実行環境で頻繁にアクセスされるテーブルの場合は、リードレプリカでテーブルを再構築できます。

リードレプリカを作成し、リードレプリカの共有テーブルスペースにテーブルのメタデータを移行できます。ALTER TABLE ステートメントはリードレプリカのアクセスをブロックしますが、出典 DB インスタンスは影響を受けません。テーブルの再構築中はリードレプリカが停滞しますが、出典 DB インスタンスはバイナリログを生成し続けます。再構築には追加ストレージ領域が必要で、再生ログファイルは大きくなる可能性があるため、出典 DB インスタンスより大きなストレージが割り当てられたリードレプリカを作成する必要があります。

リードレプリカを作成し、共有テーブルスペースを使用する InnoDB テーブルを再構築するには、次のステップに従ってください。

1. バイナリログ作成が有効になるように、出典 DB インスタンスでバックアップ保持が有効になっていることを確認します。
2. AWS マネジメントコンソール または AWS CLI を使用して、出典 DB インスタンスのリードレプリカを作成します。リードレプリカの作成にはクラッシュ回復と同じプロセスが多く含まれているため、InnoDB テーブルスペースの数が多い場合は、作成プロセスに時間がかかることがあります。出典 DB インスタンスで現在使用しているよりも大きいストレージ領域をリードレプリカに割り当てます。
3. リードレプリカが作成されたら、パラメータを `read_only = 0` および `innodb_file_per_table = 0` に設定したパラメータグループを作成します。次に、パラメータグループをリードレプリカに関連付けます。
4. レプリカで移行するすべてのテーブルに対して、次の SQL ステートメントを発行します。

```
ALTER TABLE name ENGINE = InnoDB
```

5. リードレプリカですべての ALTER TABLE ステートメントが完了したら、リードレプリカが出典 DB インスタンスに接続され、2 つのインスタンスが同期していることを確認します。
6. コンソールまたは CLI を使用して、リードレプリカをインスタンスに昇格します。新しいスタンドアロン DB インスタンスに使用されるパラメータグループの `innodb_file_per_table` パラメータが 0 に設定されていることを確認します。新しいスタンドアロン DB インスタンスの名前を変更し、アプリケーションを新しいスタンドアロン DB インスタンスにします。

RDS for MySQL のグローバルステータス履歴の管理

> **Tip**
>
> データベースのパフォーマンスを分析するには、Amazon RDS のPerformance Insights を使用することもできます。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。

MySQL はオペレーションに関する情報を提供する多くのステータス可変を維持しています。その値は、DB インスタンスのロックまたはメモリ問題の検出に役立ちます。これらのステータス可変の値は、最後に DB インスタンスがスタートされてからの累積です。ほとんどのステータス可変は、`FLUSH STATUS` コマンドを使用して 0 にリセットできます。

これらの値を経時的にモニタリングできるように、Amazon RDS は、これらのステータス可変値のスナップショットを作成し、前回のスナップショット以降に行われた変更と共にテーブルに書き込む一連の手順を提供します。このインフラストラクチャは Global Status History (GoSH) と呼ばれ、5.5.23 で始まるバージョンのすべての MySQL DB インスタンスにインストールされています。GoSH は、デフォルトでは無効化されています。

GoSH を有効にするには、初期にパラメータ `event_scheduler` を `ON` に設定し、DB パラメータグループからイベントスケジューラーを有効にします。MySQL 5.7 を実行している MySQL DB インスタンスの場合、パラメータ `show_compatibility_56` を `1` にサーバーしてください。DB パラメータグループの作成と変更の詳細については、「Amazon RDS のパラメータグループ」を参照してください。このパラメータを有効にした場合の副作用の詳細については、「MySQL 5.7 リファレンスマニュアル」の「show_patibility_56」を参照してください。

次に、以下の表の手順を使用して、GoSH を有効化し、設定します。まず MySQL DB インスタンスに接続し、次に以下に示す適切なコマンドを発行します。詳細については、「MySQL DB インスタンスへの接続」を参照してください。プロシージャごとに、次のコマンドを実行し、*procedure-name*を置き換えます。

```
CALL procedure-name;
```

次の表に、前のコマンドで *procedure-name* に使用できるすべてのプロシージャを示します。

手順	説明
mysql.rds_enable_gsh_collector	GoSH のデフォルトのスナップショット作成を有効化します。間隔は rds_set_gsh_collector で指定します。
mysql.rds_set_gsh_collector	スナップショット作成の間隔を分単位で指定します。デフォルト値は 5 です。
mysql.rds_disable_gsh_collector	スナップショットを無効にします。
mysql.rds_collect_global_status_history	オンデマンドでスナップショットを作成します。
mysql.rds_enable_gsh_rotation	mysql.rds_global_status_history テーブルから mysql.rds_global_status_history_old へのコンテンツのローテーションを有効化します。間隔は rds_set_gsh_rotation で指定します。
mysql.rds_set_gsh_rotation	テーブルのローテーション間隔を日単位で指定します。デフォルト値は 7 です。
mysql.rds_disable_gsh_rotation	テーブルのローテーションを無効にします。
mysql.rds_rotate_global_status_history	mysql.rds_global_status_history テーブルのコンテンツを mysql.rds_global_status_history_old にオンデマンドでローテーションします。

GoSH の実行中は、書き込んでいるテーブルに対してクエリを実行できます。例えば、Innodb バッファープールのヒット率を照会するには、次のクエリを実行します。

```
select a.collection_end, a.collection_start, (( a.variable_Delta-b.variable_delta)/
a.variable_delta)*100 as "HitRatio"
    from mysql.rds_global_status_history as a join mysql.rds_global_status_history as b
 on a.collection_end = b.collection_end
```

```
    where a. variable_name = 'Innodb_buffer_pool_read_requests' and b.variable_name =
'Innodb_buffer_pool_reads'
```

MySQL 8.4 でのバッファプールサイズと REDO ログ容量の設定

MySQL 8.4 の場合、Amazon RDS ではデフォルトで innodb_dedicated_server パラメータが有効になります。innodb_dedicated_server パラメータを使用すると、データベースエンジンは innodb_buffer_pool_size および innodb_redo_log_capacity パラメータを計算します。これらのパラメータの計算方法については、MySQL ドキュメントの「Configuring InnoDB Buffer Pool Size」と「Redo Log」を参照してください。

innodb_dedicated_server を有効にすると、innodb_buffer_pool_size パラメータは DB インスタンスクラスのメモリに基づいて計算されます。次の表は、検出されたサーバーメモリと対応するバッファプールサイズを示しています。

検出されたサーバーメモリ	バッファプールサイズ
1 GB 未満	デフォルト値は 128 MB
1 GB ~ 4 GB	########### * 0.5
4 GB 超	########### * 0.75

innodb_redo_log_capacity パラメータは、インスタンスクラスに合わせて自動的に (vCPU の数 / 2) GB にスケールされ、最大 16 GB まで拡張されます。インスタンスクラスが大きいほど REDO ログ容量が大きくなり、書き込み負荷の高いワークロードのパフォーマンスと耐障害性が向上します。

MySQL 8.0 から MySQL 8.4 にアップグレードする前に、アップグレードの完了後に発生する可能性のある REDO ログのサイズの増加に対応するために、ストレージ容量を増やしてください。詳細については、「DB インスタンスストレージの容量を増加する」を参照してください。

innodb_dedicated_server パラメータで innodb_buffer_pool_size および innodb_redo_log_capacity パラメータの値を計算する必要がない場合は、カスタムパラメータグループで具体的な値を設定することで、これらの値を上書きできます。または、innodb_dedicated_server パラメータを無効にし、カスタムパラメータグループで innodb_buffer_pool_size および innodb_redo_log_capacity パラメータの値を設定する

こともできます。詳細については、「デフォルトおよびカスタムパラメータグループ」を参照してください。

innodb_dedicated_server パラメータを 0 に設定して無効にし、innodb_buffer_pool_size および innodb_redo_log_capacity パラメータの値を設定しないと、Amazon RDS は後者の 2 つのパラメータをそれぞれ 128 MB と 100 MB に設定します。大きなインスタンスクラスの場合、これらのデフォルトではパフォーマンスが低下します。

MySQL DB インスタンスのローカルタイムゾーン

デフォルトでは、MySQL DB インスタンスのタイムゾーンは協定世界時 (UTC) です。代わりに、DB インスタンスのタイムゾーンをアプリケーションのローカルタイムゾーンに設定できます。

DB インスタンスのローカルタイムゾーンを設定するには、DB インスタンスのパラメータグループの `time_zone` パラメータを、このセクションで後述するサポートされている値のいずれかに設定します。パラメータグループの `time_zone` パラメータを設定すると、そのパラメータグループを使用しているすべての DB インスタンスとリードレプリカは、新しいローカルタイムゾーンを使用するように変更されます。パラメータグループのパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。

ローカルタイムゾーンを設定した後、データベースへのすべての新しい接続にその変更が反映されます。ローカルタイムゾーンを変更するときにデータベースへの接続を開いている場合、その接続を閉じて新しい接続を開くまで、ローカルタイムゾーンは更新されません。

DB インスタンスとそのリードレプリカには異なるローカルタイムゾーンを設定できます。そのためには、DB インスタンスとレプリカに異なるパラメータグループを使用し、各パラメータグループの `time_zone` パラメータを異なるローカルタイムゾーンに設定します。

AWS リージョン 間のレプリケーションを実行する場合は、ソース DB インスタンスとリードレプリカに異なるパラメータグループ (パラメータグループは AWS リージョン に固有のもの) を使用します。各インスタンスに同じローカルタイムゾーンを使用するには、インスタンスとリードレプリカのパラメータグループの `time_zone` パラメータを設定する必要があります。

DB スナップショットから DB インスタンスを復元すると、ローカルタイムゾーンが UTC に設定されます。復元が完了したら、タイムゾーンをローカルタイムゾーンに更新できます。DB インスタンスをある時点まで復元する場合、復元された DB インスタンスのローカルタイムゾーンは、復元された DB インスタンスのパラメータグループに設定されているタイムゾーンです。

Internet Assigned Numbers Authority (IANA) は年に数回、https://www.iana.org/time-zones で新しいタイムゾーンを公開します。RDS が MySQL の新しいマイナーメンテナンスリリースをリリースするたびに、リリース時の最新のタイムゾーンデータが同梱されます。MySQL の最新バージョンの RDS を使用すると、RDS からの最新のタイムゾーンデータが得られます。DB インスタンスに最新のタイムゾーンデータがあることを確認するには、DB エンジンの上位バージョンにアップグレードすることをお勧めします。または、MariaDB DB インスタンスのタイムゾーンテーブルを手動で変更することもできます。そのためには、SQL コマンドを使用するか、SQL クライアントで mysql_tzinfo_to_sql ツールを実行します。タイムゾーンデータを手動で更新して、DB インスタンスを再起動し、変更を有効にします。RDS は、実行中の DB インスタンスのタイムゾーンデータを変

更またはリセットしません。新しいタイムゾーンデータは、データベースエンジンのバージョンアップグレードを実行する場合にのみインストールされます。

ローカルタイムゾーンは以下のいずれかの値に設定できます。

ゾーン	タイムゾーン
アフリカ	Africa/Cairo、Africa/Casablanca、Africa/Harare、Africa/Monrovia、Africa/Nairobi、Africa/Tripoli、Africa/Windhoek
南北アメリカ大陸	America/Araguaina、America/Asuncion、America/Bogota、America/Buenos_Aires、America/Caracas、America/Chihuahua、America/Cuiaba、America/Denver、America/Fortaleza、America/Guatemala、America/Halifax、America/Manaus、America/Matamoros、America/Monterrey、America/Montevideo、America/Phoenix、America/Santiago、America/Tijuana
アジア	Asia/Amman、Asia/Ashgabat、Asia/Baghdad、Asia/Baku、Asia/Bangkok、Asia/Beirut、Asia/Calcutta、Asia/Damascus、Asia/Dhaka、Asia/Irkutsk、Asia/Jerusalem、Asia/Kabul、Asia/Karachi、Asia/Kathmandu、Asia/Krasnoyarsk、Asia/Magadan、Asia/Muscat、Asia/Novosibirsk、Asia/Riyadh、Asia/Seoul、Asia/Shanghai、Asia/Singapore、Asia/Taipei、Asia/Tehran、Asia/Tokyo、Asia/Ulaanbaatar、Asia/Vladivostok、Asia/Yakutsk、Asia/Yerevan
大西洋	Atlantic/Azores
オーストラリア	Australia/Adelaide、Australia/Brisbane、Australia/Darwin、Australia/Hobart、Australia/Perth、Australia/Sydney
ブラジル	Brazil/DeNoronha、Brazil/East
カナダ	Canada/Newfoundland、Canada/Saskatchewan、Canda/Yukon
欧州	Europe/Amsterdam、Europe/Athens、Europe/Dublin、Europe/Helsinki、Europe/Istanbul、Europe/Kaliningrad Europe/Moscow、Europe/Paris、Europe/Prague、Europe/Sarajevo
太平洋	Pacific/Auckland、Pacific/Fiji、Pacific/Guam、Pacific/Honolulu、Pacific/Samoa

ゾーン	タイムゾーン
米国	US/Alaska、US/Central、US/East-Indiana、US/Eastern、US/Pacific
UTC	UTC

Amazon RDS for MySQL の既知の問題と制限

Amazon RDS for MySQL を使用する際の既知の問題と制限は、以下のとおりです。

トピック

- InnoDB 予約語
- Amazon RDS for MySQL のストレージ全体の動作
- InnoDB バッファプールサイズの不整合
- インデックスマージの最適化で誤った結果が返される
- Amazon RDS DB インスタンスに使用する MySQL のパラメータの例外
- Amazon RDS での MySQL のファイルサイズ制限
- MySQL キーリングプラグインがサポートされていない
- カスタムポート
- MySQL ストアドプロシージャの制限事項
- 外部ソースインスタンスを使用した GTID ベースのレプリケーション
- MySQL デフォルト認証プラグイン
- innodb_buffer_pool_size の上書き
- MySQL 5.7 から MySQL 8.4 へのアップグレード
- InnoDB ページ圧縮

InnoDB 予約語

`InnoDB` は、RDS for MySQL の予約語です。MySQL データベースにこの名前を使用することはできません。

Amazon RDS for MySQL のストレージ全体の動作

MySQL DB インスタンスのストレージがいっぱいになると、メタデータの不一致、ディクショナリの不一致、孤立テーブルが発生する可能性があります。これらの問題を回避するために、Amazon RDS は `storage-full` 状態に達する DB インスタンスを自動的に停止します。

MySQL DB インスタンスは、次の場合に `storage-full` 状態に達します。

- DB インスタンスのストレージは 20,000 MiB 未満で、使用可能なストレージは 200 MiB 以下です。

- DB インスタンスのストレージは 102,400 MiB 超で、使用可能なストレージは 1024 MiB 以下です。
- DB インスタンスのストレージは 20,000 MiB ~ 102,400 MiB で、使用可能なストレージの 1% 未満です。

DB インスタンスが `storage-full` 状態に達したために Amazon RDS がそのインスタンスを自動的に停止した後でも、そのインスタンスを変更できます。DB インスタンスを再起動するには、次のうち少なくとも 1 つを実行します。

- DB インスタンスを変更して、ストレージのオートスケーリングを有効にします。

  ストレージのオートスケーリングの詳細については、Amazon RDS ストレージの自動スケーリングによる容量の自動管理 を参照してください。
- ストレージ容量を増やすには、DB インスタンスを変更します。

  ストレージ容量の引き上げの詳細については、DB インスタンスストレージの容量を増加する を参照してください。

これらの変更のいずれかを行うと、DB インスタンスは自動的に再起動します。DB インスタンスの変更については、Amazon RDS DB インスタンスを変更するを参照してください。

InnoDB バッファプールサイズの不整合

現在のところ、MySQL 5.7 には、InnoDB バッファプールサイズの管理方法にバグがあります。MySQL 5.7 が `innodb_buffer_pool_size` パラメータの値を大きな値に調整し、それにより InnoDB バッファプールが大きくなりすぎ、過度のメモリを使用する可能性があります。この影響により、MySQL データベースエンジンは実行を停止するか、起動できなくなる場合があります。この問題は、使用できるメモリが少ない DB インスタンスクラスでより多く発生します。

この問題を解決するには、`innodb_buffer_pool_size` パラメータの値を、`innodb_buffer_pool_instances` パラメータの値と `innodb_buffer_pool_chunk_size` パラメータの値の積の倍数に設定します。例えば、次の例に示すように、`innodb_buffer_pool_size` パラメータの値を `innodb_buffer_pool_instances` および `innodb_buffer_pool_chunk_size` パラメータの積の 8 倍に設定します。

```
innodb_buffer_pool_chunk_size = 536870912
innodb_buffer_pool_instances = 4
innodb_buffer_pool_size = (536870912 * 4) * 8 = 17179869184
```

この MySQL 5.7 のバグの詳細については、MySQL ドキュメントの https://bugs.mysql.com/bug.php?id=79379 を参照してください。

インデックスマージの最適化で誤った結果が返される

インデックスマージの最適化を使用するクエリは、MySQL 5.5.37 で導入された MySQL クエリオプティマイザのバグのために誤った結果を返す場合があります。複数のインデックスを持つテーブルに対してクエリを実行すると、オプティマイザは複数のインデックスに基づいて行の範囲をスキャンしますが、結果を正しくマージしません。クエリのクエリオプティマイザのバグの詳細については、MySQL バグデータベースの http://bugs.mysql.com/bug.php?id=72745 と http://bugs.mysql.com/bug.php?id=68194 を参照してください。

例えば、2 つのインデックスを持つテーブルで、検索引数がインデックス付き列を参照するクエリがあるとします。

```
SELECT * FROM table1
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

この場合、検索エンジンは両方のインデックスを検索します。ただし、バグが原因で、マージされた結果は正しくありません。

この問題を解決するには、次のいずれかを実行します。

- MySQL DB インスタンスの DB パラメータグループで optimizer_switch パラメータを index_merge=off に設定します。DB パラメータグループのパラメータの設定については、「Amazon RDS のパラメータグループ」を参照してください。
- MySQL DB インスタンスを MySQL バージョン 5.7 または 8.0 にアップグレードします。詳細については、「RDS for MySQL DB エンジンのアップグレード」を参照してください。
- インスタンスをアップグレードできない場合や、optimizer_switch パラメータを変更できない場合は、明示的にクエリのインデックスを指定してバグを回避できます。例えば、次のように指定します。

  ```
  SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
  ```

詳細については、MySQL ドキュメントの「インデックスマージの最適化」を参照してください。

Amazon RDS DB インスタンスに使用する MySQL のパラメータの例外

MySQL の一部のパラメータは、Amazon RDS DB インスタンスとともに使用する場合に、特別な考慮事項があります。

lower_case_table_names

Amazon RDS は、大文字と小文字を区別するファイルシステムを使用するため、`lower_case_table_names` サーバーパラメータの値を 2 (名前は指定されたとおりに保存されるが、小文字で比較される) に設定することはできません。以下は、Amazon RDS for MySQL インスタンスでサポートされている値です。

- 0 (名前は指定されたとおりに保存され、比較では大文字と小文字が区別される) は、すべての RDS for MySQL バージョンでサポートされています。
- 1 (名前は小文字で保存され、比較では大文字と小文字が区別されない) は、RDS for MySQL バージョン 5.7、バージョン 8.0.28 以降の 8.0 バージョン、およびバージョン 8.4 でサポートされています。

DB インスタンスを作成する前に、カスタム DB パラメータグループに `lower_case_table_names` パラメータを設定します。その後、DB インスタンスを作成する際にカスタム DB パラメータグループを指定します。

パラメータグループが 8.0 より低いバージョンの MySQL DB インスタンスに関連付けられている場合、`lower_case_table_names` パラメータを変更しないことをお勧めします。これを変更すると、ポイントインタイムリカバリのバックアップとリードレプリカの DB インスタンスで不整合が生じることがあります。

パラメータグループがバージョン 8.0 または 8.4 の MySQL DB インスタンスに関連付けられている場合、`lower_case_table_names` パラメータを変更できません。

リードレプリカでは、常にマスター DB インスタンスと同じ `lower_case_table_names` パラメータ値を使用する必要があります。

long_query_time

`long_query_time` パラメータを浮動小数点値に設定することで、スロークエリを MySQL スロークエリログにマイクロ秒の精度で記録できます。例えば、この値として 0.1 秒 (100 ミリ秒) を設定すると、1 秒未満のスロートランザクションのデバッグ時に役立ちます。

Amazon RDS での MySQL のファイルサイズ制限

MySQL バージョン 8.0 以降の DB インスタンスの場合、最大ファイルサイズは 16 TiB です。file-per-table テーブルスペースを使用する場合、最大ファイルサイズにより、InnoDB テーブルのサイズは 16 TiB に制限されます。テーブルがそれぞれに固有のテーブル領域に存在する InnoDB file-per-table テーブル領域は、MySQL DB インスタンスに対してデフォルトで設定されます。詳細については、MySQL ドキュメントの「InnoDB Limits」を参照してください。

> **Note**
>
> 既存の DB インスタンスには制限値がより低い場合があります。例えば、2014 年 4 月より前に作成された MySQL DB インスタンスの場合、テーブルサイズの上限は 2 TB です。この 2 TB というファイルサイズの制限は、2014 年 4 月より前に作成された DB スナップショットから作成された DB インスタンスまたはリードレプリカにも適用されます。その DB インスタンスがいつ作成されたかには関係ありません。

InnoDB file-per-table テーブルスペースの使用は、アプリケーションにより長所と短所があります。アプリケーションに最適な方法を判断するには、MySQL ドキュメントの「File-Per-Table テーブルスペース」を参照してください。

テーブルを最大ファイルサイズまで拡張可能にすることはお勧めしません。一般的に望ましいのは、データを小さなテーブルにパーティショニングすることです。これにより、パフォーマンスと復旧時間が改善される可能性があります。

大きなテーブルを小さなテーブルに分ける方法の 1 つはパーティション化です。パーティション化は、指定したルールに基づいて、大きなテーブルの各部分を個別のファイルに分散します。例えば、トランザクションを日付ごとに保存する場合、パーティション化を使用して古いトランザクションを別々のファイルに分散させるパーティションルールを作成できます。また、定期的に、アプリケーションですぐに使用する必要のない履歴トランザクションデータをアーカイブできます。詳細については、MySQL ドキュメントの「Partitioning」を参照してください。

すべてのテーブルと InnoDB システムテーブルスペースのサイズを提供する単一のシステムテーブルまたはビューはないため、テーブルスペースのサイズを決定するには複数のテーブルをクエリする必要があります。

InnoDB システムテーブルスペースとデータディクショナリテーブルスペースのサイズを確認するには

- 次の SQL コマンドを使用して、パーティション化の対象になる過度に大きなテーブルスペースがあるかどうかを確認します。

> **Note**
>
> データディクショナリテーブルスペースは MySQL 8.0 以降のバージョンに固有です。

```
select FILE_NAME,TABLESPACE_NAME, ROUND(((TOTAL_EXTENTS*EXTENT_SIZE)
/1024/1024/1024), 2)  as "File Size (GB)" from information_schema.FILES
where tablespace_name in ('mysql','innodb_system');
```

InnoDB システムテーブルスペース外の InnoDB ユーザーテーブルのサイズを確認するには (MySQL 5.7 バージョンの場合)

- 次の SQL コマンドを使用して、パーティション化の対象になる過度に大きなテーブルがあるか判断します。

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_SYS_TABLESPACES ORDER BY 3 DESC;
```

InnoDB システムテーブルスペース外の InnoDB ユーザーテーブルのサイズを確認するには (MySQL 8.0 以降のバージョンの場合)

- 次の SQL コマンドを使用して、パーティション化の対象になる過度に大きなテーブルがあるか判断します。

```
SELECT SPACE,NAME,ROUND((ALLOCATED_SIZE/1024/1024/1024), 2)
as "Tablespace Size (GB)"
FROM information_schema.INNODB_TABLESPACES ORDER BY 3 DESC;
```

InnoDB 以外のユーザーテーブルのサイズを確認するには

- 次の SQL コマンドを使用して、InnoDB 以外のユーザーテーブルが過度に大きいかどうかを確認します。

```
SELECT TABLE_SCHEMA, TABLE_NAME, round(((DATA_LENGTH + INDEX_LENGTH+DATA_FREE)
/ 1024 / 1024/ 1024), 2) As "Approximate size (GB)" FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema')
and ENGINE<>'InnoDB';
```

InnoDB file-per-table テーブルスペースを有効にするには

- DB インスタンスのパラメータグループの innodb_file_per_table パラメータを 1 に設定します。

InnoDB file-per-table テーブルスペースを無効にするには

- DB インスタンスのパラメータグループの innodb_file_per_table パラメータを 0 に設定します。

パラメータグループの更新については、「Amazon RDS のパラメータグループ」を参照してください。

InnoDB file-per-table テーブルスペースを有効または無効にすると、次の例のように ALTER TABLE コマンドを実行してテーブルをグローバルテーブルスペースから固有のテーブルスペースに、または固有のテーブルスペースからグローバルテーブルスペースに移動できます。

```
ALTER TABLE table_name TABLESPACE=innodb_file_per_table;
```

MySQL キーリングプラグインがサポートされていない

現在、Amazon RDS for MySQL では、MySQL keyring_aws Amazon Web Services のキーリングプラグインをサポートしていません。

カスタムポート

Amazon RDS は MySQL エンジンのカスタムポート 33060 への接続をブロックします。MySQL エンジン用に別のポートを選択してください。

MySQL ストアドプロシージャの制限事項

mysql.rds_kill および mysql.rds_kill_query ストアドプロシージャでは、以下の RDS for MySQL バージョンで、ユーザー名が 16 文字を超える MySQL ユーザーが所有するセッションやクエリは終了できません。

- 8.0.32 以前の 8 バージョン
- 5.7.41 以前の 5.7 バージョン

外部ソースインスタンスを使用した GTID ベースのレプリケーション

Amazon RDS は、設定時に GTID_PURGED の設定を必要とする外部 MySQL インスタンスから Amazon RDS for MySQL DB インスタンスへのグローバルなトランザクション識別子 (GTID) に基づくレプリケーションをサポートしています。ただし、この機能をサポートしているのは RDS for MySQL 8.0.37 以降のバージョンのみです。

MySQL デフォルト認証プラグイン

RDS for MySQL バージョン 8.0.34 以降の 8.0 バージョンでは、`mysql_native_password` プラグインを使用します。`default_authentication_plugin` 設定は変更できません。

RDS for MySQL バージョン 8.4 以降のバージョンでは、`caching_sha2_password` プラグインをデフォルトの認証プラグインとして使用します。MySQL 8.4 のデフォルトの認証プラグインを変更できます。`mysql_native_password` プラグインは MySQL 8.4 でも動作しますが、このプラグインのサポートは MySQL 8.4 で終了します。デフォルトの認証プラグインを変更するには、カスタムパラメータグループを作成し、`authentication_policy` パラメータの値を変更します。詳細については、「the section called “デフォルトおよびカスタムパラメータグループ”」を参照してください。

innodb_buffer_pool_size の上書き

マイクロ DB インスタンスクラスまたはスモール DB インスタンスクラスでは、`innodb_buffer_pool_size` パラメータのデフォルト値は、次のコマンドを実行して返される値とは異なる場合があります。

```
mysql> SELECT @@innodb_buffer_pool_size;
```

この違いは、DB インスタンスクラスの管理の一部として Amazon RDS がデフォルト値を上書きする必要がある場合に発生する可能性があります。必要に応じて、デフォルト値を上書きし、DB インスタンスクラスがサポートする値に設定できます。有効な値は、DB インスタンスで使用可能なメモリ使用量と合計メモリを足して決定します。詳細については、「Amazon RDS インスタンスタイプ」を参照してください。

DB インスタンスのメモリが 4 GB しかない場合、`innodb_buffer_pool_size` を 8 GB に設定することはできませんが、他のパラメータに割り当てたメモリの量によっては、3 GB に設定できる場合があります。

入力した値が大きすぎる場合、Amazon RDS は次の制限に従って値を引き下げます。

- Micro DB インスタンスクラス: 256 MB
- db.t4g.micro DB インスタンスクラス: 128 MB

MySQL 5.7 から MySQL 8.4 へのアップグレード

MySQL 5.7 から MySQL 8.4 に直接アップグレードすることはできません。最初に MySQL 5.7 から MySQL 8.0 にアップグレードし、その後で MySQL 8.0 から MySQL 8.4 にアップグレードする必要があります。詳細については、「RDS for MySQL のメジャーバージョンのアップグレード」を参照してください。

InnoDB ページ圧縮

InnoDB ページ圧縮は、ファイルシステムのブロックサイズが 16k の Amazon RDS DB インスタンスでは機能しません。これは、ファイルシステムのブロックサイズが InnoDB ページサイズより小さくなければならないためです。2024 年 2 月以降、新しく作成されるすべての DB インスタンスのファイルシステムで、ブロックサイズは 16k になります。これにより、スループットが向上し、ページフラッシュ中の IOPS 消費量が減少します。

RDS for MySQL ストアドプロシージャリファレンス

これらのトピックでは、MySQL DB エンジンを実行する Amazon RDS インスタンスで使用できるシステムストアドプロシージャについて説明します。マスターユーザーがこれらの手順を実行する必要があります。

トピック

- グローバルステータス履歴の収集と維持
- バイナリログレプリケーションの設定、開始、停止
- セッションやクエリの終了
- アクティブ/アクティブクラスターの管理
- マルチソースレプリケーションの管理
- GTID を使用したトランザクションのレプリケーション
- クエリログのローテーション
- バイナリログ構成の設定と表示
- InnoDB キャッシュのウォームアップ

グローバルステータス履歴の収集と維持

Amazon RDS は、ステータス変数の値のスナップショットを掲示的に取得し、前回のスナップショット後の変化とともにテーブルに書き込む一連のプロシージャを提供します。このインフラストラクチャは Global Status History (GoSH) と呼ばれます。詳細については、「RDS for MySQL のグローバルステータス履歴の管理」(Global Status History の管理) を参照してください。

以下のストアドプロシージャは、Global Status History (GoSH) の収集方法と保守方法を管理します。

トピック

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Global Status History (GoSH) のスナップショットをオンデマンドで作成します。

構文

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Global Status History (GoSH) によって作成されたスナップショットを無効にします。

構文

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

mysql.global_status_history テーブルのローテーションをオフにします。

構文

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Global Status History (GoSH) をオンにして、rds_set_gsh_collector で指定された間隔でデフォルトのスナップショットを作成します。

構文

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

rds_set_gsh_rotation で指定された間隔での mysql.global_status_history テーブルのコンテンツの mysql.global_status_history_old へのローテーションをオンにします。

構文

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

mysql.global_status_history テーブルのコンテンツを mysql.global_status_history_old にオンデマンドでローテーションします。

構文

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Global Status History (GoSH) によって作成されたスナップショットの間隔を分単位で指定します。

構文

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

パラメータ

intervalPeriod

スナップショット作成の間隔 (分単位)。デフォルト値は 5 です。

mysql.rds_set_gsh_rotation

mysql.global_status_history テーブルのローテーション間隔を日単位で指定します。

構文

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

パラメータ

intervalPeriod

テーブルのローテーション間隔 (日単位)。デフォルト値は 7 です。

バイナリログレプリケーションの設定、開始、停止

以下のストアドプロシージャは、トランザクションが外部データベースから RDS for MySQL、または RDS for MySQL から外部データベースに複製される方法を制御します。

caching_sha2_password で設定されたレプリケーションユーザーによるレプリケーションを、このようなストアドプロシージャを使用して管理する場合は、SOURCE_SSL=1 を指定して TLS を設定する必要があります。caching_sha2_password は RDS for MySQL 8.4 のデフォルトの認証プラグインです。詳細については、「SSL を使用した暗号化」を参照してください。

リードレプリカの設定、使用、管理については、「the section called “MySQL リードレプリカ”」を参照してください。

トピック

- mysql.rds_next_master_log (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_next_source_log (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_external_master_with_auto_position (RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source_with_auto_position (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_external_source_gtid_purged
- mysql.rds_set_master_auto_position (RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_source_auto_position (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_skip_repl_error
- mysql.rds_start_replication

- mysql.rds_start_replication_until
- mysql.rds_stop_replication

mysql.rds_next_master_log (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)

出典データベースインスタンスのログの位置を、出典データベースインスタンスの次のバイナリログの先頭に変更します。このプロシージャは、リードレプリカでレプリケーション I/O エラー 1236 が発生した場合にのみ使用してください。

構文

```
CALL mysql.rds_next_master_log(
curr_master_log
);
```

パラメータ

curr_master_log

現在のマスターログファイルのインデックス。例えば、現在のファイルが mysql-bin-changelog.012345 という名前の場合は、インデックスは 12345 になります。現在のマスターログファイルの名前を調べるには、SHOW REPLICA STATUS コマンドを実行し、Master_Log_File フィールドを確認します。

使用に関する注意事項

マスターユーザーが mysql.rds_next_master_log を実行する必要があります。

> **Warning**
>
> mysql.rds_next_master_log は、レプリケーション出典であるマルチ AZ DB インスタンスのフェイルオーバーの後でレプリケーションが失敗し、Last_IO_Errno の SHOW REPLICA STATUS フィールドが I/O エラー 1236 を示している場合にのみ呼び出してください。
> mysql.rds_next_master_log を呼び出すと、フェイルオーバーイベントが発生する前にソースインスタンスのトランザクションがディスク上のバイナリログに書き込まれなかっ

た場合、リードレプリカのデータが失われる可能性があります。ソースインスタンスのパラメータ sync_binlog および innodb_support_xa を 1 に設定することで、このような問題が発生する可能性を軽減できますが、パフォーマンスが低下する恐れがあります。詳細については、 を参照してください。MySQL リードレプリカに関する問題のトラブルシューティング

例

RDS for MySQL リードレプリカでレプリケーションが失敗すると仮定します。リードレプリカで SHOW REPLICA STATUS\G を実行すると、次の結果が返されます。

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
```

```
               Source_SSL_Cert:
             Source_SSL_Cipher:
                Source_SSL_Key:
         Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                 Last_IO_Errno: 1236
                 Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
                Last_SQL_Errno: 0
                Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Last_IO_Errno フィールドはインスタンスが I/O エラー 1236 を受け取ったことを示します。Master_Log_File フィールドは、ファイル名が mysql-bin-changelog.012345 であることを示しています。これは、ログファイルのインデックスが 12345 であることを表しています。エラーを解決するには、次のパラメータを使用して mysql.rds_next_master_log を呼び出すことができます。

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log (RDS for MySQL メジャーバージョン 8.4 以降)

出典データベースインスタンスのログの位置を、出典データベースインスタンスの次のバイナリログの先頭に変更します。このプロシージャは、リードレプリカでレプリケーション I/O エラー 1236 が発生した場合にのみ使用してください。

構文

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

パラメータ

curr_source_log

現在のソースログファイルのインデックス。例えば、現在のファイルが mysql-bin-changelog.012345 という名前の場合は、インデックスは 12345 になります。現在のソースログファイルの名前を調べるには、SHOW REPLICA STATUS コマンドを実行し、Source_Log_File フィールドを確認します。

使用に関する注意事項

管理ユーザーは、mysql.rds_next_source_logプロシージャを実行すべきです。

> **Warning**
>
> mysql.rds_next_source_log は、レプリケーション出典であるマルチ AZ DB インスタンスのフェイルオーバーの後でレプリケーションが失敗し、Last_IO_Errno の SHOW REPLICA STATUS フィールドが I/O エラー 1236 を示している場合にのみ呼び出してください。
>
> mysql.rds_next_source_log を呼び出すと、フェイルオーバーイベントが発生する前にソースインスタンスのトランザクションがディスク上のバイナリログに書き込まれなかった場合、リードレプリカのデータが失われる可能性があります。ソースインスタンスのパラメータ sync_binlog および innodb_support_xa を 1 に設定することで、このような問題が発生する可能性を軽減できますが、パフォーマンスが低下する恐れがあります。詳細については、 を参照してください。MySQL リードレプリカに関する問題のトラブルシューティング

例

RDS for MySQL リードレプリカでレプリケーションが失敗すると仮定します。リードレプリカで SHOW REPLICA STATUS\G を実行すると、次の結果が返されます。

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
```

```
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Last_IO_Errno フィールドはインスタンスが I/O エラー 1236 を受け取ったことを示します。Source_Log_File フィールドは、ファイル名が mysql-bin-changelog.012345 であることを示しています。これは、ログファイルのインデックスが 12345 であることを表しています。エ

ラーを解決するには、次のパラメータを使用して mysql.rds_next_source_log を呼び出すことができます。

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で実行している MySQL インスタンスのリードレプリカとして使用しないように再設定します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_reset_external_master;
```

使用に関する注意事項

マスターユーザーが mysql.rds_reset_external_master を実行する必要があります。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとしての設定が解除される MySQL DB インスタンス上で実行する必要があります。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されていま

す。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

レプリケーションを使用して、Amazon RDS の外部で動作する MySQL インスタンスからデータをインポートする方法の詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で実行している MySQL インスタンスのリードレプリカとして使用しないように再設定します。

Important

この手順を実行するには、`autocommit` を有効にする必要があります。これを有効にするには、`autocommit` パラメータを `1` に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_reset_external_source;
```

使用に関する注意事項

管理ユーザーは、`mysql.rds_reset_external_source`プロシージャを実行すべきです。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとしての設定が解除される MySQL DB インスタンス上で実行する必要があります。

Note

可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されています。

Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。レプリケーションを使用して、Amazon RDS の外部で動作する MySQL インスタンスからデータをインポートする方法の詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で実行している MySQL インスタンスのリードレプリカとして使用するように設定します。

Important

この手順を実行するには、`autocommit` を有効にする必要があります。これを有効にするには、`autocommit` パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

Note

mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) ストアドプロシージャを使用して、外部出典データベースインスタンスと遅延レプリケーションを設定できます。

構文

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
```

```
);
```

パラメータ

host_name

出典データベースインスタンスとなる、Amazon RDS の外部で動作する MySQL インスタンスのホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの `REPLICATION CLIENT` および `REPLICATION SLAVE` のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

`replication_user_name` で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

レプリケーション情報を含む出典データベースインスタンス上のバイナリログの名前。

mysql_binary_log_file_location

`mysql_binary_log_file_name` バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、`SHOW MASTER STATUS`出典データベースインスタンス上で実行することによって決定できます。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

使用に関する注意事項

マスターユーザーが mysql.rds_set_external_master を実行する必要があります。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定される MySQL DB インスタンス上で実行する必要があります。

mysql.rds_set_external_master を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必ず設定してください。Amazon RDS の外部で動作する MySQL インスタンスに接続するには、MySQL の外部インスタンスの replication_user_name および replication_user_password アクセス権限を持つレプリケーションユーザーを示す REPLICATION CLIENT および REPLICATION SLAVE の値を指定する必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。以下に例を示します。

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

   > **Note**
   >
   > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の権限をレプリケーションユーザーに付与します。次の例では、ドメインの「repl_user」ユーザーに、すべてのデータベースの REPLICATION CLIENT および REPLICATION SLAVE 権限を付与します。

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

暗号化されたレプリケーションを使用するには、SSL 接続を使用するようにソースデータベースインスタンスを設定します。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されています。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_master を呼び出して Amazon RDS DB インスタンスをリードレプリカとして設定した後で、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスをスタートできます。mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_master が呼び出されると、Amazon RDS では、時刻、ユーザー、set master アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

例

MySQL DB インスタンス上で実行すると、DB インスタンスが Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定されます。次に例を示します。

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で実行している MySQL インスタンスのリードレプリカとして使用するように設定します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
```

```
  , ssl_encryption
);
```

パラメータ

host_name

出典データベースインスタンスとなる、Amazon RDS の外部で動作する MySQL インスタンスのホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの `REPLICATION CLIENT` および `REPLICATION SLAVE` のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

`replication_user_name` で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

レプリケーション情報を含む出典データベースインスタンス上のバイナリログの名前。

mysql_binary_log_file_location

`mysql_binary_log_file_name` バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、`SHOW MASTER STATUS`出典データベースインスタンス上で実行することによって決定できます。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

使用に関する注意事項

管理ユーザーは、mysql.rds_set_external_sourceプロシージャを実行すべきです。このプロシージャは、Amazon RDS の外部で実行している MySQL インスタンスのリードレプリカとして設定される RDS for MySQL DB インスタンス上で実行する必要があります。

mysql.rds_set_external_source を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必ず設定してください。Amazon RDS の外部で動作する MySQL インスタンスに接続するには、MySQL の外部インスタンスの replication_user_name および replication_user_password アクセス権限を持つレプリケーションユーザーを示す REPLICATION CLIENT および REPLICATION SLAVE の値を指定する必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。以下に例を示します。

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の権限をレプリケーションユーザーに付与します。次の例では、ドメインの 「repl_user」ユーザーに、すべてのデータベースの REPLICATION CLIENT および REPLICATION SLAVE 特権を付与します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

暗号化されたレプリケーションを使用するには、SSL 接続を使用するように出典データベースインスタンスを設定します。また、mysql.rds_import_binlog_ssl_material プロシージャを使用して、DB インスタンスまたは DB クラスターに認証局証明書、クライアント証明書、およびクライアントキーをインポートします。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されています。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

`mysql.rds_set_external_source` を呼び出して RDS for MySQL DB インスタンスをリードレプリカとして設定したら、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスを開始できます。mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降) を呼び出して、リードレプリカの設定を削除することもできます。

`mysql.rds_set_external_source` が呼び出されると、Amazon RDS では、時刻、ユーザー、`set master` アクションが `mysql.rds_history` テーブルと `mysql.rds_replication_status` テーブルに記録されます。

例

RDS for MySQL DB インスタンスで実行すると、次の例に示されているように、DB インスタンスが Amazon RDS の外部で実行されている MySQL インスタンスのリードレプリカとして設定されます。

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
```

```
  1);
```

mysql.rds_set_external_master_with_auto_position (RDS for MySQL メジャーバージョン 8.0 以前)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定します。このプロシージャは、遅延レプリケーション、および、グローバルトランザクション識別子 (GTID) ベースのレプリケーションも設定します。

> **Important**
>
> この手順を実行するには、`autocommit` を有効にする必要があります。これを有効にするには、`autocommit` パラメータを `1` に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
);
```

パラメータ

host_name

出典データベースインスタンスとなる、Amazon RDS の外部で動作する MySQL インスタンスのホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの REPLICATION CLIENT および REPLICATION SLAVE のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

delay

出典データベースインスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

使用に関する注意事項

マスターユーザーが mysql.rds_set_external_master_with_auto_position を実行する必要があります。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定される MySQL DB インスタンス上で実行する必要があります。

この手順は すべての RDS for MySQL 5.7 バージョン、RDS for MySQL 8.0.26 以降の 8.0 バージョンでサポートされています。

mysql.rds_set_external_master_with_auto_position を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必

ず設定してください。Amazon RDS の外部で実行する MySQL インスタンスに接続するには、replication_user_name および replication_user_password の値を指定する必要があります。これらの値は、MySQL の外部インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE アクセス権限を持つレプリケーションユーザーを示す必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。次に例を示します。

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。次の例では、ドメインの REPLICATION CLIENT ユーザーに、すべてのデータベースの REPLICATION SLAVE および 'repl_user' 特権を付与します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されています。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_master_with_auto_position を呼び出す前に、必ず the section called “mysql.rds_set_external_source_gtid_purged” を呼び出して、外部ソースから指定された GTID 範囲を持つ gtid_purged システム変数を設定します。

mysql.rds_set_external_master_with_auto_position を呼び出して Amazon RDS DB インスタンスをリードレプリカとして設定した後で、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスをスタートできます。mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_master_with_auto_position を呼び出すと、Amazon RDS では、時刻、ユーザー、set master アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

障害復旧のために、このプロシージャを mysql.rds_start_replication_until または mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。mysql.rds_set_external_master_with_auto_position プロシージャを実行して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until_gtid でのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。GTID ベースのレプリケーションの使用に関する詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

例

MySQL DB インスタンス上で実行すると、DB インスタンスが Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定されます。次に例を示します。この例では、MySQL DB インスタンスでのレプリケーションの最小遅延間隔を 1 時間 (3600 秒) に設定します。Amazon RDS の外部で動作する MySQL 出典データベースインスタンスの変更は、少なくとも 1 時間は MySQL DB インスタンスのリードレプリカに適用されません。

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  1,
  3600);
```

mysql.rds_set_external_source_with_auto_position (RDS for MySQL メジャーバージョン 8.4 以降)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定します。このプロシージャは、遅延レプリケーション、および、グローバルトランザクション識別子 (GTID) ベースのレプリケーションも設定します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
);
```

パラメータ

host_name

出典データベースインスタンスとなる、Amazon RDS の外部で動作する MySQL インスタンスのホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの REPLICATION CLIENT および REPLICATION SLAVE のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

delay

出典データベースインスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

使用に関する注意事項

管理ユーザーは、mysql.rds_set_external_source_with_auto_positionプロシージャを実行すべきです。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定される MySQL DB インスタンス上で実行する必要があります。

mysql.rds_set_external_source_with_auto_position を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必ず設定してください。Amazon RDS の外部で実行する MySQL インスタンスに接続するには、replication_user_name および replication_user_password の値を指定する必要があります。これらの値は、MySQL の外部インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE アクセス権限を持つレプリケーションユーザーを示す必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。次に例を示します。

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。次の例では、ドメインの REPLICATION CLIENT ユーザーに、すべてのデータベースの REPLICATION SLAVE および 'repl_user' 特権を付与します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されています。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_source_with_auto_position を呼び出す前に、必ず the section called “mysql.rds_set_external_source_gtid_purged” を呼び出して、外部ソースから指定された GTID 範囲を持つ gtid_purged システム変数を設定します。

mysql.rds_set_external_source_with_auto_position を呼び出して Amazon RDS DB インスタンスをリードレプリカとして設定した後で、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスをスタートできま

す。mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_source_with_auto_position を呼び出すと、Amazon RDS では、時刻、ユーザー、set master アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

障害復旧のために、このプロシージャを mysql.rds_start_replication_until または mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。mysql.rds_set_external_source_with_auto_position プロシージャを実行して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until_gtid でのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。GTID ベースのレプリケーションの使用に関する詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

例

MySQL DB インスタンス上で実行すると、DB インスタンスが Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定されます。次に例を示します。この例では、MySQL DB インスタンスでのレプリケーションの最小遅延間隔を 1 時間 (3600 秒) に設定します。Amazon RDS の外部で動作する MySQL 出典データベースインスタンスの変更は、少なくとも 1 時間は MySQL DB インスタンスのリードレプリカに適用されません。

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  1,
  3600);
```

mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定し、さらに遅延レプリケーションを設定します。

> **Important**
>
> この手順を実行するには、`autocommit` を有効にする必要があります。これを有効にするには、`autocommit` パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_master_with_delay(
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , delay
);
```

パラメータ

host_name

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) のホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する SSH ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって公開されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの REPLICATION CLIENT および REPLICATION SLAVE のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

出典データベースインスタンスのバイナリログの名前には、レプリケーション情報が含まれています。

mysql_binary_log_file_location

mysql_binary_log_file_name バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、SHOW MASTER STATUS出典データベースインスタンス上で実行することによって決定できます。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> MASTER_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

delay

出典データベースインスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

使用に関する注意事項

マスターユーザーが mysql.rds_set_external_master_with_delay を実行する必要があります。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定される MySQL DB インスタンス上で実行する必要があります。

mysql.rds_set_external_master_with_delay を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必ず設定してください。Amazon RDS の外部で実行する MySQL インスタンスに接続するには、replication_user_name および replication_user_password の値を指定する必要があります。これらの値は、MySQL の外部インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE アクセス権限を持つレプリケーションユーザーを示す必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。次に例を示します。

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。次の例では、ドメインの REPLICATION CLIENT ユーザーに、すべてのデータベースの REPLICATION SLAVE および 'repl_user' 特権を付与します。

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されていま

す。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_master_with_delay を呼び出して Amazon RDS DB インスタンスをリードレプリカとして設定した後で、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスをスタートできます。mysql.rds_reset_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_master_with_delay を呼び出すと、Amazon RDS では、時刻、ユーザー、set master アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

障害復旧のために、このプロシージャを mysql.rds_start_replication_until または mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。mysql.rds_set_external_master_with_delay プロシージャを実行して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until でのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。GTID ベースのレプリケーションの使用に関する詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

mysql.rds_set_external_master_with_delay プロシージャは、以下のバージョンの RDS for MySQL で利用できます。

- MySQL のバージョン 8.0 (8.0.26 以降)
- すべての 5.7 バージョン

例

MySQL DB インスタンス上で実行すると、DB インスタンスが Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定されます。次に例を示します。この例で

は、MySQL DB インスタンスでのレプリケーションの最小遅延間隔を 1 時間 (3600 秒) に設定します。Amazon RDS の外部で動作する MySQL 出典データベースインスタンスの変更は、少なくとも 1 時間は MySQL DB インスタンスのリードレプリカに適用されません。

```
call mysql.rds_set_external_master_with_delay(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'SomePassW0rd',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600);
```

mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降)

RDS for MySQL DB インスタンスを、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定し、さらに遅延レプリケーションを設定します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source_with_delay (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , delay
);
```

パラメータ

host_name

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) のホスト名または IP アドレス。

host_port

Amazon RDS の外部で動作する MySQL インスタンス (出典データベースインスタンス) で使用されるポート。ポート番号を変換する SSH ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって公開されるポート番号を指定します。

replication_user_name

Amazon RDS の外部で実行される MySQL インスタンスの `REPLICATION CLIENT` および `REPLICATION SLAVE` のアクセス許可を持つユーザーの ID。外部インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

`replication_user_name` で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

出典データベースインスタンスのバイナリログの名前には、レプリケーション情報が含まれています。

mysql_binary_log_file_location

`mysql_binary_log_file_name` バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、`SHOW MASTER STATUS`出典データベースインスタンス上で実行することによって決定できます。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

delay

出典データベースインスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

使用に関する注意事項

管理ユーザーは、mysql.rds_set_external_source_with_delayプロシージャを実行すべきです。このプロシージャは、Amazon RDS の外部で動作する MySQL インスタンスのリードレプリカとして設定される MySQL DB インスタンス上で実行する必要があります。

mysql.rds_set_external_source_with_delay を実行する前に、Amazon RDS の外部で動作する MySQL インスタンスを出典データベースインスタンスとして必ず設定してください。Amazon RDS の外部で実行する MySQL インスタンスに接続するには、replication_user_name および replication_user_password の値を指定する必要があります。これらの値は、MySQL の外部インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE アクセス権限を持つレプリケーションユーザーを示す必要があります。

MySQL の外部インスタンスを出典データベースインスタンスとして設定するには

1. 選択した MySQL クライアントを使用して、MySQL の外部インスタンスに接続し、レプリケーションに使用されるユーザーアカウントを作成します。次に例を示します。

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. MySQL の外部インスタンスで、REPLICATION CLIENT と REPLICATION SLAVE の特権をレプリケーションユーザーに付与します。次の例では、ドメインの REPLICATION CLIENT ユーザーに、すべてのデータベースの REPLICATION SLAVE および 'repl_user' 特権を付与します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

詳細については、「外部のソースインスタンスを使用したバイナリログファイル位置のレプリケーションの設定」を参照してください。

> **Note**
>
> 可能な場合は、2 つの Amazon RDS DB インスタンス間のレプリケーションを管理するために、リードレプリカを使用することをお勧めします。その場合、このプロシージャとレプリケーション関連のストアドプロシージャだけを使用することをお勧めします。これらのプラクティスにより、Amazon RDS DB インスタンス間で、より複雑なレプリケーショントポロジが有効になります。これらのストアドプロシージャは、主に Amazon RDS 外部で実行されている MySQL インスタンスのレプリケーションの有効化のために提供されていま

す。Amazon RDS DB インスタンス間でのレプリケーションの管理の詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_set_external_source_with_delay を呼び出して Amazon RDS DB インスタンスをリードレプリカとして設定した後で、このリードレプリカで mysql.rds_start_replication を呼び出してレプリケーションプロセスをスタートできます。mysql.rds_reset_external_source (RDS for MySQL メジャーバージョン 8.4 以降) を呼び出して、リードレプリカの設定を削除することもできます。

mysql.rds_set_external_source_with_delay を呼び出すと、Amazon RDS では、時刻、ユーザー、set master アクションが mysql.rds_history テーブルと mysql.rds_replication_status テーブルに記録されます。

障害復旧のために、このプロシージャを mysql.rds_start_replication_until または mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。mysql.rds_set_external_source_with_delay プロシージャを実行して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until でのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。GTID ベースのレプリケーションの使用に関する詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

例

MySQL DB インスタンスで実行すると、次の例に示されているように、DB インスタンスが Amazon RDS の外部で実行されている MySQL インスタンスのリードレプリカとして設定されます。この例では、MySQL DB インスタンスでのレプリケーションの最小遅延間隔を 1 時間 (3,600 秒) に設定します。Amazon RDS の外部で動作する MySQL ソースデータベースインスタンスの変更は、少なくとも 1 時間は MySQL DB インスタンス リードレプリカに適用されません。

```
call mysql.rds_set_external_source_with_delay(
  'Externaldb.example.com',
  3306,
```

```
  'repl_user',
  'SomePassW0rd',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600);
```

mysql.rds_set_external_source_gtid_purged

外部ソースから指定された GTID 範囲を持つ gtid_purged システム変数を設定します。自動測位を使用してレプリケーションを再開するように GTID ベースのレプリケーションを設定するには、gtid_purged 値が必要です。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source_gtid_purged(
  server_uuid
  , start_pos
  , end_pos
);
```

パラメータ

server_uuid

GTID 範囲をインポートする外部サーバーの汎用一意識別子 (UUID)。

start_pos

設定する GTID 範囲の開始位置。

end_pos

設定する GTID 範囲の終了位置。

使用に関する注意事項

mysql.rds_set_external_source_gtid_purged プロシージャは、MySQL 8.0.37 以降の 8.0 バージョンでのみ使用できます。

mysql.rds_set_external_master_with_auto_position (RDS for MySQL メジャーバージョン 8.0 以前)、mysql.rds_set_external_source_with_auto_position (RDS for MySQL メジャーバージョン 8.4 以降)、または mysql.rds_set_external_source_with_auto_position_for_channel を呼び出す前に mysql.rds_set_external_source_gtid_purged を呼び出します。

mysql.rds_set_external_source_gtid_purged を呼び出す前に、データベースのアクティブなレプリケーションチャネルをすべて停止してください。チャネルのステータスを確認するには、SHOW REPLICA STATUS MySQL ステートメントを使用します。チャネルのレプリケーションを停止するには、mysql.rds_stop_replication_for_channel を呼び出します。

指定する GTID 範囲は、既存の GTID_PURGED 値のスーパーセットである必要があります。このストアドプロシージャは、GTID_PURGED 値を設定する前に以下の値をチェックします。

- server_uuid が有効である。
- start_pos の値は、0 より大きく、end_pos の値より小さい。
- end_pos の値が start_pos の値以上。

外部サーバーで設定された GTID に複数の値の範囲が含まれている場合は、異なる GTID 設定値でプロシージャを複数回呼び出すことを検討してください。

mysql.rds_set_external_source_gtid_purged を呼び出すと、Amazon RDS で、時刻、ユーザー、set gtid_purged のアクションが mysql.rds_history テーブルに記録されます。

レプリケーションに使用するバックアップに gtid_purged 値を適切に設定しないと、レプリケーションプロセス中にトランザクションが欠落したり、重複する可能性があります。正しい gtid_purged 値を設定するには、次の手順を実行します。

レプリカで gtid_purged 値を設定するには

1. レプリケーションの開始点として使用するポイントインタイムまたは特定のバックアップファイルを決定します。これは、論理バックアップ (mysqldump ファイル) または物理バックアップ (Amazon RDS スナップショット) です。
2. gtid_executed 値を決定します。この値は、サーバーでコミットされたすべての GTID のセットを表します。この値を取得するには、ソースインスタンスで次のいずれかを実行します。

- バックアップが作成された時点で SQL ステートメント SELECT @@GLOBAL.GTID_EXECUTED; を実行します。
- 関連するオプションがそれぞれのバックアップユーティリティに含まれている場合は、バックアップファイルから値を抽出します。詳細については、MySQL ドキュメントの set-gtid-purged オプションを参照してください。

3. mysql.rds_set_external_source_gtid_purged への呼び出しに使用する gtid_purged 値を決定します。gtid_purged 値には、ソースインスタンスで実行され、レプリケーションに不要になったすべての GTID を含める必要があります。したがって、gtid_purged 値は前のステップで取得した gtid_executed 値のサブセットである必要があります。

   gtid_purged 値を決定するには、バックアップに含まれておらず、レプリケーションに不要になった GTID を特定します。そのためには、バイナリログを分析するか、mysqlbinlog などのツールを使用してバイナリログから消去された GTID を検索します。

   または、バックアップポイントまでのすべてのバイナリログを含む一貫したバックアップがある場合は、gtid_purged 値をバックアップポイントの gtid_executed 値と同じ値に設定できます。
4. バックアップと一致する適切な gtid_purged 値を決定したら、RDS for MySQL DB インスタンスで mysql.rds_set_external_source_gtid_purged ストアドプロシージャを呼び出して値を設定します。

例

MySQL DB インスタンスで実行すると、次の例では、UUID 12345678-abcd-1234-efgh-123456789abc を持つ外部 MySQL サーバーからの GTID 範囲、1 の開始位置、100 の終了位置を設定します。結果の GTID 値は +12345678-abcd-1234-efgh-123456789abc:1-100 に設定されます。

```
CALL mysql.rds_set_external_source_gtid_purged('12345678-abcd-1234-efgh-123456789abc',
 1, 100);
```

mysql.rds_set_master_auto_position (RDS for MySQL メジャーバージョン 8.0 以前)

バイナリログファイルの位置、または、グローバルトランザクション識別子 (GTID) ベースのレプリケーションモードを設定します。

構文

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

パラメータ

auto_position_mode

ファイルの位置に基づくレプリケーション、または GTID ベースのレプリケーションかどうかを示す値:

- 0 - バイナリログファイルの位置に基づくレプリケーション方法を使用します。デフォルトは 0 です。
- 1 - GTID ベースのレプリケーション方法を使用します。

使用に関する注意事項

マスターユーザーが mysql.rds_set_master_auto_position を実行する必要があります。

この手順はすべての RDS for MySQL 5.7 バージョン、RDS for MySQL 8.0.26 以降の 8.0 バージョンでサポートされています。

mysql.rds_set_source_auto_position (RDS for MySQL メジャーバージョン 8.4 以降)

バイナリログファイルの位置、または、グローバルトランザクション識別子 (GTIDs) ベースのレプリケーションモードを設定します。

構文

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

パラメータ

auto_position_mode

ファイルの位置に基づくレプリケーション、または GTID ベースのレプリケーションかどうかを示す値:

- 0 - バイナリログファイルの位置に基づくレプリケーション方法を使用します。デフォルトは 0 です。
- 1 - GTID ベースのレプリケーション方法を使用します。

使用に関する注意事項

管理ユーザーは、mysql.rds_set_source_auto_positionプロシージャを実行すべきです。

mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降)

出典データベースインスタンスから現在のリードレプリカへのレプリケーションを遅延させる最小秒数を設定します。出典データベースインスタンスからのレプリケーションを遅延させるリードレプリカに接続している場合に、このプロシージャを使用します。

構文

```
CALL mysql.rds_set_source_delay(
delay
);
```

パラメータ

delay

出典データベースインスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 259,200 秒 (72 時間) です。

使用に関する注意事項

管理ユーザーは、mysql.rds_set_source_delay プロシージャを実行する必要があります。

障害復旧のために、このプロシージャを mysql.rds_start_replication_until ストアドプロシージャ、または、mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。mysql.rds_set_source_delay プロシージャを実行して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until、または mysql.rds_start_replication_until_gtid プロシージャによりレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。

例

ソースデータベースインスタンスから現在のリードレプリカへのレプリケーションを少なくとも 1 時間 (3,600 秒) 遅延させるには:

```
CALL mysql.rds_set_source_delay(3600);
```

mysql.rds_skip_repl_error

MySQL DB リードレプリカのレプリケーションエラーをスキップして、削除します。

構文

```
CALL mysql.rds_skip_repl_error;
```

使用に関する注意事項

マスターユーザーはリードレプリカに対して mysql.rds_skip_repl_error プロシージャを実行する必要があります。この手順の詳細については、「mysql.rds_skip_repl_error の手順を呼び出します。」(現在のレプリケーションエラーをスキップする) を参照してください。

MySQL の SHOW REPLICA STATUS\G コマンドを実行して、エラーがあるかどうかを判断します。レプリケーションエラーが重要ではない場合、mysql.rds_skip_repl_error を実行して、エラーをスキップすることができます。複数のエラーがある場合、mysql.rds_skip_repl_error では、初期のエラーを削除してから、他のエラーが存在することを警告します。その後で SHOW REPLICA STATUS\G を使用して、次のエラーに対処するための適切な対応方法を判断することができます。戻り値の詳細については、MySQL ドキュメントの「SHOW REPLICA STATUS statement」を参照してください。

Amazon RDS でのレプリケーションエラーの対処方法の詳細については「MySQL リードレプリカに関する問題のトラブルシューティング」を参照してください。

レプリケーション停止エラー

mysql.rds_skip_repl_error プロシージャを呼び出すと、レプリカがダウンしているか無効であることを示すエラーメッセージが表示されることがあります。

このエラーメッセージは、リードレプリカではなくプライマリインスタンスでプロシージャを実行した場合に表示されます。このプロシージャを機能させるには、リードレプリカに対してこのプロシージャを実行する必要があります。

このエラーメッセージは、リードレプリカに対してプロシージャを実行したが、レプリケーションを正常に再開できない場合にも表示されることがあります。

多数のエラーをスキップする必要がある場合は、レプリケーションの遅延により、バイナリログ (binlog) ファイルがデフォルトの保持期間を超えて増大する場合があります。この場合、binlog ファイルがリードレプリカで再生される前に破棄されるため、致命的なエラーが発生することがあります。この破棄によりレプリケーションが停止し、mysql.rds_skip_repl_error コマンドを呼び出してレプリケーションエラーをスキップすることができなくなります。

この問題は、出典データベースインスタンスでバイナリログファイルの保持時間を増加させることで軽減できます。バイナリログ保持時間を長くすると、レプリケーションを再開し、必要に応じて mysql.rds_skip_repl_error コマンドを使用できるようになります。

バイナリログの保持期間を設定するには、「mysql.rds_set_configuration」の手順を使用して、DB クラスターのバイナリログファイルの保持期間に合わせて、'binlog retention hours' の設定パラメータを指定します。以下の例では、バイナリログファイルの保持期間を 48 時間に設定しています。

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

RDS for MySQL DB インスタンスからのレプリケーションを開始します。

> **Note**
>
> mysql.rds_start_replication_until または mysql.rds_start_replication_until_gtid ストアドプロシージャを使用して、RDS for MySQL DB インスタンスからのレプリケーションを開始し、指定したバイナリログファイルの場所でレプリケーションを停止できます。

構文

```
CALL mysql.rds_start_replication;
```

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication を実行する必要があります。

Amazon RDS の外部の MySQL インスタンスからデータをインポートするには、mysql.rds_set_external_master (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) または mysql.rds_set_external_source (RDS for MySQL メジャーバージョン 8.4 以降) を呼び出してレプリケーション設定を構築した後、リードレプリカに対して mysql.rds_start_replication を呼び出して、レプリケーションプロセスを開始します。詳細については、「Amazon RDS for MySQL DB インスタンスへのバックアップの復元」を参照してください。

Amazon RDS の外部の MySQL インスタンスにデータをエクスポートするには、リードレプリカに対して mysql.rds_start_replication と mysql.rds_stop_replication を呼び出して、バイナリログの消去などのレプリケーションアクションを制御します。詳細については、「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

リードレプリカで mysql.rds_start_replication を呼び出すことで、mysql.rds_stop_replication の呼び出しによって前に停止したレプリケーションプロセスを再開することもできます。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

mysql.rds_start_replication_until

RDS for MySQL DB インスタンスからレプリケーションを開始し、指定したバイナリログファイルの場所でレプリケーションを停止します。

構文

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

パラメータ

replication_log_file

レプリケーション情報を含む出典データベースインスタンス上のバイナリログの名前。

replication_stop_point

replication_log_file バイナリログ内でレプリケーションが停止する場所。

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication_until を実行する必要があります。

mysql.rds_start_replication_until プロシージャは、以下のバージョンの RDS for MySQL で利用できます。

- すべての RDS for MySQL 8.4 バージョン
- MySQL のバージョン 8.0 (8.0.26 以降)
- すべての 5.7 バージョン

このプロシージャは、災害対策用の遅延レプリケーションで使用できます。遅延レプリケーションを設定している場合は、このプロシージャを使用して、遅延したリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。このプロシージャでのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

次のストアドプロシージャを使用して遅延レプリケーションを設定できます。

- mysql.rds_set_configuration
- mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降)

replication_log_file パラメータに指定するファイル名は、出典データベースインスタンスの binlog ファイル名と一致する必要があります。

replication_stop_pointパラメータで指定した停止場所が過去の時点である場合、レプリケーションは即座に停止します。

例

次の例では、レプリケーションをスタートし、120 バイナリログファイルの場所 mysql-bin-changelog.000777 に達するまで変更をレプリケートします。

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
```

```
  120);
```

mysql.rds_stop_replication

MySQL DB インスタンスからのレプリケーションを停止します。

構文

```
CALL mysql.rds_stop_replication;
```

使用に関する注意事項

マスターユーザーが mysql.rds_stop_replication を実行する必要があります。

Amazon RDS の外部で動作する MySQL インスタンスからデータをインポートするようにレプリケーションを設定している場合は、リードレプリカで mysql.rds_stop_replication を呼び出して、インポートが完了した後でレプリケーションプロセスを停止することができます。詳細については、「Amazon RDS for MySQL DB インスタンスへのバックアップの復元」を参照してください。

Amazon RDS の外部にある MySQL インスタンスにデータをエクスポートするようにレプリケーションを設定している場合は、リードレプリカで mysql.rds_start_replication と mysql.rds_stop_replication を呼び出して、バイナリログの消去などのレプリケーションアクションを制御できます。詳細については、「レプリケーションを使用した MySQL DB インスタンスからのデータのエクスポート」を参照してください。

mysql.rds_stop_replication を使用して、2 つ Amazon RDS DB インスタンス間のレプリケーションを停止することもできます。通常、レプリケーションの停止は、リードレプリカでの長時間のオペレーション (リードレプリカで大規模なインデックスを作成するなど) を実行するために行います。停止したレプリケーションプロセスは、リードレプリカで mysql.rds_start_replication を呼び出して再開できます。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

セッションやクエリの終了

次のストアドプロシージャは、セッションまたはクエリを終了します。

トピック

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

MySQL サーバーへの接続を終了します。

構文

```
CALL mysql.rds_kill(processID);
```

パラメータ

processID

終了する接続スレッドの識別子。

使用に関する注意事項

MySQL サーバーへの個々の接続は別々のスレッドで実行されます。接続を終了するには、`mysql.rds_kill` プロシージャを使用して、その接続のスレッド ID を渡します。スレッド ID を取得するには、MySQL の PROCESSLIST コマンドを使用します。

制限の詳細については、「MySQL ストアドプロシージャの制限事項」を参照してください。

例

次の例では、4243 のスレッド ID を持つ接続を終了します。

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

MySQL サーバーに対して実行中のクエリを終了します。

構文

```
CALL mysql.rds_kill_query(processID);
```

パラメータ

processID

終了するクエリを実行しているプロセスまたはスレッドの ID。

使用に関する注意事項

MySQL サーバーに対して実行しているクエリを終了するには、mysql_rds_kill_query プロシージャを使用して、クエリを実行しているスレッドの接続 ID を渡します。その後、プロシージャはクエリを終了しますが、接続自体はそのまま残ります。

ID を取得するには、MySQL INFORMATION_SCHEMA PROCESSLIST テーブルまたは MySQL SHOW PROCESSLIST コマンドを使用します。SHOW PROCESSLIST または SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST の ID 列の値は *processID* です。

制限の詳細については、「MySQL ストアドプロシージャの制限事項」を参照してください。

例

次の例では、クエリスレッド ID が 230040 のクエリを停止します。

```
CALL mysql.rds_kill_query(230040);
```

アクティブ/アクティブクラスターの管理

以下のストアドプロシージャは、RDS for MySQL のアクティブ/アクティブクラスターを設定および管理します。詳細については、「the section called “アクティブ/アクティブクラスターの設定”」を参照してください。

これらのストアドプロシージャは、次のバージョンを実行している RDS for MySQL DB インスタンスでのみ使用できます。

- すべての MySQL 8.4 バージョン
- MySQL 8.0.35 以降のマイナーバージョン

トピック

- mysql.rds_group_replication_advance_gtid
- mysql.rds_group_replication_create_user
- mysql.rds_group_replication_set_recovery_channel
- mysql.rds_group_replication_start
- mysql.rds_group_replication_stop

mysql.rds_group_replication_advance_gtid

現在の DB インスタンスにプレースホルダー GTID を作成します。

構文

```
CALL mysql.rds_group_replication_advance_gtid(
begin_id
, end_id
, server_uuid
);
```

パラメータ

begin_id

作成する開始トランザクション ID。

end_id

作成する終了トランザクション ID。

begin_id

作成するトランザクションの group_replication_group_name。group_replication_group_name は、DB インスタンスに関連付けられた DB パラメータグループの UUID として指定されます。

使用に関する注意事項

アクティブ/アクティブクラスターでは、DB インスタンスがグループに参加するには、新しい DB インスタンスで実行されるすべての GTID トランザクションが、クラスター内の他のメンバーに存在する必要があります。通常とは異なる場合、インスタンスをグループに結合する前にトランザクションが実行されると、新しい DB インスタンスで、より多くのトランザクションが発生する可能性があります。この場合、既存のトランザクションを削除することはできませんが、この手順を使用して、グループ内の他の DB インスタンスに対応するプレースホルダー GTID を作成できます。その前に、トランザクションがレプリケートされたデータ に影響を与えないことを確認します。

この手順を呼び出すと、server_uuid:begin_id-end_id の GTID トランザクションが空のコンテンツで作成されます。レプリケーションの問題を回避するには、この手順を他の条件で使用しないでください。

> **Important**
>
> アクティブ/アクティブクラスターが正常に機能している場合は、この手順を呼び出さないでください。この手順は、作成するトランザクションの考えられる結果を理解しない限り、呼び出さないでください。この手順を呼び出すと、データが矛盾する可能性があります。

例

次の例は、現在の DB インスタンスにプレースホルダー GTID を作成します。

```
CALL mysql.rds_group_replication_advance_gtid(5, 6,
 '11111111-2222-3333-4444-555555555555');
```

mysql.rds_group_replication_create_user

DB インスタンスにグループレプリケーション用のレプリケーションユーザー rdsgrprepladmin を作成します。

構文

```
CALL mysql.rds_group_replication_create_user(
replication_user_password
);
```

パラメータ

replication_user_password

レプリケーションユーザー rdsgrprepladmin のパスワード。

使用に関する注意事項

- レプリケーションユーザー rdsgrprepladmin のパスワードは、アクティブ/アクティブクラスター内のすべての DB インスタンスで同じである必要があります。
- rdsgrprepladmin ユーザー名は、グループレプリケーション接続用に予約されています。マスターユーザーを含め、他のユーザーがこのユーザー名を持つことはできません。

例

次の例は、DB インスタンスにグループレプリケーション用のレプリケーションユーザー rdsgrprepladmin を作成します。

```
CALL mysql.rds_group_replication_create_user('password');
```

mysql.rds_group_replication_set_recovery_channel

アクティブ/アクティブクラスターの group_replication_recovery チャネルを設定します。この手順では、予約済み rdsgrprepladmin ユーザーを使用してチャネルを設定します。

構文

```
CALL mysql.rds_group_replication_set_recovery_channel(
```

```
replication_user_password);
```

パラメータ

replication_user_password

レプリケーションユーザー rdsgrprepladmin のパスワード。

使用に関する注意事項

レプリケーションユーザー rdsgrprepladmin のパスワードは、アクティブ/アクティブクラスター内のすべての DB インスタンスで同じである必要があります。mysql.rds_group_replication_create_user の呼び出しは、パスワードを指定します。

例

次の例は、アクティブ/アクティブクラスターの group_replication_recovery チャネルを設定します。

```
CALL mysql.rds_group_replication_set_recovery_channel('password');
```

mysql.rds_group_replication_start

現在の DB インスタンスでグループレプリケーションを開始します。

構文

```
CALL mysql.rds_group_replication_start(
bootstrap
);
```

パラメータ

bootstrap

新しいグループを初期化するか、既存のグループに参加するかを指定する値。1 は、現在の DB インスタンスで新しいグループを初期化します。0 は、DB インスタンスに関連付けられた DB パラメータグループの group_replication_group_seeds パラメータで定義されたエンドポイントに接続することで、現在の DB インスタンスを既存のグループに結合します。

例

次の例は、現在の DB インスタンスで新しいグループを初期化します。

```
CALL mysql.rds_group_replication_start(1);
```

mysql.rds_group_replication_stop

現在の DB インスタンスでのグループレプリケーションを停止します。

構文

```
CALL mysql.rds_group_replication_stop();
```

使用に関する注意事項

DB インスタンスでのレプリケーションを停止しても、アクティブ/アクティブクラスター内の他の DB インスタンスには影響しません。

マルチソースレプリケーションの管理

以下のストアドプロシージャは、RDS for MySQL マルチソースレプリカでレプリケーションチャネルを設定および管理します。詳細については、「the section called “マルチソースレプリケーションの設定”」を参照してください。

これらのストアドプロシージャは、次のエンジンバージョンを実行している RDS for MySQL DB インスタンスでのみ使用できます。

- すべての 8.4 バージョン
- 8.0.35 以降のマイナーバージョン
- 5.7.44 以降のマイナーバージョン

`caching_sha2_password` で設定されたレプリケーションユーザーによるレプリケーションを、ストアドプロシージャを使用して管理する場合は、`ssl_encryption=1` を指定して TLS を設定する必要があります。`caching_sha2_password` は RDS for MySQL 8.4 のデフォルトの認証プラグインです。

> **Note**
>
> このドキュメントではソース DB インスタンスを RDS for MySQL DB インスタンスと呼びますが、これらの手順は Amazon RDS の外部で実行されている MySQL インスタンスでも機能します。

トピック

- mysql.rds_next_source_log_for_channel
- mysql.rds_reset_external_source_for_channel
- mysql.rds_set_external_source_for_channel
- mysql.rds_set_external_source_with_auto_position_for_channel
- mysql.rds_set_external_source_with_delay_for_channel
- mysql.rds_set_source_auto_position_for_channel
- mysql.rds_set_source_delay_for_channel
- mysql.rds_skip_repl_error_for_channel
- mysql.rds_start_replication_for_channel

- mysql.rds_start_replication_until_for_channel
- mysql.rds_start_replication_until_gtid_for_channel
- mysql.rds_stop_replication_for_channel

mysql.rds_next_source_log_for_channel

ソース DB インスタンスログの位置を、チャネルのソース DB インスタンスの次のバイナリログの先頭に変更します。このプロシージャは、マルチソースレプリカでレプリケーション I/O エラー 1236 が発生した場合にのみ使用してください。

構文

```
CALL mysql.rds_next_source_log_for_channel(
curr_master_log,
channel_name
);
```

パラメータ

curr_master_log

現在のソースログファイルのインデックス。例えば、現在のファイルが mysql-bin-changelog.012345 という名前の場合は、インデックスは 12345 になります。現在のソースログファイルの名前を調べるには、SHOW REPLICA STATUS FOR CHANNEL '*channel_name*' コマンドを実行し、Source_Log_File フィールドを確認します。

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_next_source_log_for_channel を実行する必要があります。例えば、IO_Thread エラーが発生した場合、このプロシージャを使用して現在のバイナリログファイルのすべてのイベントをスキップし、channel_name で指定されたチャネルの次のバイナリログファイルからレプリケーションを再開できます。

例

マルチソースレプリカのチャネルでレプリケーションが失敗するとします。マルチソースレプリカで SHOW REPLICA STATUS FOR CHANNEL 'channel_1'\G を実行すると、次の結果が返されます。

```
mysql> SHOW REPLICA STATUS FOR CHANNEL 'channel_1'\G
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: ReplicationUser
                  Source_Port: 3306
                Connect_Retry: 60
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: replica-relay-bin.000003
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:.
              .
              .
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
               .
               .
                 Channel_name: channel_1
              .
              .
 -- Some fields are omitted in this example output
```

Last_IO_Errno フィールドはインスタンスが I/O エラー 1236 を受け取ったことを示します。Source_Log_File フィールドは、ファイル名が mysql-bin-changelog.012345 であることを示しています。これは、ログファイルのインデックスが 12345 で

あることを表しています。エラーを解決するには、以下のパラメータを使用して mysql.rds_next_source_log_for_channel を呼び出すことができます。

```
CALL mysql.rds_next_source_log_for_channel(12345,'channel_1');
```

mysql.rds_reset_external_source_for_channel

指定されたチャネルでのレプリケーションプロセスを停止し、マルチソースレプリカからチャネルおよび関連する設定を削除します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_reset_external_source_for_channel (channel_name);
```

パラメータ

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_reset_external_source_for_channel を実行する必要があります。このプロシージャは、削除されるチャネルに属するすべてのリレーログを削除します。

mysql.rds_set_external_source_for_channel

RDS for MySQL DB インスタンスのレプリケーションチャネルを設定して、別の RDS for MySQL DB インスタンスからデータをレプリケートします。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

> **Note**
>
> 代わりに the section called “mysql.rds_set_external_source_with_delay_for_channel” ストアドプロシージャを使用して、このチャネルを遅延レプリケーションで設定できます。

構文

```
CALL mysql.rds_set_external_source_for_channel (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
  , channel_name
);
```

パラメータ

host_name

RDS for MySQL ソース DB インスタンスのホスト名または IP アドレス。

host_port

RDS for MySQL ソース DB インスタンスで使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

RDS for MySQL ソース DB インスタンスの REPLICATION CLIENT および REPLICATION SLAVE アクセス許可を持つユーザーの ID。ソース DB インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

レプリケーション情報を含むソース DB インスタンスのバイナリログの名前。

mysql_binary_log_file_location

mysql_binary_log_file_name バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、ソース DB インスタンスで SHOW BINARY LOG STATUS を実行することによって決定できます。

> **Note**
>
> MySQL の旧バージョンは、SHOW BINARY LOG STATUS ではなく SHOW MASTER STATUS を使用していました。8.4 より前の MySQL バージョンを使用している場合は、SHOW MASTER STATUS を使用します。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

channel_name

レプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_set_external_source_for_channel を実行する必要があります。このプロシージャは、レプリケーションチャネルを作成するターゲット RDS for MySQL DB インスタンスで実行する必要があります。

mysql.rds_set_external_source_for_channel を実行する前に、マルチソースレプリカに必要な権限を持つレプリケーションユーザーをソース DB インスタンスに設定します。マルチソースレプリカをソース DB インスタンスに接続するには、ソース DB インスタンスに対する REPLICATION CLIENT および REPLICATION SLAVE アクセス許可を持つレプリケーションユーザーの replication_user_name および replication_user_password の値を指定する必要があります。

ソース DB インスタンスでレプリケーションユーザーを設定するには

1. 選択した MySQL クライアントを使用して、ソース DB インスタンスに接続し、レプリケーションに使用するユーザーアカウントを作成します。以下に例を示します。

   > **Important**
   >
   > セキュリティのベストプラクティスとして、次の例に示すプレースホルダー値以外のパスワードを指定します。

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. ソース DB インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE 権限をレプリケーションユーザーに付与します。次の例では、ドメインの「repl_user」ユーザーに、すべてのデータベースの REPLICATION CLIENT および REPLICATION SLAVE 権限を付与します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

暗号化されたレプリケーションを使用するには、SSL 接続を使用するようにソース DB インスタンスを設定します。

mysql.rds_set_external_source_for_channel を呼び出し、このレプリケーションチャネルを設定した後、レプリカに対して mysql.rds_start_replication_for_channel を呼び出してチャネルでレプリケーションプロセスを開始できます。the section called "mysql.rds_reset_external_source_for_channel" を呼び出してチャンネルのレプリケーションを停止し、レプリカからチャンネル設定を削除できます。

mysql.rds_set_external_source_for_channel を呼び出すと、Amazon RDS は、set channel source の時間、ユーザー、およびアクションをチャネル固有の詳細なしで mysql.rds_history テーブルに記録し、mysql.rds_replication_status テーブルにチャネル名を記録します。この情報は、内部使用およびモニタリングの目的でのみ記録されます。監査目的で完全なプロシージャ呼び出しを記録するには、アプリケーションの特定の要件に基づいて、監査ログまたは一般ログを有効にすることを検討してください。

例

RDS for MySQL DB インスタンスで実行すると、次の例は、ホスト sourcedb.example.comとポート 3306 で指定されたソースからデータをレプリケートするように、この DB インスタンスで channel_1 という名前のレプリケーションチャネルを設定します。

```
call mysql.rds_set_external_source_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  0,
  'channel_1');
```

mysql.rds_set_external_source_with_auto_position_for_channel

オプションのレプリケーション遅延を使用して、RDS for MySQL DB インスタンスのレプリケーションチャネルを設定します。このレプリケーションは、グローバルトランザクション識別子 (GTID) に基づきます。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source_with_auto_position_for_channel (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
  , delay
  , channel_name
);
```

パラメータ

host_name

RDS for MySQL ソース DB インスタンスのホスト名または IP アドレス。

host_port

RDS for MySQL ソース DB インスタンスで使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

RDS for MySQL ソース DB インスタンスの REPLICATION CLIENT および REPLICATION SLAVE アクセス許可を持つユーザーの ID。ソース DB インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

delay

ソース DB インスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

channel_name

レプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが `mysql.rds_set_external_source_with_auto_position_for_channel` を実行する必要があります。このプロシージャは、レプリケーションチャネルを作成するターゲット RDS for MySQL DB インスタンスで実行する必要があります。

`rds_set_external_source_with_auto_position_for_channel` を実行する前に、マルチソースレプリカに必要な権限を持つレプリケーションユーザーをソース DB インスタンスに設定します。マルチソースレプリカをソース DB インスタンスに接続するには、ソース DB インスタンスに対する `REPLICATION CLIENT` および `REPLICATION SLAVE` アクセス許可を持つレプリケーションユーザーの `replication_user_name` および `replication_user_password` の値を指定する必要があります。

ソース DB インスタンスでレプリケーションユーザーを設定するには

1. 選択した MySQL クライアントを使用して、ソース DB インスタンスに接続し、レプリケーションに使用するユーザーアカウントを作成します。以下に例を示します。

   > **Important**
   >
   > セキュリティのベストプラクティスとして、次の例に示すプレースホルダー値以外のパスワードを指定します。

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. ソース DB インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE 権限をレプリケーションユーザーに付与します。次の例では、ドメインの 「repl_user」ユーザーに、すべてのデータベースの REPLICATION CLIENT および REPLICATION SLAVE 権限を付与します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

暗号化されたレプリケーションを使用するには、SSL 接続を使用するようにソース DB インスタンスを設定します。

mysql.rds_set_external_source_with_auto_position_for_channel を呼び出す前に、必ず the section called “mysql.rds_set_external_source_gtid_purged” を呼び出して、外部ソースから指定された GTID 範囲を持つ gtid_purged システム変数を設定します。

mysql.rds_set_external_source_with_auto_position_for_channel を呼び出して、Amazon RDS DB インスタンスを特定のチャネルのリードレプリカとして設定した後、このリードレプリカで the section called “mysql.rds_start_replication_for_channel” を呼び出して、そのチャネルのレプリケーションプロセスを開始できます。

mysql.rds_set_external_source_with_auto_position_for_channel を呼び出し、このレプリケーションチャネルを設定した後、レプリカに対して mysql.rds_start_replication_for_channel を呼び出してチャネルでレプリケーションプロセスを開始できます。the section called “mysql.rds_reset_external_source_for_channel” を呼び出してチャンネルのレプリケーションを停止し、レプリカからチャンネル設定を削除できます。

例

RDS for MySQL DB インスタンスで実行すると、次の例は、この DB インスタンスで channel_1 という名前のレプリケーションチャネルを設定して、ホスト sourcedb.example.com とポート 3306 で指定されたソースからデータをレプリケートします。最小レプリケーション遅延は 1 時間 (3,600 秒) に設定されます。つまり、ソース RDS for MySQL DB インスタンスからの変更は、少なくとも 1 時間はマルチソースレプリカに適用されません。

```
call mysql.rds_set_external_source_with_auto_position_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  1,
  3600,
  'channel_1');
```

mysql.rds_set_external_source_with_delay_for_channel

指定されたレプリケーション遅延を使用して、RDS for MySQL DB インスタンスのレプリケーションチャネルを設定します。

> **Important**
>
> この手順を実行するには、autocommit を有効にする必要があります。これを有効にするには、autocommit パラメータを 1 に設定します。パラメータの変更については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

構文

```
CALL mysql.rds_set_external_source_with_delay_for_channel (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
```

```
  , delay
  , channel_name
);
```

パラメータ

host_name

RDS for MySQL ソース DB インスタンスのホスト名または IP アドレス。

host_port

RDS for MySQL ソース DB インスタンスで使用されるポート。ポート番号を変換する Secure Shell (SSH) ポートのレプリケーションがネットワーク設定に含まれる場合、SSH によって発行されるポート番号を指定します。

replication_user_name

RDS for MySQL ソース DB インスタンスの REPLICATION CLIENT および REPLICATION SLAVE アクセス許可を持つユーザーの ID。ソース DB インスタンスのレプリケーション専用のアカウントを使用することをお勧めします。

replication_user_password

replication_user_name で指定されたユーザー ID のパスワード。

mysql_binary_log_file_name

ソース DB インスタンスのバイナリログの名前には、レプリケーション情報が含まれています。

mysql_binary_log_file_location

mysql_binary_log_file_name バイナリログ内の場所。レプリケーションでは、この場所からレプリケーション情報の読み取りをスタートします。

binlog ファイルの名前と場所は、SHOW BINARY LOG STATUS出典データベースインスタンス上で実行することによって決定できます。

> **Note**
>
> MySQL の旧バージョンは、SHOW BINARY LOG STATUS ではなく SHOW MASTER STATUS を使用していました。8.4 より前の MySQL バージョンを使用している場合は、SHOW MASTER STATUS を使用します。

ssl_encryption

レプリケーション接続で Secure Socket Layer (SSL) 暗号化を使用するかどうかを指定する値。1 は SSL 暗号化を使用することを指定し、0 は暗号化を使用しないことを指定します。デフォルトは 0 です。

> **Note**
>
> SOURCE_SSL_VERIFY_SERVER_CERT オプションはサポートされていません。このオプションは 0 に設定されます。つまり、接続は暗号化されますが、証明書は検証されません。

delay

ソース DB インスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

channel_name

レプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_set_external_source_with_delay_for_channel を実行する必要があります。このプロシージャは、レプリケーションチャネルを作成するターゲット RDS for MySQL DB インスタンスで実行する必要があります。

mysql.rds_set_external_source_with_delay_for_channel を実行する前に、マルチソースレプリカに必要な権限を持つレプリケーションユーザーをソース DB インスタンスに設定します。マルチソースレプリカをソース DB インスタンスに接続するには、ソース DB インスタンスに対する REPLICATION CLIENT および REPLICATION SLAVE アクセス許可を持つレプリケーションユーザーの replication_user_name および replication_user_password の値を指定する必要があります。

ソース DB インスタンスでレプリケーションユーザーを設定するには

1. 選択した MySQL クライアントを使用して、ソース DB インスタンスに接続し、レプリケーションに使用するユーザーアカウントを作成します。以下に例を示します。

   > **Important**
   >
   > セキュリティのベストプラクティスとして、次の例に示すプレースホルダー値以外のパスワードを指定します。

   ```
   CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
   ```

2. ソース DB インスタンスで、REPLICATION CLIENT および REPLICATION SLAVE 権限をレプリケーションユーザーに付与します。次の例では、ドメインの「repl_user」ユーザーに、すべてのデータベースの REPLICATION CLIENT および REPLICATION SLAVE 権限を付与します。

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com';
   ```

暗号化されたレプリケーションを使用するには、SSL 接続を使用するようにソース DB インスタンスを設定します。

mysql.rds_set_external_source_with_delay_for_channel を呼び出し、このレプリケーションチャネルを設定した後、レプリカに対して mysql.rds_start_replication_for_channel を呼び出してチャネルでレプリケーションプロセスを開始できます。the section called “mysql.rds_reset_external_source_for_channel” を呼び出してチャンネルのレプリケーションを停止し、レプリカからチャンネル設定を削除できます。

mysql.rds_set_external_source_with_delay_for_channel を呼び出すと、Amazon RDS は、set channel source の時間、ユーザー、およびアクションをチャネル固有の詳細なしで mysql.rds_history テーブルに記録し、mysql.rds_replication_status テーブルにチャネル名を記録します。この情報は、内部使用およびモニタリングの目的でのみ記録されます。監査目的で完全なプロシージャ呼び出しを記録するには、アプリケーションの特定の要件に基づいて、監査ログまたは一般ログを有効にすることを検討してください。

例

RDS for MySQL DB インスタンスで実行すると、次の例は、この DB インスタンスで channel_1 という名前のレプリケーションチャネルを設定して、ホスト sourcedb.example.com とポート 3306 で指定されたソースからデータをレプリケートします。最小レプリケーション遅延は 1 時間 (3,600 秒) に設定されます。つまり、ソース RDS for MySQL DB インスタンスからの変更は、少なくとも 1 時間はマルチソースレプリカに適用されません。

```
call mysql.rds_set_external_source_with_delay_for_channel(
  'sourcedb.example.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.000777',
  120,
  1,
  3600,
  'channel_1');
```

mysql.rds_set_source_auto_position_for_channel

バイナリログファイルの位置、またはグローバルトランザクション識別子 (GTID) に基づいて、指定されたチャネルのレプリケーションモードを設定します。

構文

```
CALL mysql.rds_set_source_auto_position_for_channel (
auto_position_mode
 , channel_name
);
```

パラメータ

auto_position_mode

ファイルの位置に基づくレプリケーション、または GTID ベースのレプリケーションかどうかを示す値:

- 0 - バイナリログファイルの位置に基づくレプリケーション方法を使用します。デフォルトは 0 です。
- 1 - GTID ベースのレプリケーション方法を使用します。

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_set_source_auto_position_for_channel を実行する必要があります。このプロシージャは、指定されたチャネルでレプリケーションを再起動して、指定された自動位置モードを適用します。

例

次の例は、channel_1 の自動位置モードを、GTID ベースのレプリケーション方法を使用するように設定します。

```
call mysql.rds_set_source_auto_position_for_channel(1,'channel_1');
```

mysql.rds_set_source_delay_for_channel

指定されたチャネルについて、ソースデータベースインスタンスからマルチソースレプリカへのレプリケーションを遅延させる最小秒数を設定します。

構文

```
CALL mysql.rds_set_source_delay_for_channel(delay, channel_name);
```

パラメータ

delay

ソース DB インスタンスからのレプリケーションを遅延させる最小秒数。

このパラメータの上限は 1 日 (86400 秒) です。

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_set_source_delay_for_channel を実行する必要があります。このプロシージャを使用するには、まず、mysql.rds_stop_replication_for_channel を呼び出して、レプリケーションを停止します。次に、このプロシージャを呼び出して、レプリケーションの遅延値を設定します。遅延が設定されたら、mysql.rds_start_replication_for_channel を呼び出して、レプリケーションを再開します。

例

次の例は、マルチソースレプリカの channel_1 のソースデータベースインスタンスからのレプリケーションの遅延を少なくとも 1 時間 (3,600 秒) に設定します。

```
CALL mysql.rds_set_source_delay_for_channel(3600,'channel_1');
```

mysql.rds_skip_repl_error_for_channel

バイナリログイベントをスキップし、指定したチャネルの MySQL DB マルチソースレプリカのレプリケーションエラーを削除します。

構文

```
CALL mysql.rds_skip_repl_error_for_channel(channel_name);
```

パラメータ

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーはリードレプリカに対して mysql.rds_skip_repl_error_for_channel プロシージャを実行する必要があります。mysql.rds_skip_repl_error を使用してリードレプリカ

のエラーをスキップする場合と同様の方法で、このプロシージャを使用できます。詳細については、を参照してください。mysql.rds_skip_repl_error の手順を呼び出します。

> **Note**
>
> GTID ベースのレプリケーションのエラーをスキップするには、代わりにプロシージャ the section called “mysql.rds_skip_transaction_with_gtid” を使用することをお勧めします。

MySQL の SHOW REPLICA STATUS FOR CHANNEL '*channel_name*'\G コマンドを実行して、エラーがあるかどうかを判断します。レプリケーションエラーが重要ではない場合、mysql.rds_skip_repl_error_for_channel を実行して、エラーをスキップすることができます。複数のエラーがある場合、mysql.rds_skip_repl_error_for_channel は、指定されたレプリケーションチャネルの最初のエラーを削除してから、他にもエラーが存在することを警告します。その後で SHOW REPLICA STATUS FOR CHANNEL '*channel_name*'\G を使用して、次のエラーに対処するための適切な対応方法を判断することができます。戻り値の詳細については、MySQL ドキュメントの「SHOW REPLICA STATUS statement」を参照してください。

mysql.rds_start_replication_for_channel

RDS for MySQL DB インスタンスから指定されたチャネルのマルチソースレプリカへのレプリケーションを開始します。

> **Note**
>
> mysql.rds_start_replication_until_for_channel、または mysql.rds_start_replication_until_gtid_for_channel ストアドプロシージャを使用して、RDS for MySQL DB インスタンスからレプリケーションをスタートし、指定したバイナリログファイルの場所でレプリケーションを停止できます。

構文

```
CALL mysql.rds_start_replication_for_channel(channel_name);
```

パラメータ

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication_for_channel を実行する必要があります。ソース RDS for MySQL DB インスタンスからデータをインポートした後、マルチソースレプリカでこのコマンドを実行して、指定したチャネルでレプリケーションを開始します。

例

次の例では、マルチソースレプリカの channel_1 でレプリケーションを開始します。

```
CALL mysql.rds_start_replication_for_channel('channel_1');
```

mysql.rds_start_replication_until_for_channel

指定されたチャネルの RDS for MySQL DB インスタンスからレプリケーションを開始し、指定されたバイナリログファイルの場所でレプリケーションを停止します。

構文

```
CALL mysql.rds_start_replication_until_for_channel (
replication_log_file
  , replication_stop_point
  , channel_name
);
```

パラメータ

replication_log_file

ソース DB インスタンスのバイナリログの名前には、レプリケーション情報が含まれています。

replication_stop_point

replication_log_file バイナリログ内でレプリケーションが停止する場所。

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication_until_for_channel を実行する必要があります。このプロシージャでは、レプリケーションが開始され、指定されたバイナリログファイルの位置に達すると停止します。このプロシージャでは、SQL_THREAD と IO_THREAD の両方が停止します。

replication_log_file パラメータで指定されたファイル名は、ソース DB インスタンスの binlog ファイル名と一致する必要があります。

replication_stop_point パラメータで指定した停止場所が過去の時点である場合、レプリケーションは即座に停止します。

例

次の例では、channel_1 でレプリケーションを開始し、mysql-bin-changelog.000777 バイナリログファイルの位置 120 に達するまで変更をレプリケートします。

```
call mysql.rds_start_replication_until_for_channel(
  'mysql-bin-changelog.000777',
  120,
  'channel_1'
  );
```

mysql.rds_start_replication_until_gtid_for_channel

指定されたチャネルで RDS for MySQL DB インスタンスからレプリケーションを開始し、指定されたグローバルトランザクション識別子 (GTID) でレプリケーションを停止します。

構文

```
CALL mysql.rds_start_replication_until_gtid_for_channel(gtid,channel_name);
```

パラメータ

gtid

その後でレプリケーションを停止する GTID。

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication_until_gtid_for_channel を実行する必要があります。このプロシージャは、指定されたチャネルでレプリケーションを開始し、指定された GTID 値までのすべての変更を適用します。その後、チャンネルのレプリケーションを停止します。

gtidパラメータで指定したトランザクションがレプリカによって既に実行されている場合、レプリケーションは即座に停止します。

この手順を実行する前に、replica_parallel_workers または slave_parallel_workers の値を 0 に設定して、マルチスレッドレプリケーションを無効にする必要があります。

例

次の例では、channel_1 でレプリケーションを開始し、GTID 3E11FA47-71CA-11E1-9E33-C80AA9429562:23 に達するまで変更をレプリケートします。

```
call mysql.rds_start_replication_until_gtid_for_channel('3E11FA47-71CA-11E1-9E33-
C80AA9429562:23','channel_1');
```

mysql.rds_stop_replication_for_channel

指定されたチャネルで MySQL DB インスタンスからのレプリケーションを停止します。

構文

```
CALL mysql.rds_stop_replication_for_channel(channel_name);
```

パラメータ

channel_name

マルチソースレプリカのレプリケーションチャネルの名前。各レプリケーションチャネルは、特定のホストとポートで実行されているシングルソース RDS for MySQL DB インスタンスからバイナリログイベントを受け取ります。

使用に関する注意事項

マスターユーザーが mysql.rds_stop_replication_for_channel を実行する必要があります。

例

次の例では、マルチソースレプリカの channel_1 でレプリケーションを停止します。

```
CALL mysql.rds_stop_replication_for_channel('channel_1');
```

GTID を使用したトランザクションのレプリケーション

以下のストアドプロシージャは、RDS for MySQL でグローバルトランザクション識別子 (GTID) を使用してトランザクションをレプリケートする方法を制御します。RDS for MySQL での GTID ベースのレプリケーションの詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

caching_sha2_password で設定されたレプリケーションユーザーによるレプリケーションを、ストアドプロシージャを使用して管理する場合は、SOURCE_SSL=1 を指定して TLS を設定する必要があります。caching_sha2_password は RDS for MySQL 8.4 のデフォルトの認証プラグインです。

トピック

- mysql.rds_skip_transaction_with_gtid
- mysql.rds_start_replication_until_gtid

mysql.rds_skip_transaction_with_gtid

MySQL DB インスタンスで、指定されたグローバルトランザクション識別子 (GTID) のあるトランザクションのレプリケーションをスキップします。

特定の GTID トランザクションが問題の原因となることが知られている場合、障害復旧のためにこのプロシージャを使用できます。このストアドプロシージャを使用して、問題となるトランザクションをスキップします。問題のあるトランザクションの例には、レプリケーションを無効にしたり、重要なデータを削除したり、DB インスタンスを利用不可にするトランザクションが含まれます。

構文

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

パラメータ

gtid_to_skip

スキップするレプリケーショントランザクションの GTID。

使用に関する注意事項

マスターユーザーが mysql.rds_skip_transaction_with_gtid を実行する必要があります。

この手順はすべての RDS for MySQL 5.7 バージョン、すべての RDS for MySQL 8.0 バージョン、すべての RDS for MySQL 8.4 バージョンでサポートされています。

例

次の例では、GTID 3E11FA47-71CA-11E1-9E33-C80AA9429562:23 を使用したトランザクションのレプリケーションをスキップします。

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid

RDS for MySQL DB インスタンスからのレプリケーションを開始し、指定したグローバルトランザクション識別子 (GTID) の後でレプリケーションを停止します。

構文

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

パラメータ

gtid

レプリケーションがその後で停止する GTID。

使用に関する注意事項

マスターユーザーが mysql.rds_start_replication_until_gtid を実行する必要があります。

この手順はすべての RDS for MySQL 5.7 バージョン、すべての RDS for MySQL 8.0 バージョン、すべての RDS for MySQL 8.4 バージョンでサポートされています。

このプロシージャは、災害対策用の遅延レプリケーションで使用できます。遅延レプリケーションを設定している場合は、このプロシージャを使用して、遅延したリードレプリカへの変更を災害発生

直前の時点までロールフォワードできます。このプロシージャでのレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

次のストアドプロシージャを使用して遅延レプリケーションを設定できます。

- mysql.rds_set_configuration
- mysql.rds_set_external_master_with_delay (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前)
- mysql.rds_set_external_source_with_delay (RDS for MySQL メジャーバージョン 8.4 以降)
- mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降)

gtidパラメータで指定したトランザクションがレプリカによって既に実行されている場合、レプリケーションは即座に停止します。

例

次の例では、レプリケーションをスタートし、GTID 3E11FA47-71CA-11E1-9E33-C80AA9429562:23 に達するまで変更をレプリケートします。

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

クエリログのローテーション

以下のストアドプロシージャは、MySQL ログをバックアップテーブルにローテーションします。詳細については、「 MySQL データベースのログファイル」を参照してください。

トピック

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

mysql.general_log テーブルをバックアップテーブルとしてローテーションさせます。

構文

```
CALL mysql.rds_rotate_general_log;
```

使用に関する注意事項

mysql.general_log テーブルのバックアップテーブルとしてのローテーションは、mysql.rds_rotate_general_log プロシージャを呼び出すことで実行できます。ログテーブルのローテーションが実行されると、現在のログテーブルがバックアップのログテーブルにコピーされ、現在のログテーブル内にあるエントリは削除されます。バックアップのログテーブルがすでに存在する場合は、現在のログテーブルをバックアップにコピーする前に、削除されます。バックアップのログテーブルは、必要に応じて照会することができます。mysql.general_log テーブルに対するバックアップのログテーブルは、mysql.general_log_backup という名前になります。

log_output パラメータが TABLE に設定されている場合にのみ、このプロシージャを実行できます。

mysql.rds_rotate_slow_log

mysql.slow_log テーブルをバックアップテーブルとしてローテーションさせます。

構文

```
CALL mysql.rds_rotate_slow_log;
```

使用に関する注意事項

mysql.slow_log テーブルのバックアップテーブルとしてのローテーション
は、mysql.rds_rotate_slow_log プロシージャを呼び出すことで実行できます。ログテーブル
のローテーションが実行されると、現在のログテーブルがバックアップのログテーブルにコピーさ
れ、現在のログテーブル内にあるエントリは削除されます。バックアップのログテーブルがすでに存
在する場合は、現在のログテーブルをバックアップにコピーする前に、削除されます。

バックアップのログテーブルは、必要に応じて照会することができます。mysql.slow_log テーブ
ルに対するバックアップのログテーブルは、mysql.slow_log_backup という名前になります。

バイナリログ構成の設定と表示

次のストアドプロシージャは、バイナリログファイルの保存などの設定パラメータを設定および表示します。

トピック

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

バイナリログを保持する時間数またはレプリケーションを遅延させる秒数を指定します。

構文

```
CALL mysql.rds_set_configuration(name,value);
```

パラメータ

.name

設定する設定パラメータの名前。

#

設定パラメータの値。

使用に関する注意事項

mysql.rds_set_configuration プロシージャは、以下の設定パラメータをサポートしています。

- バイナリログの保持時間
- ソース遅延
- ターゲット遅延

設定パラメータは永続的に保存され、DB インスタンスの再起動やフェイルオーバー後も存続します。

バイナリログの保持時間

binlog retention hours パラメータは、バイナリログファイルを保持する時間数を指定するために使用されます。Amazon RDS では、通常、バイナリログは可能な限りすみやかに消去されますが、RDS の外部にある MySQL データベースでのレプリケーションのためにバイナリログが必要になる場合があります。

binlog retention hours の初期値は NULL です。RDS for MySQL の場合、NULL はバイナリログが保持されないことを意味します (0 時間)。

DB インスタンスのバイナリログを保持する時間数を指定するには、mysql.rds_set_configuration ストアドプロシージャを使用して、次の例のように、レプリケーションを実行するのに十分な期間を指定します。

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> binlog retention hours には、値 0 は使用できません。

MySQL DB インスタンスの場合、binlog retention hours の最大値は 168 (7 日) です。

保持期間を設定したら、DB インスタンスのストレージ使用状況をモニタリングして、保持されたバイナリログに必要以上の容量が使用されないようにします。

マルチ AZ DB クラスターデプロイでは、ライター DB インスタンスからのみバイナリログ保持を設定でき、設定はすべてのリーダー DB インスタンスに非同期的に伝播されます。DB クラスターのバイナリログがローカルストレージ領域の合計の半分を超えると、Amazon RDS は自動的に古いログを EBS ボリュームに移動します。ただし、最新のログはローカルストレージに残るため、ホストの交換を必要とする障害が発生した場合、またはデータベースをスケールアップまたはスケールダウンした場合、ログは失われる可能性があります。

ソース遅延

リードレプリカで source delay パラメータを使用して、リードレプリカからソース DB インスタンスへのレプリケーションを遅延させる秒数を指定します。Amazon RDS は、通常、変更をできるだけ早くレプリケートしますが、環境によっては、レプリケーションを遅延させたい場合があります。例えば、レプリケーションを遅延させると、遅延させたリードレプリカを災害発生直前の時点ま

でロールフォワードできます。テーブルを誤って削除した場合は、遅延レプリケーションを使用して早急に復旧できます。target delay のデフォルト値は 0 です (レプリケーションを遅延させません)。

このパラメータを使用すると、mysql.rds_set_source_delay (RDS for MySQL メジャーバージョン 8.4 以降) を実行して、CHANGE primary TO MASTER_DELAY = 入力値が適用されます。成功すると、プロシージャは source delay パラメータを mysql.rds_configuration テーブルに保存します。

Amazon RDS でソース DB インスタンスへのレプリケーションを遅延させる秒数を指定するには、mysql.rds_set_configuration ストアドプロシージャを使用して、レプリケーションを遅延させる秒数を指定します。次の例では、レプリケーションは少なくとも 1 時間 (3600 秒) 遅延されます。

```
call mysql.rds_set_configuration('source delay', 3600);
```

その後、プロシージャは mysql.rds_set_source_delay(3600) を実行します。

source delay パラメータの上限は 1 日 (86400 秒) です。

ターゲット遅延

target delay パラメータを使用して、DB インスタンスと、このインスタンスから将来作成される RDS 管理リードレプリカとのレプリケーションを遅延させる秒数を指定します。このパラメータは、RDS で管理されていないリードレプリカでは無視されます。Amazon RDS は、通常、変更をできるだけ早くレプリケートしますが、環境によっては、レプリケーションを遅延させたい場合があります。例えば、レプリケーションを遅延させると、遅延させたリードレプリカを災害発生直前の時点までロールフォワードできます。テーブルを誤って削除した場合は、遅延レプリケーションを使用して早急に復旧できます。target delay のデフォルト値は 0 です (レプリケーションを遅延させません)。

障害復旧のために、この設定パラメータを mysql.rds_start_replication_until ストアドプロシージャ、または、mysql.rds_start_replication_until_gtid ストアドプロシージャで使用できます。このパラメータを設定した mysql.rds_set_configuration プロシージャを実行して、遅延させたリードレプリカへの変更を災害発生直前の時点までロールフォワードできます。mysql.rds_start_replication_until、または mysql.rds_start_replication_until_gtid プロシージャによりレプリケーションが停止したら、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」の手順に従ってリードレプリカを新しいプライマリ DB インスタンスに昇格させることができます。

mysql.rds_rds_start_replication_until_gtid プロシージャを使用するためには、GTID ベースのレプリケーションを有効にする必要があります。障害の原因となることが知られている特定の GTID ベースの処理をスキップするために、mysql.rds_skip_transaction_with_gtid ストアドプロシージャを使用できます。GTID ベースのレプリケーションの使用に関する詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

Amazon RDS でリードレプリカへのレプリケーションを遅延させる秒数を指定するには、mysql.rds_set_configuration ストアドプロシージャを使用し、レプリケーションを遅延させる秒数を指定します。次の例では、レプリケーションを少なくとも 1 時間 (3600 秒) 遅延させることを指定します。

```
call mysql.rds_set_configuration('target delay', 3600);
```

target delay パラメータの上限は 1 日 (86400 秒) です。

mysql.rds_show_configuration

バイナリログを保持する時間数。

構文

```
CALL mysql.rds_show_configuration;
```

使用に関する注意事項

Amazon RDS がバイナリログを保持する時間数を確認するには、mysql.rds_show_configuration ストアドプロシージャを使用します。

例

以下の例では、保持期間を表示しています。

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

InnoDB キャッシュのウォームアップ

以下のストアドプロシージャは、RDS for MySQL DB インスタンスの InnoDB バッファープールを保存、ロード、またはロードをキャンセルします。詳細については、「Amazon RDS の MySQL に対する InnoDB キャッシュウォーミング」を参照してください。

トピック

- mysql.rds_innodb_buffer_pool_dump_now
- mysql.rds_innodb_buffer_pool_load_abort
- mysql.rds_innodb_buffer_pool_load_now

mysql.rds_innodb_buffer_pool_dump_now

バッファープールの現在の状態をディスクにダンプします。

構文

```
CALL mysql.rds_innodb_buffer_pool_dump_now();
```

使用に関する注意事項

マスターユーザーが `mysql.rds_innodb_buffer_pool_dump_now` を実行する必要があります。

mysql.rds_innodb_buffer_pool_load_abort

保存したバッファープールの状態のロードを途中でキャンセルします。

構文

```
CALL mysql.rds_innodb_buffer_pool_load_abort();
```

使用に関する注意事項

マスターユーザーが `mysql.rds_innodb_buffer_pool_load_abort` を実行する必要があります。

mysql.rds_innodb_buffer_pool_load_now

保存したバッファープールの状態をディスクからロードします。

構文

```
CALL mysql.rds_innodb_buffer_pool_load_now();
```

使用に関する注意事項

マスターユーザーが mysql.rds_innodb_buffer_pool_load_now を実行する必要があります。

Amazon RDS for Oracle

Amazon RDS は、以下のバージョンおよびエディションの Oracle データベースを実行する DB インスタンスをサポートしています。

- Oracle Database 26ai (26.0.0.0)
- Oracle Database 21c (21.0.0.0)
- Oracle Database 19c (19.0.0.0)

> **Note**
>
> Oracle Database 11g、Oracle Database 12c、および Oracle Database 18c は、Amazon RDS でサポートされなくなったレガシーバージョンです。

DB インスタンスを作成する前に、このガイドの「Amazon RDS 環境のセットアップ」セクションの手順を完了してください。マスターアカウントを使用して DB インスタンスを作成すると、アカウントには DBA 権限が付与されます。ただし、いくつかの制限があります。例えば、マスターユーザーは SYSDBA 権限を必要とするオペレーションを実行できず、Oracle が提供する特定のパッケージやテーブルへのアクセスが制限されます。詳細については、「RDS for Oracle のユーザーと権限」を参照してください。このアカウントは、追加のデータベースアカウントの作成などの管理タスクに使用します。SYS、SYSTEM、またはその他の Oracle 提供の管理アカウントを使用することはできません。SYS が所有するオブジェクトに権限を付与するには、`rdsadmin.rdsadmin_util.grant_sys_object` プロシージャを使用します。詳細については、「SYS オブジェクトに対する権限を管理する方法」を参照してください。

以下を作成することができます。

- DB インスタンス
- DB スナップショット
- ポイントインタイムの復元
- 自動バックアップ
- 手動バックアップ

VPC 内で Oracle Database を実行している DB インスタンスを使用できます。また、Oracle Spatial や Oracle Statspack などのさまざまなオプションを有効にして、DB インスタンスに機能を追加することもできます。オプションを使用するには、オプショングループを作成してオプションを追加し、そのオプショングループを DB インスタンスに関連付けます。詳細については、「Oracle DB インスタンスへのオプションの追加」を参照してください。Amazon RDS は、可用性の高いフェイルオーバーソリューションとして Oracle のマルチ AZ 配置をサポートしています。

> **Important**
>
> マネージドサービスエクスペリエンスを提供するために、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする、特定のシステムプロシージャやテーブルへのアクセスも制限しています。データベースへのアクセスには、Oracle SQL *Plus などの標準的な SQL クライアントアプリケーションを使用します。ただし、Telnet またはセキュアシェル (SSH) を使用してホストに直接アクセスすることはできません。

トピック

- Amazon RDS での Oracle の概要
- Oracle DB インスタンスへの接続
- Oracle DB インスタンス接続の保護
- RDS for Oracle で CDB を使用する
- RDS for Oracle DB インスタンスの管理
- RDS for Oracle でストレージを使用する
- RDS for Oracle の高度な機能の設定
- Oracle の AI 機能
- Amazon RDS の Oracle にデータをインポートする
- Amazon RDS for Oracle でのリードレプリカの使用
- Oracle DB インスタンスへのオプションの追加
- RDS for Oracle DB エンジンのアップグレード
- RDS for Oracle DB インスタンスでのサードパーティーソフトウェアの使用
- Oracle データベースエンジンのリリースノート

Amazon RDS での Oracle の概要

次のセクションを読むと、RDS for Oracle の概要を確認できます。

トピック

- RDS for Oracle の機能
- RDS for Oracle のリリース
- RDS for Oracle のライセンスオプション
- RDS for Oracle のユーザーと権限
- RDS for Oracle DB インスタンスクラス
- Oracle データベースアーキテクチャの RDS
- RDS for Oracle 初期化パラメータ
- RDS for Oracle 文字セット
- RDS for Oracle の制限事項

RDS for Oracle の機能

Amazon RDS for Oracle は、Oracle Database のほとんどの機能をサポートしています。一部の機能には、制限付きのサポートまたは制限された特権があります。機能によっては、Enterprise Edition でのみ使用可能なものや、追加のライセンスが必要なものがあります。特定の Oracle データベースバージョンの Oracle データベース機能の詳細については、使用しているバージョンの「Oracle データベースのライセンス情報のユーザーマニュアル」を参照してください。

トピック

- RDS for Oracle の新機能
- RDS for Oracle でサポートされる機能
- RDS for Oracle でサポートされていない機能

RDS for Oracle の新機能

RDS for Oracle の新機能を表示するには、ドキュメント履歴 をキーワード **Oracle** で検索します。

RDS for Oracle でサポートされる機能

Amazon RDS for Oracle では、次の Oracle データベース機能をサポートしています。

> **Note**
>
> 以下のリストは完全なものではありません。

- 高度な圧縮
- AI ベクトル検索 (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「AI Vector Search」を参照してください。
- Amazon Bedrock の統合 (Oracle Database 26ai 以降)

  詳細については、「Amazon Bedrock と RDS for Oracle の統合」を参照してください。
- Oracle Application Express (APEX)

  詳細については、「Oracle Application Express (APEX)」を参照してください。
- 自動メモリ管理
- SQL エラーの自動緩和 (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「SQL_ERROR_MITIGATION」を参照してください。
- 自動 UNDO 管理
- 自動ワークロードリポジトリ (AWR)

  詳細については、「自動ワークロードリポジトリ (AWR) を使用したパフォーマンスレポートの生成」を参照してください。
- 同じ AWS リージョンまたは複数の AWS リージョン間での、Maximum Performance の Active Data Guard (Enterprise Edition のみ。Oracle Active Data Guard オプションライセンスが必要)

  詳細については、「Amazon RDS for Oracle でのリードレプリカの使用」を参照してください。
- ブロックチェーンテーブル (Oracle Database 21c 以上)

  詳細については、Oracle Database のドキュメントの「ブロックチェーンテーブルの管理」を参照してください。
- 継続的なクエリ通知

  詳細については、Oracle ドキュメントの「連続問合せ通知 (CQN) の使用」を参照してください。
- データマスキングとサブセット

詳細については、AWS データベースブログの「Amazon RDS for Oracle でのデータマスキング」を参照してください。

- データの改訂
- Database In-Memory (Enterprise Edition のみ。Oracle Database In-Memory オプションライセンスが必要です)
- 分散クエリと分散トランザクション
- エディションベースの再定義

  詳細については、「DB インスタンスのデフォルトエディションの設定」を参照してください。
- EM Express

  詳細については、「Oracle Enterprise Manager」を参照してください。
- ファイングレイン監査
- フラッシュバックテーブル、フラッシュバッククエリ、フラッシュバックトランザクションクエリ
- アプリケーションの段階的なパスワードロールオーバー (Oracle Database 21c 以降)

  詳細については、Oracle Database のドキュメントの「アプリケーションの段階的なデータベースパスワードロールオーバーの管理」を参照してください。
- HugePages

  詳細については、「サポートされている RDS for Oracle インスタンスで HugePages をオンにする」を参照してください。
- インポート/エクスポート (レガシーと Data Pump) と SQL*Loader

  詳細については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。
- Java Virtual Machine (JVM)

  詳細については、「Oracle Java Virtual Machine」を参照してください。
- JavaScript (Oracle Database 21c 以上)

  詳細については、Oracle Database のドキュメントの「DBMS_MLE」を参照してください。
- JavaScript ストアドプロシージャ (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「JavaScript 開発者ガイド」を参照してください。

- JSON 二元性ビュー (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「JSON-Relational Duality」を参照してください。

- Label Security

  詳細については、「Oracle Label Security」を参照してください。

- Locator

  詳細については、「Oracle Locator」を参照してください。

- マテリアライズドビュー
- マルチテナント

  Oracle マルチテナントアーキテクチャは、Oracle Database 19c 以降のすべてのリリースでサポートされています。詳細については、「RDS for Oracle で CDB を使用する」を参照してください。

- ネットワーク暗号化

  詳細については、「Oracle ネイティブネットワーク暗号化」および「Oracle Secure Sockets Layer」を参照してください。

- パーティション
- PL/SQL から SQL へのトランスパイル (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「SQL_TRANSPILER」を参照してください。

- プロパティグラフビュー (Oracle Database 26ai 以降)

  詳細については、Oracle Database ドキュメントの「Oracle Property Graph」を参照してください。

- Real Application Testing

  完全なキャプチャおよびリプレイ機能を使用するには、Amazon Elastic File System (Amazon EFS) を使用して、Oracle Real Application Testing によって生成されたファイルにアクセスする必要があります。詳細については、「Amazon EFS の統合」とブログ記事「Use Oracle Real Application Testing features with Amazon RDS for Oracle」を参照してください。

- リアルタイム SQL 計画管理 (Oracle Database 26ai 以降)

  詳細については、「RDS for Oracle でのリアルタイム SQL 計画管理」を参照してください。

- スキーマ権限 (Oracle Database 26ai 以降)

詳細については、Oracle Database ドキュメントの「Managing Schema Privilege」を参照してください。

- アプリケーションレベルのシャーディング (ただし、Oracle シャーディング機能は除く)
- テーブルスペースの縮小 (Oracle Database 26ai 以降)

  詳細については、「RDS for Oracle でのテーブルスペースの縮小」を参照してください。
- Spatial と Graph

  詳細については、「Oracle Spatial」を参照してください。
- スタークエリの最適化
- ストリームと高度なキューイング
- サマリー管理 – マテリアライズドビュークエリリライト
- テキスト。Amazon RDS for Oracle はホストファイルシステムへのアクセスまたはアウトバウンドファイルの取得を提供しないため、`FILE` および `URL` データストアタイプはサポートされていません。
- トータルリコール
- Transparent Data Encryption (TDE)

  詳細については、「Oracle Transparent Data Encryption」を参照してください。

  > **Note**
  >
  > DB インスタンスで TDE を有効にすると、無効にすることはできません。

- 統合監査、混合モード

  詳細については、Oracle ドキュメントの「混合モード監査」を参照してください。
- XML DB (XML DB Protocol Server を使用しない)

  詳細については、「Oracle XML DB」を参照してください。
- 仮想プライベートデータベース

RDS for Oracle でサポートされていない機能

Amazon RDS for Oracle では、次の Oracle データベース機能はサポートしていません。

> **Note**
>
> 以下のリストは完全なものではありません。

- Automatic Storage Management (ASM)
- Database Vault
- フラッシュバックデータベース

> **Note**
>
> 代替ソリューションについては、AWS データベースブログ記事「Amazon RDS for Oracle の Oracle フラッシュバックデータベース機能の代替方法」を参照してください。

- FTP および SFTP
- パーティション分割されたハイブリッドテーブル
- メッセージングゲートウェイ
- Oracle Enterprise Manager Cloud Control Management Repository
- 優先トランザクション
- Real Application Clusters (Oracle RAC)
- Real Application Security (RAS)
- SQL ファイアウォール
- True Cache
- 統合監査、Pure モード
- Workspace Manager (WMSYS) のスキーマ

> **Warning**
>
> 一般に、Amazon RDS では、サポートされていない機能のスキーマを作成することはできません。ただし、SYSDBA 権限を必要とする Oracle 機能およびコンポーネントのスキーマを作成すると、データディクショナリが破損し、DB インスタンスの可用性に影響する可能性があります。Oracle DB インスタンスへのオプションの追加 で使用可能なサポートされている機能およびスキーマのみを使用します。

RDS for Oracle のリリース

RDS for Oracle は、複数の Oracle Database のリリースをサポートしています。

> **Note**
>
> リリースのアップグレードの詳細については、「RDS for Oracle DB エンジンのアップグレード」を参照してください。

トピック

- Oracle Database 26ai と Amazon RDS
- Oracle Database 21c と Amazon RDS
- Oracle Database 19c と Amazon RDS

Oracle Database 26ai と Amazon RDS

Amazon RDS は、Enterprise Edition でのみ Oracle Database 26ai をサポートしています。Oracle は Linux x86-64 用の Oracle Database 26ai を Enterprise Edition としてのみリリースしました。Standard Edition 2 (SE2) はこのプラットフォームの Oracle Database 26ai では使用できません。Oracle Database 26ai (26.0.0.0) には、多くの新機能と以前のバージョンからの更新が含まれています。Oracle Database 21c と同様に、Oracle Database 26ai はマルチテナントアーキテクチャのみをサポートしています。従来の非 CDB としてデータベースを作成することはできません。CDB と非 CDB の相違点の詳細については、「RDS for Oracle CDB の制限事項」を参照してください。

このセクションでは、Amazon RDS で Oracle Database 26ai (26.0.0.0) を使用する際の重要な機能や変更について説明します。変更の完全な一覧については、Oracle Database 26ai ドキュメントを参照してください。Oracle Database 26ai の各エディションでサポートされている機能の詳細なリストについては、Oracle ドキュメントの「Permitted features, options, and management packs by Oracle database offering」を参照してください。アップグレードの詳細については、「Oracle Database 26ai へのアップグレード」を参照してください。

Oracle Database 26ai (26.0.0.0) の Amazon RDS パラメータ変更

Oracle Database 26ai (26.0.0.0) では、いくつかの新しいパラメータが追加されています。さらに値の範囲とデフォルト値が更新されたパラメータがいくつかあります。

トピック

- 新しいパラメータ
- 互換パラメータの変更点
- 削除されたパラメータ

新しいパラメータ

次の表は、Oracle Database 26ai (26.0.0.0) の新しい Amazon RDS パラメータの一覧です。

名前	値の範囲	デフォルトの値	変更可能	説明
allow_legacy_reco_protocol	TRUE | FALSE	TRUE	N	分散トランザクションリカバリプロセス (RECO) がレガシーリカバリプロトコルを使用するか、アップグレードされたプロトコルを使用するかを指定します。
allow_weak_crypto	TRUE | FALSE	TRUE	N	非推奨または弱いアルゴリズムを DBMS_CRYPTO パッケージで使用できるかどうかを制御します。
blockchain_table_retention_threshold	0 ~ 365000	16	Y	TABLE RETENTION 権限なしでブロックチェーンまたはイミュータブルテーブルを作成する際に設定できる最大アイドル時間 (日数) を指定します。
calendar_fiscal_year_start	NONE | *DATE[, FORMAT]*	NONE	Y	SQL 会計関数で使用する会計年度の開始を指定します。
client_prefetch_rows	0 ~ 4294967295	0	Y	ネットワークラウンドトリップを減らすために Oracle クライアントドライバーでプリ

名前	値の範囲	デフォルトの値	変更可能	説明
				フェッチされる行数を指定します。
db_recovery_auto_rekey	ON | OFF	ON	Y	Data Guard スタンバインスタンスで自動テーブルスペースキー更新リカバリを有効または無効にします。
default_credential	*credential_name*	NONE	Y	セッションのデフォルトの認証情報を指定します。
error_message_details	ON | OFF | DISALLOWED	ON	Y	データベースでエラーが発生したデータ値の詳細を含む追加のスタックエラーメッセージを生成するかどうかを制御します。
group_by_position_enabled	TRUE | FALSE	FALSE	Y	GROUP BY 句の正の整数を選択リストの式の序数位置として扱うかどうかを制御します。
hybrid_read_only	TRUE | FALSE	FALSE	N	PDB のハイブリッド読み取り専用モードを有効または無効にして、ローカルユーザーが読み取り専用モードを参照している間に CDB の共通ユーザーがパッチを適用できるようにします。
json_behavior	ON_ERROR: ERROR | ON_ERROR: NULL	NONE	Y	SQL/JSON 関数と条件のセッションレベルでのデフォルトの動作を指定します。

名前	値の範囲	デフォルトの値	変更可能	説明
json_expression_check	ON | OFF	OFF	Y	JSON クエリステートメントのチェックを有効または無効にします。
lockfree_reservation	ON | OFF	ON	Y	ロックフリー予約を有効または無効にします。
max_columns	STANDARD | EXTENDED	STANDARD	Y	ワイドテーブルを有効または無効にします。有効にすると、最大列数の上限が 4096 に引き上げられます。
max_saga_duration	1 ~ 2147483647	86400	Y	Saga の最大期間 (秒単位) を指定します。
memoptimize_write_area_size	0 ~ 2199023255552	0	Y	SGA の大きなプール内の取り込みバッファのサイズを指定します。
memoptimize_writes	HINT | ON	HINT	Y	Memoptimized Rowstore テーブルに対して高速取り込みを自動的に使用するか、MEMOPTIMIZE_WRITE ヒントが指定されている場合にのみ使用するかを制御します。
memory_max_size	0、または 1610612736 ～21990232555 52	0	N	MEMORY_SIZE を設定できる最大値を指定します。
memory_size	0、または 1610612736 ～21990232555 52	0	N	インスタンス全体の使用可能なメモリ (SGA と PGA の組み合わせ) のサイズを指定します。

名前	値の範囲	デフォルトの値	変更可能	説明
mle_prog_languages	ALL | JAVASCRIPT | OFF	ALL	Y	多言語エンジン (MLE) を有効または無効にするか、特定の言語を選択的に有効にします。
paranoid_concurrency_mode	TRUE | FALSE	FALSE	N	厳密に耐久性の高いクエリデータの取得を有効または無効にします。
pdb_lockdown_error	RAISE | IGNORE	RAISE	Y	PDB ロックダウンエラーを無視または発生させるモードを制御します。
plsql_function_dynamic_stats	PREFERENCE | ON | OFF | CHOOSE	PREFERENCE	Y	セッションレベルで PL/SQL 関数の動的統計を生成するかどうかを指定します。
plsql_implicit_conversion_bool	TRUE | FALSE	FALSE	Y	数値、文字、および BOOLEAN データ型間の PL/SQL での暗黙的な変換を有効または無効にします。
priority_txns_high_wait_target	1 ~ 2147483647	2147483647	Y	HIGH の優先トランザクションがブロッカーをロールバックする前に行ロックを待機する最大秒数を指定します。
priority_txns_medium_wait_target	1 ~ 2147483647	2147483647	Y	MEDIUM の優先トランザクションがブロッカーをロールバックする前に行ロックを待機する最大秒数を指定します。

名前	値の範囲	デフォルトの値	変更可能	説明
priority_txns_mode	ROLLBACK | TRACK	ROLLBACK	Y	優先トランザクションのモードを指定します (有効にする場合は ROLLBACK、適用せずにモニタリングする場合は TRACK)。
read_only	TRUE | FALSE	FALSE	Y	ユーザーセッションの読み取り専用モードを有効または無効にします。
result_cache_auto_blocklist	ON | OFF	ON	Y	適応結果キャッシュオブジェクトの除外を有効または無効にします。
result_cache_integrity	ENFORCED | TRUSTED	TRUSTED	Y	データベースが決定論的コンストラクトの結果キャッシュの整合性を適用するかどうかを指定します。
run_addm_for_awr_report	ALL | IMPORT_ONLY | NONE	NONE	Y	AWR レポート内で ADDM を実行できる AWR スナップショットのタイプを指定します。
session_exit_on_package_state_error	TRUE | FALSE	FALSE	Y	有効にすると、セッションの PL/SQL パッケージの状態が無効になった場合に、ハードセッションを強制的に終了します。
soda_behavior	*behavior_string*	NONE	Y	SODA (Simple Oracle Document Access) オペレーションの動作を制御します。

名前	値の範囲	デフォルトの値	変更可能	説明
sql_error_mitigation	ON | OFF	ON	Y	SQL ステートメントの自動エラー緩和を有効または無効にします。
sql_history_enabled	TRUE | FALSE	FALSE	Y	各セッションでユーザーが発行したステートメントの SQL 履歴モニタリングを有効または無効にします。
sql_transpiler	ON | OFF	OFF	Y	SQL トランスパイラを有効または無効にします。これにより、PL/SQL 関数が SQL 式に自動的に変換されます。
standby_parse_limit_seconds	0 ~ 2147483647	300	Y	スタンバイ解析の時間制限 (秒単位) を指定します。
statement_redirect_service	NONE | *service_name*	NONE	Y	ステートメントのリダイレクトサービスを指定します。
tablespace_encryption_default_cipher_mode	CFB | XTS	XTS	Y	デフォルトのテーブルスペース暗号化ブロック暗号モードを指定します。
time_at_dbtimezone	OFF | USER_SQL | DATABASE	OFF	Y	現在の時刻を計算する際に DB タイムゾーンを使用します。
tracefile_content_classification	ENABLE | DISABLE | DEFAULT	DEFAULT	Y	トレースレコードのセキュリティラベルプレフィックスの出力を有効または無効にします。

名前	値の範囲	デフォルトの値	変更可能	説明
transaction_recovery	ENABLED | DISABLED	ENABLED	Y	PDB でトランザクション復旧を有効または無効にするかどうかを指定します。
txn_priority	HIGH | MEDIUM | LOW	HIGH	Y	セッション内のすべてのユーザートランザクションの優先度を指定します。
unified_audit_trail_exclude_columns	NONE | *column[, column] ...*	NONE	N	機密データを含む列が UNIFIED_AUDIT_TRAIL ビューに入力されるかどうかを制御します。
vector_index_neighbor_graph_reload	RESTART | OFF	RESTART	Y	HNSW の再ロードを有効にするかどうかを指定します。
vector_memory_size	0 ~ 2199023255552	0	Y	SGA のベクトルプールのサイズを指定します。
vector_query_capture	ON | OFF	ON	Y	クエリベクトルのキャプチャを有効または無効にします。
xml_client_side_decoding	TRUE | FALSE	TRUE	Y	XML クライアント側のデコードを有効または無効にします。

名前	値の範囲	デフォルトの値	変更可能	説明
xml_handling_of_invalid_chars	RAISE_ERROR | CONVERT_TO_QUESTION_MARK | CONVERT_TO_REFERENCE_CHAR | CONVERT_TO_NULL	RAISE_ERROR	Y	xmlelement 中に無効な文字を処理します。
xml_params	NONE | *parameter=value[, parameter=value] ...*	NONE	Y	XML の動作を変更するパラメータを指定します。

互換パラメータの変更点

compatible パラメータには、Amazon RDS の Oracle Database 26ai (26.0.0.0) 用の新しい最大値があります。次の表に、新しいデフォルト値を示します。

パラメータ名	Oracle Database 26ai (26.0.0.0) の最大値
compatible	23.6.0

削除されたパラメータ

以下のパラメータは、Oracle Database 26ai (26.0.0.0) で削除されました。

- allow_deprecated_rpcs
- audit_sys_operations
- log_archive_start

Oracle Database 21c と Amazon RDS

Amazon RDS は、Oracle Enterprise Edition と Oracle Standard Edition 2 が含まれる Oracle Database 21c をサポートしています。Oracle Database 21c (21.0.0.0) には、多くの新機能と以前のバージョンからの更新が含まれています。重要な変更点は、Oracle Database 21c がマルチテナントアーキテクチャのみをサポートすることで、従来の非 CDB としてデータベースを作成することはできなくなります。CDB と非 CDB の相違点の詳細については、「RDS for Oracle CDB の制限事項」を参照してください。

このセクションでは、Amazon RDS で Oracle Database 21c (21.0.0.0) を使用する際の重要な機能や変更について説明します。変更の詳細な一覧については、Oracle Database 21c ドキュメントを参照してください。Oracle Database 21c の各エディションでサポートされている機能の詳細なリストについては、Oracle ドキュメントの「Oracle Database 製品で許可される機能、オプションおよび Management Pack」を参照してください。

Oracle Database 21c (21.0.0.0) の Amazon RDS パラメータ変更

Oracle Database 21c (21.0.0.0) では、いくつかの新しいパラメータが追加されています。さらに値の範囲とデフォルト値が更新されたパラメータがいくつかあります。

トピック

- 新しいパラメータ
- 互換パラメータの変更点
- 削除されたパラメータ

新しいパラメータ

次の表は、Oracle Database 21c (21.0.0.0) の新しい Amazon RDS パラメータの一覧です。

名前	値の範囲	デフォルトの値	変更可能	説明
blockchain_table_max_no_drop	NONE | 0	NONE	Y	ブロックチェーンテーブルの作成時に指定できる最大アイドル時間を制御できます。
dbnest_enable	NONE | CDB_RESOU	NONE	N	dbNest を有効または無効にできます。dbNest は、オペ

名前	値の範囲	デフォルトの値	変更可能	説明
	RCE_PDB_ALL			レーティングシステムリソースの分離と管理、ファイルシステムの分離、およびセキュアなコンピューティングを PDB に提供します。
dbnest_pdb_fs_conf	NONE | *pathname*	NONE	N	PDB の dbNest ファイルシステム構成ファイルを指定します。
diagnostics_control	ERROR | WARNING | IGNORE	IGNORE	Y	安全でない可能性があるデータベース診断操作を実行するユーザーをコントロールおよびモニタリングできます。
drcp_dedicated_opt	YES | NO	YES	Y	データベース常駐接続プーリング (DRCP) での専用最適化の使用を有効または無効にします。
enable_per_pdb_drcp	true | false	true	N	データベース常駐接続プーリング (DRCP) で、CDB 全体に対して 1 つの接続プールを設定するか、PDB ごとに 1 つの分離接続プールを設定するかを制御します。
inmemory_deep_vectorization	true | false	true	Y	インメモリ列ストアのスキャンに対するディープベクトル化を有効または無効にします。
mandatory_user_profile	*profile_name*	該当なし	N	CDB または PDB の必須ユーザープロファイルを指定します。

名前	値の範囲	デフォルトの値	変更可能	説明
optimizer_capture_sql_quarantine	true | false	false	Y	SQL 隔離設定の自動作成を有効または無効にします。
optimizer_use_sql_quarantine	true | false	false	Y	SQL 隔離設定の使用を有効または無効にします。
result_cache_execution_threshold	0～68719476736	2	Y	結果が結果キャッシュに格納される前に、PL/SQL 関数を実行できる最大回数を指定します。
result_cache_max_temp_result	0～100	5	Y	キャッシュされた単一のクエリ結果が消費できる RESULT_CACHE_MAX_TEMP_SIZE の割合 (%) を指定します。
result_cache_max_temp_size	0～2199023255552	RESULT_CACHE_SIZE * 10	Y	結果キャッシュで消費できる一時テーブルスペースの最大量 (バイト単位) を指定します。
sga_min_size	0～2199023255552 (最大値は sga_target の 50%)	0	Y	プラグ可能なデータベース (PDB) の SGA の使用可能な最小値を示します。

名前	値の範囲	デフォルトの値	変更可能	説明
tablespace_encryption_default_algorithm	GOST256 | SEED128 | ARIA256 | ARIA192 | ARIA128 | 3DES168 | AES256 | AES192 | AES128	AES128	Y	テーブルスペースの暗号化時にデータベースが使用するデフォルトのアルゴリズムを指定します。

互換パラメータの変更点

compatible パラメータには、Amazon RDS の Oracle Database 21c (21.0.0.0) 用の新しい最大値があります。次の表に、新しいデフォルト値を示します。

パラメータ名	Oracle Database 21c (21.0.0.0) の最大値
compatible	21.0.0

削除されたパラメータ

以下のパラメータは、Oracle Database 21c (21.0.0.0) で削除されました。

- remote_os_authent
- sec_case_sensitive_logon
- unified_audit_sga_queue_size

Oracle Database 19c と Amazon RDS

Amazon RDS は、Oracle Enterprise Edition と Oracle Standard Edition Two が含まれる Oracle Database 19c をサポートしています。

Oracle Database 19c (19.0.0.0) には、多くの新機能と以前のバージョンからの更新が含まれています。このセクションでは、Amazon RDS で Oracle Database 19c (19.0.0.0) を使用する際の重要な機能や変更について説明します。変更の詳細な一覧については、Oracle Database 19c ドキュメントを参照してください。Oracle Database 19c の各エディションでサポートされている機能の詳細なリストについては、Oracle ドキュメントの「Oracle Database 製品で許可される機能、オプションおよび Management Pack」を参照してください。

Oracle Database 19c (19.0.0.0) の Amazon RDS パラメータ変更

Oracle Database 19c (19.0.0.0) では、いくつかの新しいパラメータが追加されています。さらに値の範囲とデフォルト値が更新されたパラメータがいくつかあります。

トピック

- 新しいパラメータ
- 互換パラメータの変更点
- 削除されたパラメータ

新しいパラメータ

次の表は、Oracle Database 19c (19.0.0.0) の新しい Amazon RDS パラメータの一覧です。

名前	値	変更可能	説明
lob_signature_enable	TRUE、FALSE (デフォルト)	Y	LOB ロケーター署名機能を有効または無効にします。
max_datapump_parallel_per_job	1 ~ 1024、または AUTO	Y	各 Oracle Data Pump ジョブに許可される並列プロセスの最大数を指定します。

互換パラメータの変更点

`compatible` パラメータには、Amazon RDS の Oracle Database 19c (19.0.0.0) 用の新しい最大値があります。次の表に、新しいデフォルト値を示します。

パラメータ名	Oracle Database 19c (19.0.0.0) の最大値
compatible	19.0.0

削除されたパラメータ

以下のパラメータは、Oracle Database 19c (19.0.0.0) で削除されました。

- `exafusion_enabled`
- `max_connections`
- `o7_dictionary_access`

RDS for Oracle のライセンスオプション

Amazon RDS for Oracle には、ライセンス込み (LI) と Bring-Your-Own-License (BYOL) の 2 つのライセンスオプションがあります。Amazon RDS で Oracle DB インスタンスを作成したら、DB インスタンスを変更してライセンスモデルを変更できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Important**
>
> DB インスタンスクラスと Oracle Database エディション用の適切な Oracle Database ライセンス (ソフトウェア更新ライセンスおよびサポート) を所持していることを確認してください。また、個別にライセンスされた Oracle Database 機能のライセンスがあることを確認してください。

トピック

- SE2 のライセンス込みモデル
- EE および SE2 の Bring Your Own License (BYOL)
- Oracle マルチ AZ 配置のライセンス

SE2 のライセンス込みモデル

License Included モデルでは、Oracle Database のライセンスを別途購入する必要はありません。Oracle Database ソフトウェアのライセンスは、AWS に含まれています。License Included モデルは、Amazon RDS for Oracle Database Standard Edition 2 (SE2) でのみサポートされます。

このモデルで、ケースサポートがある AWS サポート アカウントをお持ちの場合に、Amazon RDS と Oracle Database のサービスリクエストを行うには、サポート にお問い合わせください。RDS for Oracle の LI オプションの使用には、「AWS サービス条件」のセクション 10.3.1 が適用されます。

EE および SE2 の Bring Your Own License (BYOL)

BYOL モデルでは、既存の Oracle Database のライセンスを使用して Amazon RDS でデータベースをデプロイできます。Amazon RDS は、Oracle Database Enterprise Edition (EE) および Oracle Database Standard Edition 2 (SE2) の BYOL モデルのみをサポートしています。

実行する DB インスタンスクラスと Oracle Database エディション用の適切な Oracle Database ライセンス (ソフトウェア更新ライセンスおよびサポート) を所持している必要があります。また、クラウドコンピューティング環境での Oracle Database ソフトウェアのライセンス化に関する Oracle のポリシーに従う必要があります。Amazon EC2 用の Oracle のライセンスポリシーの詳細については、「クラウドコンピューティング環境での Oracle ソフトウェアのライセンス」を参照してください。

このモデルでは、アクティブな Oracle サポートアカウントを継続して使用できます。Oracle データベースのサービスリクエストについては、直接 Oracle にお問い合わせください。ケースサポート付きの AWS サポート アカウントをお持ちであれば、Amazon RDS の問題については サポート にお問い合わせいただけます。

AWS License Manager との統合

BYOL モデルで Oracle ライセンスの使用状況をモニタリングしやすくするために、AWS License Manager が Amazon RDS for Oracle と統合されています。License Manager は、仮想コア (vCPUs) に基づく RDS for Oracle エンジンのエディションとライセンスパックの追跡をサポートしています。また AWS Organizations で License Manager を使用して、すべての組織アカウントを一元管理することもできます。

次の表に、RDS for Oracle の製品情報フィルターを示します。

フィルタ	名前	説明
エンジンのエディション	`oracle-ee`	Oracle Database Enterprise Edition (EE)
	`oracle-se2`	Oracle Database Standard Edition 2 (SE2)
ライセンスパック	`data guard`	「Amazon RDS for Oracle でのリードレプリカの使用 (Oracle Active Data Guard)」を参照してください。
	`olap`	「Oracle OLAP」を参照してください。
	`ols`	「Oracle Label Security」を参照してください。
	`diagnostic pack sqlt`	「Oracle SQLT」を参照してください。
	`tuning pack sqlt`	「Oracle SQLT」を参照してください。

Oracle DB インスタンスのライセンス使用状況を追跡するには、AWS License Manager を使用してセルフマネージドライセンスを作成できます。この場合、製品情報フィルターに一致する Oracle リソースは、セルフマネージドライセンスに自動的に関連付けられます。Oracle DB インスタンスの検出には最長で 24 時間かかることがあります。AWS Resource Access Manager を使用して、アカウント間でライセンスを追跡することもできます。

コンソール

Oracle DB インスタンスのライセンス使用状況を追跡するセルフマネージドライセンスを AWS License Manager で作成するには

1. https://console.aws.amazon.com/license-manager/ に移動します。
2. [セルフマネージドライセンスを作成] を選択します。

   手順については、「AWS License Manager ユーザーガイド」の「セルフマネージドライセンスを作成する」を参照してください。

   [Product Information (製品情報)] パネルで [RDS Product Information Filter (RDS 製品情報フィルター)] のルールを追加します。

詳細については、AWS License Manager API リファレンスの「ProductInformation」を参照してください。

3. (クロスアカウント追跡のみ) セルフマネージドライセンスを任意の AWS アカウントと共有する場合、または AWS Organizations を通じて共有する場合に AWS Resource Access Manager を使用します。詳細については、「AWS リソースの共有」を参照してください。

AWS CLI

AWS CLI を使用してセルフマネージドライセンスを作成するには、create-license-configuration コマンドを呼び出します。`--cli-input-json` または `--cli-input-yaml` パラメータを使用して、コマンドにパラメータを渡すことができます。

Example

次の例は、Oracle Enterprise Edition のセルフマネージドライセンスを作成します。

```
aws license-manager create-license-configuration --cli-input-json file://rds-oracle-
ee.json
```

次に、この例で使用されているサンプルの `rds-oracle-ee.json` ファイルを示します。

```
{
    "Name": "rds-oracle-ee",
    "Description": "RDS Oracle Enterprise Edition",
    "LicenseCountingType": "vCPU",
    "LicenseCountHardLimit": false,
    "ProductInformationList": [
        {
            "ResourceType": "RDS",
            "ProductInformationFilterList": [
                {
                    "ProductInformationFilterName": "Engine Edition",
                    "ProductInformationFilterValue": ["oracle-ee"],
                    "ProductInformationFilterComparator": "EQUALS"
                }
            ]
        }
    ]
}
```

製品情報の詳細については、AWS License Manager ユーザーガイドの「リソースインベントリの自動検出」を参照してください。

`--cli-input` パラメータの詳細については、AWS CLI ユーザーガイドの「JSON または YAML 入力ファイルからの AWS CLI スケルトンと入力パラメータの生成」を参照してください。

Oracle Database のエディション間での移行

実行する予定の DB インスタンスのエディションとクラス向けに適切な未使用の BYOL Oracle Database ライセンスを保有している場合、任意の Standard Edition 2 (SE2) から Enterprise Edition (EE) に移行できます。ただし、EE から別のエディションには移行できません。

Oracle Database のエディションを変更してデータを保持するには

1. DB インスタンスのスナップショットを作成します。

   詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
2. 新しい DB インスタンスにスナップショットを復元して、使用する Oracle データベースエディションを選択します。

   詳細については、「DB インスタンスへの復元」を参照してください。
3. (オプション) エディションに適切な Oracle データベースのライセンスを持っていて、それを実行し続けることを希望する場合を除き、古い DB インスタンスを削除します。

   詳細については、「DB インスタンスを削除する」を参照してください。

Oracle マルチ AZ 配置のライセンス

Amazon RDS は、可用性の高いフェイルオーバーソリューションとして Oracle のマルチ AZ 配置をサポートしています。本稼働のワークロードではマルチ AZ をお勧めします。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

Bring-Your-Own-License モデルを使用する場合、マルチ AZ 配置でプライマリ DB インスタンスとスタンドバイ DB インスタンスの両方でライセンスを保持していることが必要です。

RDS for Oracle のユーザーと権限

Amazon RDS for Oracle DB インスタンスを作成すると、デフォルトのマスターユーザーは DB インスタンスに対する最大のユーザーアクセス許可のほとんどを持ちます。データベースでの追加

のユーザーアカウントの作成などの管理タスクには、このマスターユーザーアカウントを使用します。RDS はマネージドサービスであるため、SYS および SYSTEM として ログインすることはできず、したがって、SYSDBA 権限を持ちません。

トピック

- Oracle DBA 権限の制限事項
- SYS オブジェクトに対する権限を管理する方法

Oracle DBA 権限の制限事項

ロールとは、ユーザーに対して付与または取り消すことができる権限のコレクションです。Oracle データベースは、セキュリティを提供するために ロールを使用します。詳細については、Oracle Database ドキュメントの「権限と ロール承認の設定」を参照してください。

事前に定義された ロール DBA は、通常、Oracle データベースに対するすべての管理権限を付与します。マスターユーザーアカウントを使用して DB インスタンスを作成すると、アカウントには DBA 権限が付与されます。ただし、いくつかの制限があります。マネージドエクスペリエンスを提供するために、RDS for Oracle データベースは DBA ロールに次の権限を提供しません。

- `ALTER DATABASE`
- `ALTER SYSTEM`
- `CREATE ANY DIRECTORY`
- `DROP ANY DIRECTORY`
- `GRANT ANY PRIVILEGE`
- `GRANT ANY ROLE`

RDS for Oracle のシステム権限と ロールの詳細については、「マスターユーザーアカウント権限」を参照してください。

SYS オブジェクトに対する権限を管理する方法

SYS オブジェクトに対する権限は、`rdsadmin.rdsadmin_util` パッケージを使用して管理できます。例えば、データベースユーザー `myuser` を作成する場合、`rdsadmin.rdsadmin_util.grant_sys_object` プロシージャを使用して、`V_$SQLAREA` に対する `SELECT` 権限を `myuser` に付与できます。詳細については、以下の各トピックを参照してください。

- SYS オブジェクトへの SELECT または EXECUTE 権限の付与
- SYS オブジェクトに対する SELECT または EXECUTE 権限の取り消し
- 非マスターユーザーへの権限の付与

RDS for Oracle DB インスタンスクラス

RDS for Oracle DB インスタンスの計算とメモリの容量は、インスタンスクラスによって決まります。必要な DB インスタンスクラスは、処理能力とメモリの要件によって異なります。

サポートされている RDS for Oracle DB インスタンスクラス

サポートされる RDS for Oracle インスタンスクラスは、RDS DB インスタンスクラスのサブセットです。RDS インスタンスクラスの完全なリストについては、「DB インスタンスクラス」を参照してください。

RDS for Oracle 事前設定 DB インスタンスクラス

RDS for Oracle では、追加のメモリ、ストレージ、および vCPU あたりの I/O 数を必要とするワークロード向けに事前設定されたインスタンスクラスも提供しています。これらのインスタンスクラスでは、以下の命名規則が使用されます。

```
db.r5b.instance_size.tpcthreads_per_core.memratio
db.r5.instance_size.tpcthreads_per_core.memratio
```

以下は、追加のメモリが事前設定されたインスタンスクラスの例です。

```
db.r5b.4xlarge.tpc2.mem2x
```

前述のインスタンスクラス名の構成要素は次のとおりです。

- `db.r5b.4xlarge` – インスタンスクラスの名前。
- `tpc2` – コアごとのスレッド。値が 2 である場合、マルチスレッドがオンであることを意味します。1 の値は、マルチスレッドがオフであることを意味します。
- `mem2x` – そのインスタンスクラスの標準メモリに対する追加メモリの比率。この例では、最適化によって標準の db.r5.4xlarge DB インスタンスの 2 倍のメモリが提供されています。

> **Note**
>
> 事前設定された RDS for Oracle DB インスタンスクラスの正規化係数については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

RDS for Oracle でサポートされているエディション、インスタンスクラス、ライセンスの組み合わせ

RDS コンソールを使用している場合は、[データベースの作成] を選択し、別のオプションを指定することで、特定のエディション、インスタンスクラス、ライセンスの組み合わせがサポートされているかどうかを確認できます。AWS CLI で、以下のコマンドを実行できます。

```
aws rds describe-orderable-db-instance-options --engine engine-type --license-
model license-type
```

次の表は、RDS for Oracle でサポートされているすべてのエディション、インスタンスクラス、ライセンスタイプの一覧です。メモリ属性の各タイプの詳細については、RDS for Oracle インスタンスタイプを参照してください。料金の詳細については、「Amazon RDS for Oracle の料金」を参照してください。

Oracle エディション	Oracle Database 19c 以降
Enterprise Edition (EE)	スタンダードインスタンスクラス
Bring-Your-Own-License (BYOL)	db.m8i.large–db.m8i.96xlarge、db.m8i.metal-48xl、db.m8i.metal-96xl db.m7i.large–db.m7i.48xlarge、db.m7i.metal-24xl、db.m7i.metal-48xl db.m6in.large–db.m6in.32xlarge、db.m6in.metal db.m6id.large–db.m6id.32xlarge、db.m6id.metal db.m6i.large–db.m6i.32xlarge、db.m6i.metal db.m5d.large–db.m5d.24xlarge db.m5.large-db.m5.24xlarge

Oracle エディション	Oracle Database 19c 以降
	メモリ最適化インスタンスクラス
	db.r8i.large–db.r8i.96xlarge、db.r8i.metal-48xl、db.r8i.metal-96xl
	db.r7i.large–db.r7i.48xlarge、db.r7i.metal-24xl、db.r7i.metal-48xl
	db.r6in.large–db.r6in.32xlarge、db.r6in.metal
	db.r6id.large–db.r6id.32xlarge、db.r6id.metal
	db.r6i.large–db.r6i.32xlarge、db.r6i.metal
	db.r5d.large–db.r5d.24xlarge
	db.r5b.large–db.r5b.24xlarge
	db.r5.large-db.r5.24xlarge
	db.x2iedn.xlarge–db.x2iedn.32xlarge、db.x2iedn.metal
	db.x2iezn.2xlarge–db.x2iezn.12xlarge、db.x2iezn.metal
	db.x2idn.16xlarge–db.x2idn.32xlarge、db.x2idn.metal
	db.z1d.large–db.z1d.12xlarge
	メモリ最適化の事前設定済みインスタンスクラス

Oracle エディション	Oracle Database 19c 以降
	db.r7i.8xlarge.tpc2.mem3x
	db.r7i.8xlarge.tpc2.mem2x
	db.r7i.6xlarge.tpc2.mem4x
	db.r7i.6xlarge.tpc2.mem2x
	db.r7i.4xlarge.tpc2.mem4x
	db.r7i.4xlarge.tpc2.mem3x
	db.r7i.4xlarge.tpc2.mem2x
	db.r7i.3xlarge.tpc2.mem4x
	db.r7i.2xlarge.tpc2.mem8x
	db.r7i.2xlarge.tpc2.mem4x
	db.r7i.xlarge.tpc2.mem4x
	db.r7i.xlarge.tpc2.mem2x
	db.r6i.8xlarge.tpc2.mem4x
	db.r6i.8xlarge.tpc2.mem3x
	db.r6i.6xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem3x
	db.r6i.4xlarge.tpc2.mem2x
	db.r6i.2xlarge.tpc2.mem8x
	db.r6i.2xlarge.tpc2.mem4x
	db.r6i.2xlarge.tpc1.mem2x

Oracle エディション	Oracle Database 19c 以降
	db.r6i.xlarge.tpc2.mem4x
	db.r6i.xlarge.tpc2.mem2x
	db.r6i.large.tpc1.mem2x
	db.r5b.8xlarge.tpc2.mem3x
	db.r5b.6xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem3x
	db.r5b.4xlarge.tpc2.mem2x
	db.r5b.2xlarge.tpc2.mem8x
	db.r5b.2xlarge.tpc2.mem4x
	db.r5b.2xlarge.tpc1.mem2x
	db.r5b.xlarge.tpc2.mem4x
	db.r5b.xlarge.tpc2.mem2x
	db.r5b.large.tpc1.mem2x
	db.r5.12xlarge.tpc2.mem2x
	db.r5.8xlarge.tpc2.mem3x
	db.r5.6xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem3x
	db.r5.4xlarge.tpc2.mem2x
	db.r5.2xlarge.tpc2.mem8x

Oracle エディション	Oracle Database 19c 以降
	db.r5.2xlarge.tpc2.mem4x
	db.r5.2xlarge.tpc1.mem2x
	db.r5.xlarge.tpc2.mem4x
	db.r5.xlarge.tpc2.mem2x
	db.r5.large.tpc1.mem2x
	バーストパフォーマンスインスタンスクラス
	db.t3.small–db.t3.2xlarge
Standard Edition 2 (SE2)	スタンダードインスタンスクラス
Bring-Your-Own-License (BYOL)	db.m8i.large–db.m8i.4xlarge、db.m8i.metal-48xl、db.m8i.metal-96xl
	db.m7i.large–db.m7i.4xlarge、db.m7i.metal-24xl、db.m7i.metal-48xl
	db.m6in.large–db.m6in.4xlarge、db.m6in.metal
	db.m6id.large–db.m6id.4xlarge、db.m6id.metal
	db.m6i.large–db.m6i.4xlarge、db.m6i.metal
	db.m5d.large–db.m5d.4xlarge
	db.m5.large-db.m5.4xlarge
	メモリ最適化インスタンスクラス

Oracle エディション	Oracle Database 19c 以降
	db.r8i.large–db.r8i.4xlarge、db.r8i.metal-48xl、db.r8i.metal-96xl
	db.r7i.large–db.r7i.4xlarge、db.r7i.metal-24xl、db.r7i.metal-48xl
	db.r6in.large–db.r6in.4xlarge、db.r6in.metal
	db.r6id.large–db.r6id.4xlarge、db.r6id.metal
	db.r6i.large–db.r6i.4xlarge、db.r6i.metal
	db.r5d.large–db.r5d.4xlarge
	db.r5b.large–db.r5b.4xlarge
	db.r5.large-db.r5.4xlarge
	db.x2iedn.xlarge–db.x2iedn.4xlarge、db.x2iedn.metal
	db.x2iezn.2xlarge–db.x2iezn.4xlarge、db.x2iezn.metal
	db.x2idn.metal
	db.z1d.large–db.z1d.3xlarge
	メモリ最適化の事前設定済みインスタンスクラス

Oracle エディション	Oracle Database 19c 以降
	db.r7i.xlarge.tpc2.mem2x
	db.r7i.xlarge.tpc2.mem4x
	db.r7i.2xlarge.tpc2.mem4x
	db.r7i.2xlarge.tpc2.mem8x
	db.r7i.3xlarge.tpc2.mem4x
	db.r7i.4xlarge.tpc2.mem2x
	db.r7i.4xlarge.tpc2.mem3x
	db.r7i.4xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem4x
	db.r6i.4xlarge.tpc2.mem3x
	db.r6i.4xlarge.tpc2.mem2x
	db.r6i.2xlarge.tpc2.mem8x
	db.r6i.2xlarge.tpc2.mem4x
	db.r6i.2xlarge.tpc1.mem2x
	db.r6i.xlarge.tpc2.mem4x
	db.r6i.xlarge.tpc2.mem2x
	db.r6i.large.tpc1.mem2x
	db.r5b.4xlarge.tpc2.mem4x
	db.r5b.4xlarge.tpc2.mem3x
	db.r5b.4xlarge.tpc2.mem2x
	db.r5b.2xlarge.tpc2.mem8x

Oracle エディション	Oracle Database 19c 以降
	db.r5b.2xlarge.tpc2.mem4x
	db.r5b.2xlarge.tpc1.mem2x
	db.r5b.xlarge.tpc2.mem4x
	db.r5b.xlarge.tpc2.mem2x
	db.r5b.large.tpc1.mem2x
	db.r5.4xlarge.tpc2.mem4x
	db.r5.4xlarge.tpc2.mem3x
	db.r5.4xlarge.tpc2.mem2x
	db.r5.2xlarge.tpc2.mem8x
	db.r5.2xlarge.tpc2.mem4x
	db.r5.2xlarge.tpc1.mem2x
	db.r5.xlarge.tpc2.mem4x
	db.r5.xlarge.tpc2.mem2x
	db.r5.large.tpc1.mem2x
	バーストパフォーマンスインスタンスクラス
	db.t3.small–db.t3.2xlarge
Standard Edition 2 (SE2)	スタンダードインスタンスクラス
ライセンス込み	db.m8i.large–db.m8i.4xlarge
	db.m7i.large–db.m7i.4xlarge
	db.m5.large-db.m5.4xlarge
	メモリ最適化インスタンスクラス

Oracle エディション	Oracle Database 19c 以降
	db.r8i.large–db.r8i.4xlarge
	db.r7i.large–db.r7i.4xlarge
	db.r6i.large–db.r6i.4xlarge
	db.r5.large-db.r5.4xlarge
	バーストパフォーマンスインスタンスクラス
	db.t3.small–db.t3.2xlarge

非推奨の RDS for Oracle DB インスタンスクラス

以下 のDB インスタンスクラスは、RDS for Oracle で非推奨となっています。

- db.m1、db.m2、db.m3、db.m4
- db.t1、db.t2
- db.r1、db.r2、db.r3、db.r4
- db.x1、db.x1e

前出の DB インスタンスは、パフォーマンスに優り、一般に低いコストで利用可能な DB インスタンスと置き換えられています。非推奨となった DB インスタンスクラスを使用する DB インスタンスがある場合、以下のオプションがあります。

- Amazon RDS が、非推奨になっていない同等の DB インスタンスクラスを使用するように、各 DB インスタンスを自動的に変更できるようにします。非推奨のタイムラインについては、「DB インスタンスクラスタイプ」を参照してください。
- DB インスタンスを変更することで、DB インスタンスクラスを変更できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> 非推奨の DB インスタンスクラスを使用していた DB インスタンスの DB スナップショットがある場合、DB スナップショットを復元する際に、非推奨ではない DB インスタンスクラスを選択できます。詳細については、「DB インスタンスへの復元」を参照してください。

Oracle データベースアーキテクチャの RDS

Oracle マルチテナントアーキテクチャ (CDB アーキテクチャ) により、Oracle データベースをマルチテナントコンテナデータベース (CDB) として機能させることができます。CDB には、お客様が作成したプラグ可能なデータベース (PDB) を含めることができます。非 CDB は、PDB を含めることができない従来のアーキテクチャを使用する Oracle データベースです。マルチテナントアーキテクチャの詳細は、Oracle Multitenant Administrator's Guide を参照してください。

Oracle Database 19c 以降の場合、CDB アーキテクチャを使用する RDS for Oracle DB インスタンスを作成できます。RDS for Oracle では、PDB はテナントデータベースと呼ばれます。クライアントアプリケーションは CDB レベルではなくテナントデータベース (PDB) レベルで接続します。Amazon RDS for Oracle は CDB アーキテクチャの次の設定をサポートしています。

マルチテナント設定

Amazon RDS では、データベースエディションと必要なオプションライセンスに応じて、1 ~ 30 個のテナントデータベースを CDB インスタンスに含めることができます。RDS API を使用して、テナントデータベースを追加、変更、削除できます。RDS for Oracle のマルチテナント設定は、特殊なタイプの PDB であるアプリケーション PDB またはプロキシ PDB をサポートしていません。アプリケーション PDB とプロキシ PDB の詳細については、Oracle Databaseドキュメントの「PDBの種類」を参照してください。

> **Note**
>
> Amazon RDS 設定は、単なる Oracle DB エンジンではなく、Amazon RDS の機能であるため、「multitenant」ではなく「multi-tenant」と呼ばれています。同様に、RDS の用語「テナント」は、Oracle PDB だけでなく、RDS 設定のすべてのテナントを指します。RDS のドキュメントの「Oracle マルチテナント」という用語は、オンプレミスデプロイと RDS デプロイの両方に対応する Oracle データベース CDB アーキテクチャのみを指します。

シングルテナント設定

Amazon RDS は、RDS for Oracle CDB インスタンスを 1 個のテナントデータベース (PDB) に制限します。RDS API を使用して PDB をこれ以上追加することはできません。シングルテナント設定では、非 CDB アーキテクチャと同じ RDS API を使用します。したがって、シングルテナント設定での CDB の使用経験は、非 CDB での作業とほとんど同じです。

シングルテナント設定を使用する CDB をマルチテナント設定に変換して、PDB を CDB に追加できます。このアーキテクチャの変更は永続的で、元に戻すことはできません。詳細については、「シングルテナント設定からマルチテナント設定への変換」を参照してください。

> **Note**
>
> CDB 自体にはアクセスできません。

> **Note**
>
> 設定の選択に役立つ方法: Oracle Database 19c を実行していて、マルチテナント機能を必要としない場合は、非 CDB を選択します。Oracle Database 21c 以降 (CDB が必要) への今後の移行に備える場合は、シングルテナント CDB を選択します。共有インフラストラクチャでワークロードを分離する必要がある場合、または複数のデータベースを 1 つのインスタンスに統合する場合は、マルチテナント CDB を選択します。

Oracle データベース 21c 以降では、すべてのデータベースが CDB です。一方、Oracle Database 19c DB インスタンスを CDB または非 CDB として作成することができます。非 CDB を CDB にアップグレードすることはできませんが、Oracle Database 19c の非 CDB を CDB に変換してからアップグレードすることは可能です。CDB を非 CDB に変換することはできません。

詳細については、以下のリソースを参照してください。

- RDS for Oracle で CDB を使用する
- RDS for Oracle CDB の制限事項
- Amazon RDS DB インスタンスの作成

RDS for Oracle 初期化パラメータ

Amazon RDS では、DB パラメータグループを使用してパラメータを管理します。このグループを使用して、初期化パラメータをカスタマイズできます。例えば、sort_area_size を使用してソートエリアサイズを設定できます。特定の DB パラメータグループに関連付けられているすべての RDS for Oracle DB インスタンスは、同じパラメータ設定を使用します。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

RDS for Oracle でサポートされている初期化パラメータ

DB インスタンスでサポートされているパラメータは、Oracle Database のエディションとバージョンによって異なります。特定の Oracle Database エディションとバージョンでサポートされている初期化パラメータを表示するには、AWS CLI コマンド describe-engine-default-parameters を実行します。例えば、Oracle Database 19c の Enterprise Edition でサポートされている初期化パラメータの名前を一覧表示するには、次のコマンドを実行します (サンプル出力を含む)。

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family oracle-ee-19 \
    --output json | jq -r '.EngineDefaults.Parameters[].ParameterName'

_add_col_optim_enabled
_adg_parselock_timeout
_allow_insert_with_update_check
_allow_level_without_connect_by
_always_semi_join
_autotask_max_window
_awr_disabled_flush_tables
_awr_mmon_cpuusage
_awr_mmon_deep_purge_all_expired
_b_tree_bitmap_plans
_bct_bitmaps_per_file
_bloom_filter_enabled
_buffered_publisher_flow_control_threshold
_bug29394014_allow_triggers_on_vpd_table
_cleanup_rollback_entries
_client_enable_auto_unregister
_clusterwide_global_transactions
_complex_view_merging
_connect_by_use_union_all
_cost_equality_semi_join
_cursor_features_enabled
```

```
_cursor_obsolete_threshold
_datafile_write_errors_crash_instance
_db_block_buffers
...
```

単一の初期化パラメータを記述するには、次のコマンドを使用し、sga_max_size をパラメータの名前に置き換えます (サンプル出力を含む)。

```
aws rds describe-engine-default-parameters \
    --db-parameter-group-family oracle-ee-19 \
    --query 'EngineDefaults.Parameters[?ParameterName==`sga_max_size`]' \
    --output json

[
    {
        "ParameterName": "sga_max_size",
        "Description": "max total SGA size",
        "Source": "engine-default",
        "ApplyType": "static",
        "DataType": "integer",
        "AllowedValues": "0-2199023255552",
        "IsModifiable": true
    }
]
```

Oracle Database の初期化パラメータに関する一般的な説明については、Oracle Database マニュアルの「初期化パラメータ」を参照してください。RDS for Oracle では、ARCHIVE_LAG_TARGET パラメータに特別な考慮事項があることに注意してください。このパラメータは、指定した時間が経過すると、強制的にオンライン REDO ログを切り替えます。RDS for Oracle では、目標復旧時点 (RPO) が 5 分であるため、ARCHIVE_LAG_TARGET は 300 に設定されています。この目標を達成するために、RDS for Oracle では 5 分ごとにオンライン REDO ログを切り替えて Amazon S3 バケットに保存しています。

オンライン REDO ログの切り替えの頻度が原因で RDS for Oracle データベースのパフォーマンスが低下する場合は、DB インスタンスとストレージをスケールして、より高い IOPS とスループットを使用できます。また、RDS Custom for Oracle を使用する場合や Amazon EC2 に Oracle データベースをデプロイする場合は、ARCHIVE_LAG_TARGET 初期化パラメータの設定を調整できます。

RDS for Oracle の有効なパラメータ値

RDS for Oracle では、以下の文字のみがパラメータ値として有効です。

- 文字 (A-Z および a-z)
- 数字 (0-9)
- 空白 (スペース、タブ、改行)
- 特殊文字の _ / . : + = ( ) ' * , % $ - (ハイフン)

RDS for Oracle 文字セット

RDS for Oracle では、DB 文字セットと各国語文字セットの 2 種類の文字セットがサポートされています。

DB 文字セット

Oracle データベースの文字セットは CHAR、VARCHAR2、CLOB の各データ型で使用します。データベースは、テーブル名、列名、SQL 文などのメタデータにも、この文字セットを使用します。Oracle データベースの文字セットは、通常 DB 文字セットと呼ばれます。

文字セットは、DB インスタンスの作成時に設定します。データベースの作成後に DB 文字セットを変更することはできません。

サポートされている DB 文字セット

次の表は、Amazon RDS でサポートされている Oracle DB 文字セットの一覧です。このテーブルの値は、AWS CLI の create-db-instance コマンドの --character-set-name パラメータ、または Amazon RDS API の CreateDBInstance オペレーションの CharacterSetName パラメータで使用できます。

> **Note**
>
> CDB の文字セットは常に AL32UTF8 です。PDB のためにのみ、別の文字セットを設定できます。

値	説明
AL32UTF8	Unicode 5.0 UTF-8 Universal 文字セット (デフォルト)
AR8ISO8859P6	ISO 8859-6 ラテン文字/アラビア文字

値	説明
AR8MSWIN1256	Microsoft Windows コードページ 1256 8-bit ラテン文字/アラビア文字
BLT8ISO8859P13	ISO 8859-13 バルト語
BLT8MSWIN1257	Microsoft Windows コードページ 1257 8-bit バルト語
CL8ISO8859P5	ISO 8859-5 ラテン文字/キリル文字
CL8MSWIN1251	Microsoft Windows コードページ 1251 8-bit ラテン文字/キリル文字
EE8ISO8859P2	ISO 8859-2 東ヨーロッパ
EL8ISO8859P7	ISO 8859-7 ラテン文字/ギリシャ文字
EE8MSWIN1250	Microsoft Windows コードページ 1250 8-bit 東ヨーロッパ
EL8MSWIN1253	Microsoft Windows コードページ 1253 8-bit ラテン文字/ギリシャ文字
IW8ISO8859P8	ISO 8859-8 ラテン文字/ヘブライ文字
IW8MSWIN1255	Microsoft Windows コードページ 1255 8-bit ラテン文字/ヘブライ文字
JA16EUC	EUC 24-bit 日本語
JA16EUCTILDE	Unicode との間の波線およびチルダのマッピングを除き、JA16EUC と同じ
JA16SJIS	Shift-JIS 16-bit 日本語
JA16SJISTILDE	Unicode との間の波線およびチルダのマッピングを除き、JA16SJIS と同じ
KO16MSWIN949	Microsoft Windows コードページ 949 韓国語

値	説明
NE8ISO8859P10	ISO 8859-10 北ヨーロッパ
NEE8ISO8859P4	ISO 8859-4 北および北東ヨーロッパ
TH8TISASCII	タイ語 Industrial Standard 620-2533-ASCII 8-bit
TR8MSWIN1254	Microsoft Windows コードページ 1254 8-bit トルコ語
US7ASCII	ASCII 7-bit アメリカ
UTF8	Unicode 3.0 UTF-8 Universal 文字セット、CESU-8 準拠
VN8MSWIN1258	Microsoft Windows コードページ 1258 8-bit ベトナム語
WE8ISO8859P1	西ヨーロッパ 8-bit ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 西ヨーロッパ
WE8ISO8859P9	ISO 8859-9 西ヨーロッパおよびトルコ語
WE8MSWIN1252	Microsoft Windows コードページ 1252 8-bit 西ヨーロッパ
ZHS16GBK	GBK 16-bit 簡体字中国語
ZHT16HKSCS	香港補助文字セット HKSCS-2001 を含む Microsoft Windows コードページ 950。文字セット変換は、Unicode 3.0 に基づいています。
ZHT16MSWIN950	Microsoft Windows コードページ 950 繁体字中国語
ZHT32EUC	EUC 32-bit 繁体字中国語

NLS_LANG 環境変数

ロケールとは、特定の言語と国に対する言語上の条件と文化的条件に対処するための一連の情報です。Oracle のロケール動作を指定するには、クライアントの環境で NLS_LANG 環境変数を設定するのが最も簡単です。この変数は、クライアントアプリケーションとデータベースサーバーで使用する言語や地域を設定します。また、クライアントの文字セットを指定します。これは、クライアントアプリケーションによって入力または表示されるデータの文字セットに対応します。NLS_LANG と文字セットの詳細については、Oracle ドキュメントの「キャラクタ・セットまたはコードページとは何ですか」を参照してください。

NLS 初期化パラメータ

また、次の各国語サポート (NLS) の初期化パラメータを Amazon RDS の Oracle DB インスタンスのインスタンスレベルで設定することもできます。

- NLS_COMP
- NLS_DATE_FORMAT
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_SORT
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

インスタンスのパラメータの変更については、「Amazon RDS のパラメータグループ」を参照してください。

SQL クライアントで NLS 初期化パラメータを設定できます。例えば、次のステートメントで Oracle DB インスタンスに接続している SQL クライアントで NLS_LANGUAGE 初期化パラメータをドイツ語に設定します。

```
ALTER SESSION SET NLS_LANGUAGE=GERMAN;
```

Oracle DB インスタンスを SQL クライアントに接続する詳細については、「Oracle DB インスタンスへの接続」を参照してください。

各国語文字セット

各国語文字セットは、NCHAR、NVARCHAR2、NCLOB の各データ型で使用します。各国語文字セットは、通常 NCHAR 文字セットと呼ばれます。DB 文字セットとは異なり、NCHAR 文字セットはデータベースのメタデータには影響しません。

NCHAR 文字セットは、次の文字セットをサポートしています。

- AL16UTF16 (デフォルト)
- UTF8

create-db-instance コマンド (`--nchar-character-set-name` バージョン 2 のみ) の AWS CLI パラメータを使用して、どちらの値も指定できます。Amazon RDS API を使用する場合は、CreateDBInstance オペレーションの `NcharCharacterSetName` パラメータを指定します。データベースの作成後に各国語文字セットを変更することはできません。

Oracle データベースの Unicode の詳細については、Oracle ドキュメントの「Unicode を使用した多言語データベースのサポート」を参照してください。

RDS for Oracle の制限事項

以下のセクションでは、RDS for Oracle を使用する際の重要な制限について説明します。CDB 固有の制限については、「RDS for Oracle CDB の制限事項」を参照してください。

> **Note**
>
> これはすべてを網羅したリストではありません。

トピック

- Amazon RDS での Oracle のファイルサイズ制限
- RDS for Oracle のブロックサイズ制限
- Oracle が提供するスキーマのパブリックシノニム
- RDS for Oracle ではサポートされていない機能のスキーマ
- RDS for Oracle には、DBA 権限に制限があります。
- RDS for Oracle での、TLS 1.0 および 1.1 Transport Layer Security の非推奨

Amazon RDS での Oracle のファイルサイズ制限

RDS for Oracle DB インスタンスの 1 ファイルの最大サイズは 16 TiB (テビバイト) です。この制限は、インスタンスが使用する ext4 ファイルシステムによって設定されます。したがって、Oracle のビッグファイルデータファイルは 16 TiB に制限されています。bigfile テーブルスペース内のデータファイルを制限を超える値に変更しようとすると、以下のようなエラーが発生します。

```
ORA-01237: cannot extend datafile 6
ORA-01110: data file 6: '/rdsdbdata/db/mydir/datafile/myfile.dbf'
ORA-27059: could not reduce file size
Linux-x86_64 Error: 27: File too large
Additional information: 2
```

RDS for Oracle のブロックサイズ制限

RDS for Oracle DB インスタンスは、デフォルトのデータベースブロックサイズ (DB_BLOCK_SIZE) が 8 KB で作成されます。デフォルトのデータベースブロックサイズはデータベースの作成時に設定され、変更できません。SYSTEM および一時テーブルスペースは、常にデフォルトのデータベースブロックサイズを使用します。対応する DB_nK_CACHE_SIZE パラメータ (DB_16K_CACHE_SIZE など) を設定してブロックサイズのバッファキャッシュを割り当て、CREATE TABLESPACE ステートメントで BLOCKSIZE 句を指定することで、デフォルト以外のブロックサイズを持つ追加のテーブルスペースを作成できます。

Oracle が提供するスキーマのパブリックシノニム

Oracle 提供のスキーマのパブリックシノニム (例: SYS、SYSTEM、RDSADMIN) を作成または変更しないでください。それを行うと、コアデータベースコンポーネントは無効になり、DB インスタンスの可用性に影響を及ぼす可能性があります。

自分のスキーマ内のオブジェクトを参照するパブリックシノニムを作成できます。

RDS for Oracle ではサポートされていない機能のスキーマ

一般に、Amazon RDS では、サポートされていない機能のスキーマを作成することはできません。ただし、SYS 権限を必要とする Oracle 機能およびコンポーネントのスキーマを作成すると、データディクショナリが破損し、お使いのインスタンスの可用性に影響する可能性があります。Oracle DB インスタンスへのオプションの追加 で使用可能なサポートされている機能およびスキーマのみを使用します。

RDS for Oracle には、DBA 権限に制限があります。

ロールとは、ユーザーに対して付与または取り消すことができる権限のコレクションです。Oracle データベースは、セキュリティを提供するためにロールを使用します。

事前に定義されたロール DBA は、通常、Oracle データベースに対するすべての管理権限を付与します。マスターユーザーアカウントを使用して DB インスタンスを作成すると、アカウントには DBA 権限が付与されます。ただし、いくつかの制限があります。マネージドエクスペリエンスを提供するために、RDS for Oracle データベースは DBA ロールに次の権限を提供しません。

- ALTER DATABASE
- ALTER SYSTEM
- CREATE ANY DIRECTORY
- DROP ANY DIRECTORY
- GRANT ANY PRIVILEGE
- GRANT ANY ROLE

データベースで追加のユーザーアカウントを作成するなどの管理タスクには、このマスターユーザーアカウントを使用します。SYS、SYSTEM および Oracle が提供するその他の管理アカウントを使用することはできません。

RDS for Oracle での、TLS 1.0 および 1.1 Transport Layer Security の非推奨

Transport Layer Security プロトコルバージョン 1.0 および 1.1 (TLS 1.0 および TLS 1.1) は非推奨です。セキュリティのベストプラクティスに従って、Oracle は TLS 1.0 および TLS 1.1 の使用を非推奨としました。セキュリティ要件を満たすために、代わりに TLS 1.2 を使用することを強くお勧めします。

Oracle DB インスタンスへの接続

Amazon RDS によって Oracle DB インスタンスがプロビジョニングされた後、標準の SQL クライアントアプリケーションを使用して DB インスタンスにログインできます。RDS はマネージドサービスであるため、SYS または SYSTEM としてログインすることはできません。詳細については、「RDS for Oracle のユーザーと権限」を参照してください。

このトピックでは、Oracle SQL Developer または SQL*Plus を使用して RDS for Oracle DB インスタンスに接続する方法について説明します。サンプルの DB インスタンスの作成と接続のプロセスを示す手順の例は、「Oracle DB インスタンスを作成して接続する」を参照してください。

トピック

- RDS for Oracle DB インスタンスのエンドポイントを見つける
- Oracle SQL Developer を使用した DB インスタンスへの接続
- SQL *Plus を使用した DB インスタンスへの接続
- セキュリティグループに関する考慮事項
- プロセスアーキテクチャに関する考慮事項
- Oracle DB インスタンスへの接続のトラブルシューティング
- sqlnet.ora パラメータを使用した接続プロパティの変更

RDS for Oracle DB インスタンスのエンドポイントを見つける

各 Amazon RDS DB インスタンスにはエンドポイントがあり、各エンドポイントに DB インスタンスの DNS 名とポート番号があります。SQL クライアントアプリケーションを使用して DB インスタンスに接続するには、DB インスタンスの DNS 名とポート番号が必要です。

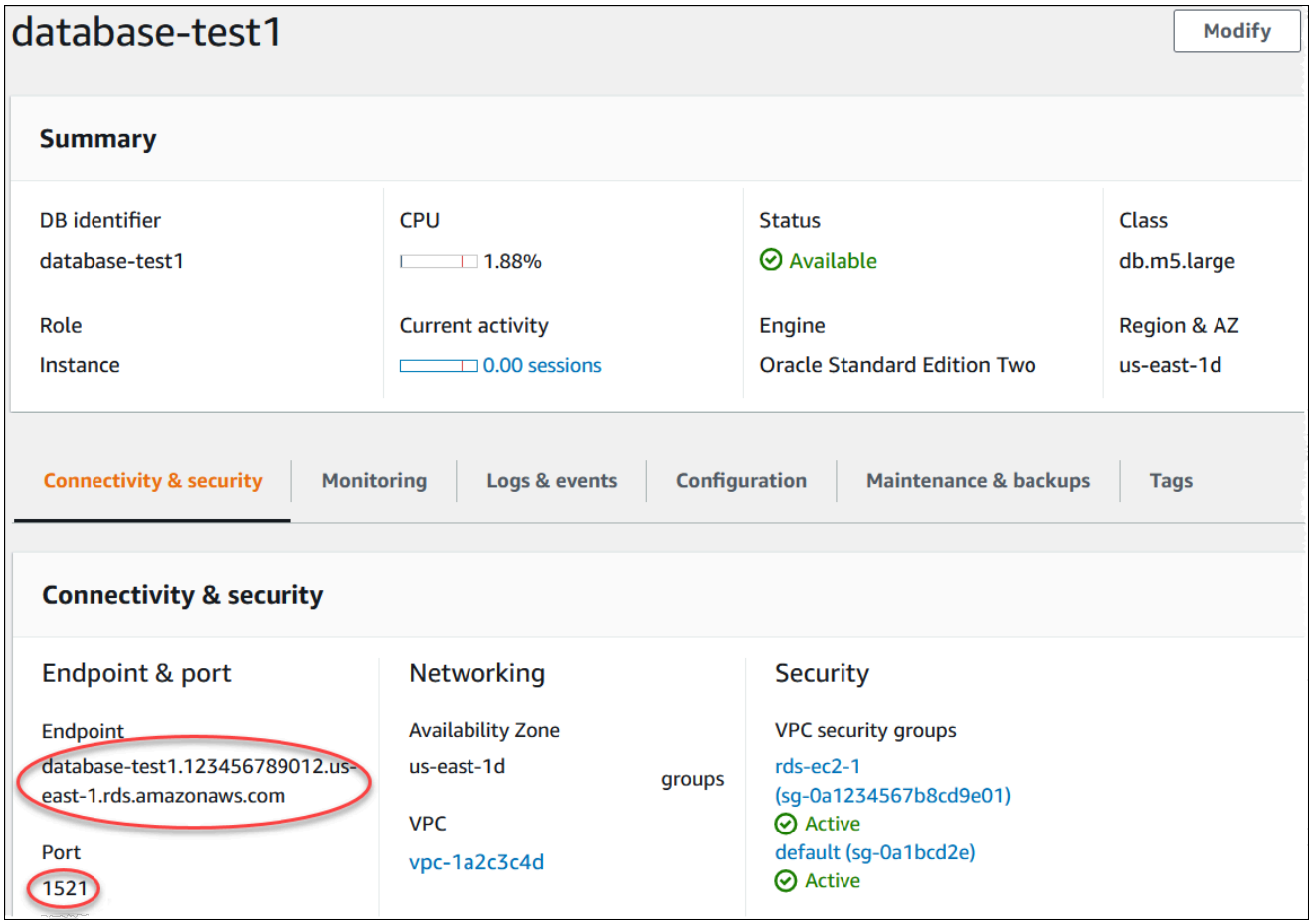
Amazon RDS コンソールまたは AWS CLI を使用して、DB インスタンスのエンドポイントを見つけることができます。

> **Note**
>
> Kerberos 認証を使用している場合は、「Oracle を Kerberos 認証に接続する」を参照してください。

コンソール

コンソールを使用してエンドポイントを見つけるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. コンソールで、DB インスタンスの AWS リージョンを選択します。
3. DB インスタンスの DNS 名とポート番号を見つけます。
   a. [データベース] を選択して DB インスタンスを一覧表示します。
   b. 詳細を表示する Oracle DB インスタンスの名前を選択します。
   c. [接続とセキュリティ] タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



AWS CLI

AWS CLI を使用して Oracle DB インスタンスのエンドポイントを確認するには、describe-db-instances コマンドを呼び出します。

Example AWS CLI を使用してエンドポイントを見つけるには

```
aws rds describe-db-instances
```

出力で `Endpoint` を検索して、DB インスタンスの DNS 名とポート番号を検索します。出力の `Address` 行には DNS 名が含まれています。JSON エンドポイント出力の例を以下に示します。

```
"Endpoint": {
    "HostedZoneId": "Z1PVIF0B656C1W",
    "Port": 1521,
    "Address": "myinstance.123456789012.us-west-2.rds.amazonaws.com"
},
```

> **Note**
>
> 出力には、複数の DB インスタンスに関する情報が含まれている場合があります。

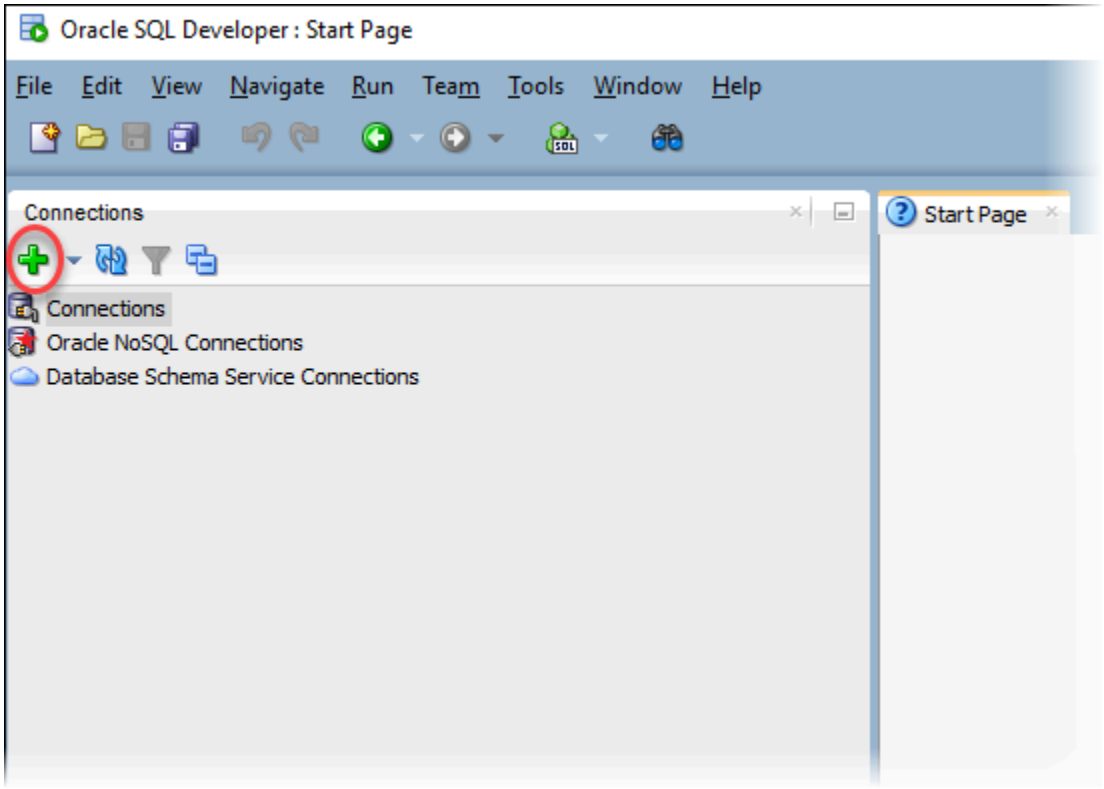
Oracle SQL Developer を使用した DB インスタンスへの接続

この手順では、Oracle SQL Developer を使用して DB インスタンスに接続します。このユーティリティのスタンドアロンバージョンをダウンロードするには、Oracle SQL デベロッパーのダウンロードページを参照してください。

DB インスタンスに接続するには、DNS 名とポート番号が必要です。DB インスタンスの DNS 名とポート番号を見つける方法については、RDS for Oracle DB インスタンスのエンドポイントを見つける を参照してください。

SQL Developer を使用して DB インスタンスに接続するには

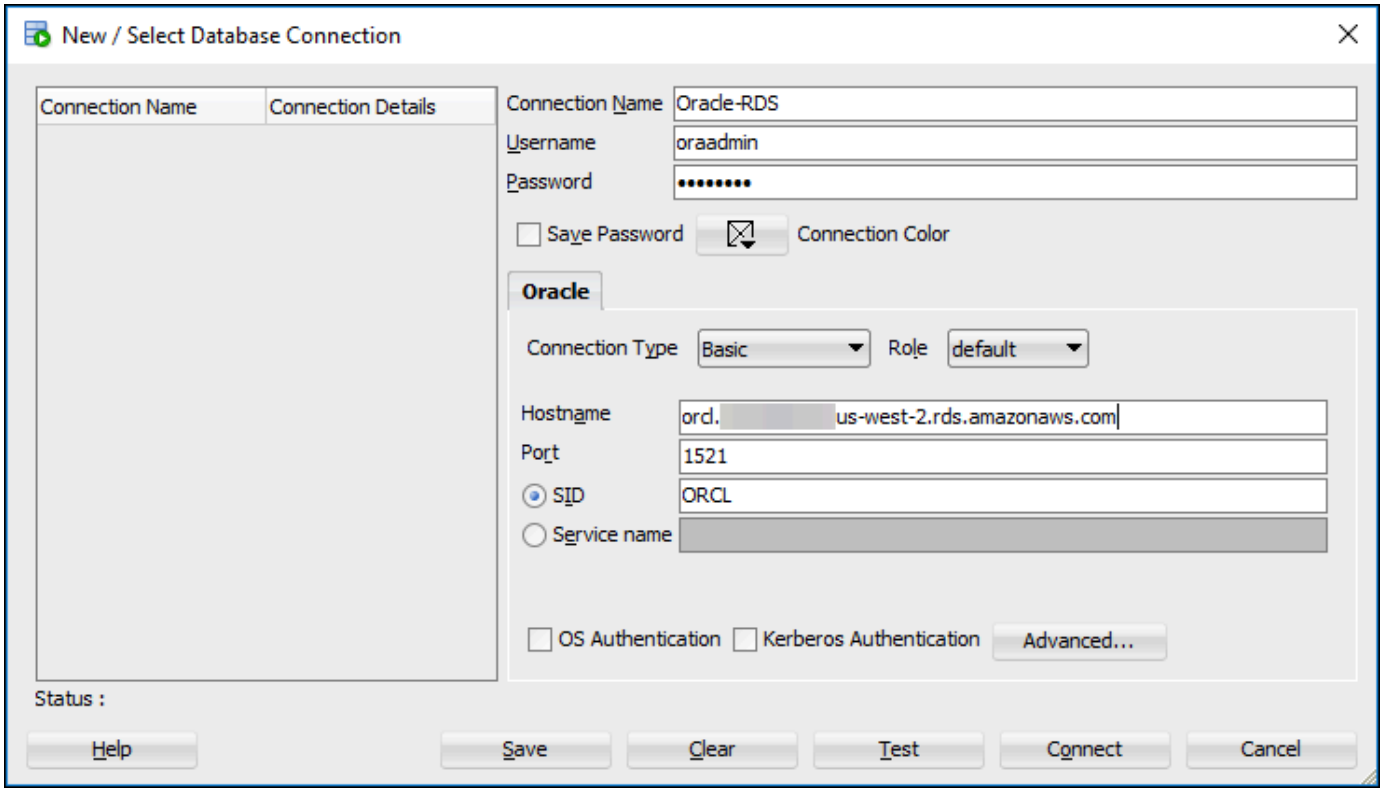
1. Oracle SQL Developer をスタートします。
2. [Connections] タブで、[add (+)] アイコンを選択します。



3. [New/Select Database Connection] ダイアログボックスで、DB インスタンスの情報を提供します。

   - [Connection Name (接続名)] に、接続の名前 (`Oracle-RDS` など) を入力します。
   - [Username (ユーザーネーム)] に、DB インスタンスのデータベース管理者の名前を入力します。
   - [Password (パスワード)] に、データベース管理者のパスワードを入力します。
   - [Hostname (ホスト名)] に、DB インスタンスの DNS 名を入力します。
   - [Port (ポート)] に、ポート番号を入力します。
   - SID には、DB 名を入力します。DB 名は、データベース詳細ページの [Configuration] (設定) タブで確認できます。
   - DB インスタンスが CDB アーキテクチャ (マルチテナント) を使用している場合は、[SID] の代わりに [サービス名] を選択し、PDB 名を入力してテナントデータベースに直接接続することもできます。

   完了したダイアログボックスは次のように表示されます。

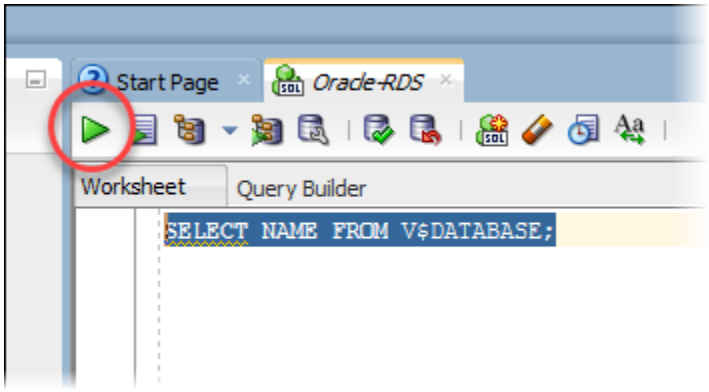


4. 接続 を選択します。
5. これで、通常どおり DB インスタンスに対しクエリを実行できるようになりました。DB インスタンスに対してテストクエリを実行するには、次を実行します。

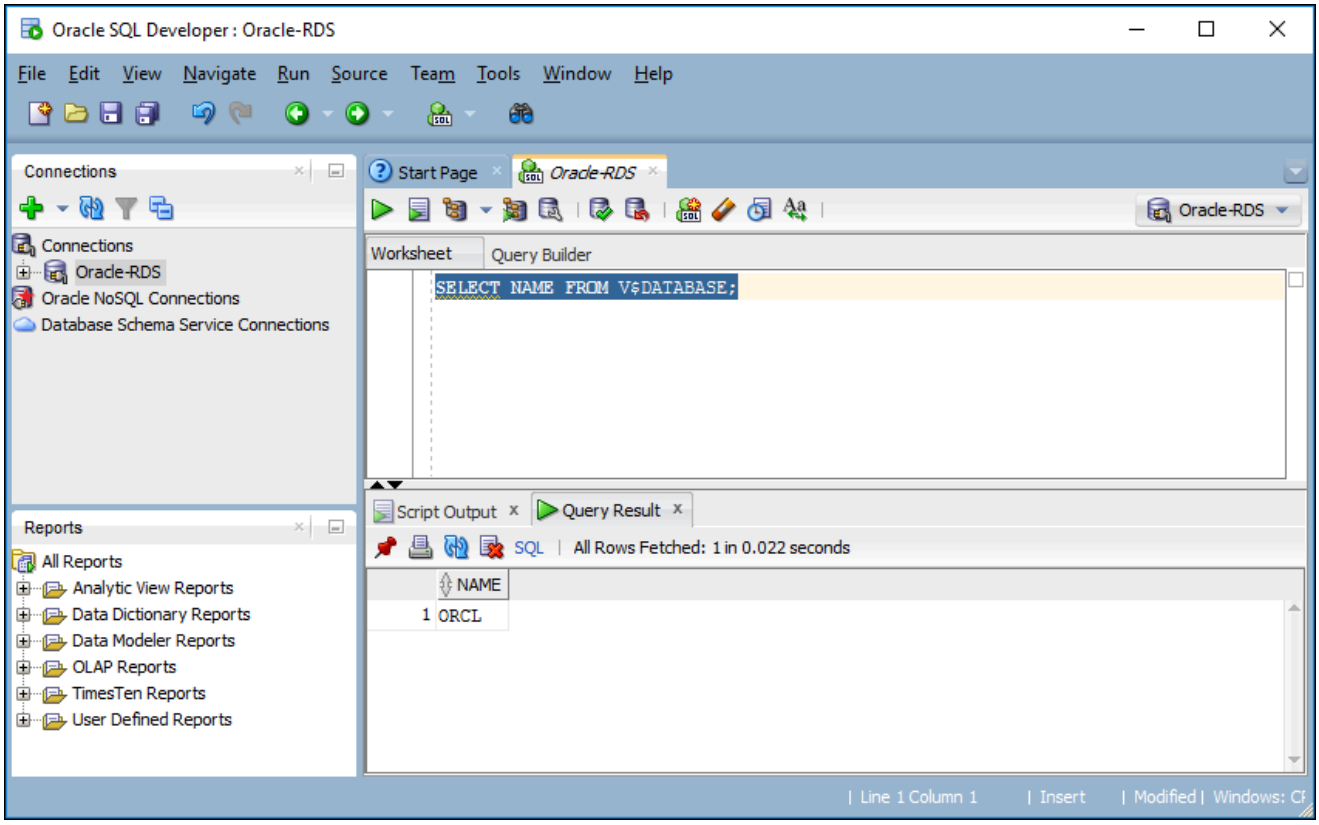
   a. 接続の [Worksheet (ワークシート)] タブに、次の SQL クエリを入力します。

   ```
   SELECT NAME FROM V$DATABASE;
   ```

   b. 実行アイコンを選択して、クエリを実行します。

   SQL Developer はデータベース名を返します。



SQL *Plus を使用した DB インスタンスへの接続

SQL*Plus などのユーティリティを使用して、Oracle を実行している Amazon RDS DB インスタンスに接続できます。SQL*Plus のスタンドアロンバージョンを含む Oracle Instant Client をダウンロードするには、Oracle Instant Client Downloads を参照してください。

DB インスタンスに接続するには、DNS 名とポート番号が必要です。DB インスタンスの DNS 名とポート番号を見つける方法については、RDS for Oracle DB インスタンスのエンドポイントを見つける を参照してください。

Example SQL*Plus を使用して Oracle DB インスタンスに接続するには

次の例では、DB インスタンス管理者のユーザー名を使用します。また、DNS 名を DB インスタンスに置き換えて、ポート番号および Oracle SID を含めます。SID 値は、DB インスタンスを作成したときに指定した DB インスタンスのデータベースの名前であり、DB インスタンスの名前ではありません。

Linux、macOS、Unix の場合:

```
sqlplus 'user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))'
```

Windows の場合:

```
sqlplus user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))
```

次のような出力が表示されます。

```
SQL*Plus: Release 19.0.0.0.0 Production on Wed Jan 15 14:23:45 2025
```

ユーザーのパスワードを入力すると、SQL プロンプトが表示されます。

```
SQL>
```

> **Note**
>
> SQL*Plus は、コマンドの実行後にパスワードの入力を求めます。または、パスワードをインラインで指定することもできます (例: *user_name*/*password*@(DESCRIPTION=...))。ただし、この方法ではシェル履歴にパスワードが公開されます。

> **Note**
>
> `sqlplus USER/PASSWORD@`*longer-than-63-chars-rds-endpoint-here*`:1521/`*database-identifier* のような短い形式の接続文字列 (EZ Connect) は、最大文字数制限に達する可能性があるため、接続には使用しないことをお勧めします。

> **Tip**
>
> エラーを返さずに接続がハングする場合は、セキュリティグループがクライアントの IP アドレスからのインバウンドトラフィックをデータベースポートで許可していることを確認します。

セキュリティグループに関する考慮事項

DB インスタンスに接続するためには、DB インスタンスを、必要な IP アドレスとネットワーク設定を含むセキュリティグループに関連付ける必要があります。DB インスタンスは、デフォルトのセキュリティグループを使用することがあります。DB インスタンスの作成時に、デフォルトの設定されていないセキュリティグループを割り当てた場合は、ファイアウォールによって接続が禁止されます。新しいセキュリティグループの作成方法については、セキュリティグループによるアクセス制御を参照してください。

新しいセキュリティグループを作成したら、そのセキュリティグループと関連付けるように DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

SSL を使用して DB インスタンスへの接続を暗号化することで、セキュリティを高めることができます。詳細については、「Oracle Secure Sockets Layer」を参照してください。

プロセスアーキテクチャに関する考慮事項

サーバープロセスは、ユーザーの Oracle DB インスタンスへの接続を処理します。デフォルトでは、Oracle DB インスタンスは専用サーバープロセスを使用します。専用サーバープロセスでは、各サーバープロセスは 1 人のユーザープロセスにのみ設定できます。任意で共有サーバープロセスを設定できます。共有サーバープロセスでは、各サーバープロセスは複数のユーザープロセスを設定できます。

多数のユーザーセッションがサーバー上でメモリを過度に使用している場合は、共有サーバープロセスの使用を検討することもできます。また、セッションが頻繁に接続されたり切断されてパフォーマンス上の問題を引き起こす場合も、共有サーバープロセスを検討できます。共有サーバープロセスの使用には欠点もあります。例えば、CPU リソースに負荷をかけることがあり、設定や管理がより複雑になります。

専用インスタンスと共有サーバーのプロセスの詳細については、Oracle ドキュメントの「専用および共有サーバープロセスについて」を参照してください。RDS for Oracle DB インスタンスで共有サーバープロセスを設定する方法の詳細については、ナリッジセンターの共有サーバーを使用するように Amazon RDS for Oracle Database を設定する方法を参照してください。

Oracle DB インスタンスへの接続のトラブルシューティング

以下は、Oracle DB インスタンスの接続時に発生する可能性がある問題です。

問題	トラブルシューティングの推奨事項
DB インスタンスに接続できません。	新しく作成された DB インスタンスでは、使用できるようになるまで、DB インスタンスのステータスは [creating] となります。ステータスが [available] に変わると、DB インスタンスに接続できます。DB インスタンスクラスとストレージの合計によっては、新しい DB インスタンスを使用できるようになるまで最長 20 分かかることがあります。
DB インスタンスに接続できません。	DB インスタンスを作成したときに指定したポートを経由して通信を送信または受信できない場合は、DB インスタンスに接続できません。DB インスタンスに指定したポートでインバウンドおよびアウトバウンド通信できることを検証するよう、ネットワーク管理者に確認してください。
DB インスタンスに接続できません。	ローカルファイアウォールによって実施されるアクセスルールと、DB インスタンスのセキュリティグループで DB インスタンスへのアクセスを許可した IP アドレスが一致しない可能性があります。問題は、ほとんどの場合ファイアウォールのインバウンドまたはアウトバウンドルールです。 セキュリティグループでインバウンドのルールを追加または編集できます。[ソース] には [マイ IP] を選択します。これにより、ブラウザで検出された IP アドレスから DB インスタンスへのアクセスが許可されます。詳細については、「Amazon VPC と Amazon RDS」を参照してください。 セキュリティグループの詳細については、セキュリティグループによるアクセス制御 を参照してください。 セキュリティグループにルールを設定する手順については、チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用) を参照してください。
ターゲットホストまたはオブジェクトが存在しないため接続に失敗しま	サーバー名とポート番号を正しく指定していることを確認します。[Server name (サーバー名)] に、コンソールからの DNS 名を入力します。

問題	トラブルシューティングの推奨事項
した - Oracle、エラー: ORA-12545	DB インスタンスの DNS 名とポート番号を見つける方法については、RDS for Oracle DB インスタンスのエンドポイントを見つけるを参照してください。
無効なユーザーネーム/パスワード、ログインが拒否されました - Oracle、エラー: ORA-01017	DB インスタンスには到達できましたが、接続が拒否されました。通常、これは誤ったユーザー名やパスワードが指定されたときに発生します。ユーザー名とパスワードを確認し、再試行します。
TNS: リスナーは現在、接続ディスクリプタで指定された SID を知りません - Oracle、エラー: ORA-12505	正しい SID が入力されていることを確認してください。SID は DB 名と同じです。インスタンスの [Databases] (データベース) ページの [Configuration] (設定) タブで DB 名を探します。また、AWS CLI を使用して DB 名を確認することもできます。 `aws rds describe-db-instances --query 'DBInstances[*].[DBInstanceIdentifier,DBName]' --output text`
アカウントがロックされています – Oracle、エラー: ORA-28000	ログイン試行に何度も失敗したため、ユーザーアカウントがロックされました。マスターユーザーとして接続し、以下を実行します。 `ALTER USER username ACCOUNT UNLOCK;`
パスワードの有効期限が切れています – Oracle、エラー: ORA-28001	割り当てられたプロファイルに基づき、ユーザーパスワードの有効期限が切れています。マスターユーザーとして接続し、パスワードをリセットします。 `ALTER USER username IDENTIFIED BY new_password ;`

接続の問題の詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

sqlnet.ora パラメータを使用した接続プロパティの変更

sqlnet.ora ファイルに含まれているパラメータでは、Oracle データベースサーバーおよびクライアントの Oracle Net 機能を設定します。sqlnet.ora ファイルのパラメータを使用して、データベースに出入りする接続のプロパティを変更できます。

sqlnet.ora を設定する理由の詳細については、Oracle ドキュメントのプロファイルパラメータの設定に関する記事を参照してください。

sqlnet.ora パラメータの設定

Amazon RDS for Oracle パラメータグループには、sqlnet.ora パラメータのサブセットが含まれています。これらのパラメータは、他の Oracle パラメータと同じ方法で設定します。`sqlnetora.` プレフィックスは、どのパラメータが sqlnet.ora パラメータであるかを判別します。例えば、Amazon RDS の Oracle パラメータグループの場合、`default_sdu_size` sqlnet.ora パラメータは `sqlnetora.default_sdu_size` です。

パラメータグループの管理とパラメータ値の設定については、「Amazon RDS のパラメータグループ」を参照してください。

サポートされている sqlnet.ora パラメータ

Amazon RDS は、以下の sqlnet.ora パラメータをサポートしています。動的な sqlnet.ora パラメータの変更は即時に反映されます。

Parameter	有効な値	静的/動的	説明
`sqlnetora.default_sdu_size`	`512` ～ `2097`	動的	セッションデータユニット (SDU) のサイズ (バイト単位)。 SDU は、バッファに配置され、ネットワークで一度に送信されるデータの量です。
`sqlnetora.diag_adr_enabled`	`ON`, `OFF`	動的	自動診断リポジトリ (ADR) のトレースを有効化/無効化する値。

Parameter	有効な値	静的/動的	説明
			ON は、ADR ファイルのトレースを使用することを指定します。 OFF は、ADR 以外のファイルのトレースを使用することを指定します。
sqlnetora.recv_buf_size	8192 ~ 2684	動的	セッションの受信オペレーションのバッファ容量制限。TCP/IP、TCP/IP with SSL、SDP の各プロトコルでサポートされます。
sqlnetora.send_buf_size	8192 ~ 2684	動的	セッションの送信オペレーションのバッファ容量制限。TCP/IP、TCP/IP with SSL、SDP の各プロトコルでサポートされます。
sqlnetora.sqlnet.allowed_logon_version_client	8, 10, 11, 12	動的	Oracle DB インスタンスへの接続を確立するために、クライアント、およびクライアントとして機能するサーバーに許可される最小認証プロトコルバージョン。
sqlnetora.sqlnet.allowed_logon_version_server	8, 9, 10, 11, 12, 12a	動的	Oracle DB インスタンスへの接続を確立するために許可される最小認証プロトコルバージョン。

Parameter	有効な値	静的/動的	説明
sqlnetora.sqlnet.expire_time	0 ～1440	動的	チェックを送信してクライアントサーバー接続がアクティブであることを確認する時間間隔 (分単位)。
sqlnetora.sqlnet.inbound_connect_timeout	0、または 10～72	動的	クライアントからデータベースサーバーに接続し、必要な認証情報を提供するまでの時間 (秒単位)。
sqlnetora.sqlnet.outbound_connect_timeout	0、または 10～72	動的	クライアントから DB インスタンスへの Oracle Net 接続を確立するまでの時間 (秒単位)。
sqlnetora.sqlnet.recv_timeout	0、または 10～72	動的	接続の確立後にデータベースサーバーがクライアントデータを待機する時間 (秒単位)。
sqlnetora.sqlnet.send_timeout	0、または 10～72	動的	接続の確立後にデータベースサーバーからクライアントへの送信オペレーションが完了するまでの時間 (秒単位)。
sqlnetora.tcp.connect_timeout	0、または 10～72	動的	クライアントからデータベースサーバーへの TCP 接続を確立するまでの時間 (秒単位)。

Parameter	有効な値	静的/動的	説明
sqlnetora.trace_level_server	0, 4, 10, 16, OFF, USER, ADMIN, SUPPOF	動的	ADR 以外のトレースの場合、指定したレベルでトレースをオンにするか、トレースをオフにします。

サポートされている各 sqlnet.ora パラメータのデフォルト値は、リリースの Oracle Database デフォルトです。

sqlnet.ora パラメータの表示

AWS マネジメントコンソール、AWS CLI、または SQL を使用して、sqlnet.ora のパラメータとその設定を表示できます。

コンソールでの sqlnet.ora パラメータの表示

パラメータグループ内のパラメータの表示方法については、「Amazon RDS のパラメータグループ」を参照してください。

Oracle パラメータグループでは、sqlnetora. プレフィックスにより、どのパラメータが sqlnet.ora パラメータであるかを判別します。

AWS CLI での sqlnet.ora パラメータの表示

Oracle パラメータグループに設定されている sqlnet.ora パラメータを表示するには、AWS CLI の describe-db-parameters コマンドを使用します。

Oracle DB インスタンスのすべての sqlnet.ora パラメータを表示するには、AWS CLI の download-db-log-file-portion コマンドを呼び出します。DB インスタンス識別子、ログファイル名、および出力のタイプを指定します。

Example

次のコードでは、mydbinstance のすべての sqlnet.ora パラメータを一覧表示します。

Linux、macOS、Unix の場合:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier mydbinstance \
    --log-file-name trace/sqlnet-parameters \
    --output text
```

Windows の場合:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier mydbinstance ^
    --log-file-name trace/sqlnet-parameters ^
    --output text
```

SQL クライアントでの sqlnet.ora パラメータの表示

SQL クライアントで Oracle DB インスタンスに接続すると、次のクエリで sqlnet.ora パラメータが一覧表示されます。

```
SELECT * FROM TABLE
   (rdsadmin.rds_file_util.read_text_file(
        p_directory => 'BDUMP',
        p_filename  => 'sqlnet-parameters'));
```

SQL クライアントで Oracle DB インスタンスに接続する方法については、「Oracle DB インスタンスへの接続」を参照してください。

Oracle DB インスタンス接続の保護

Amazon RDS for Oracle では SSL/TLS 暗号化接続がサポートされています。また、Oracle Native Network Encryption (NNE) オプションを使用して、アプリケーションと Oracle DB インスタンス間の接続を暗号化できます。Oracle Native Network Encryption オプションについては、「Oracle ネイティブネットワーク暗号化」を参照してください。

トピック

- RDS for Oracle DB インスタンスでの SSL の使用
- 新しい SSL/TLS 証明書を使用して Oracle DB インスタンスに接続するようにアプリケーションを更新する
- RDS for Oracle DB インスタンスでネイティブネットワークの暗号化を使用する
- Amazon RDS for Oracle の Kerberos 認証の設定
- 証明書と Oracle ウォレットを使用した、UTL_HTTP アクセスの設定

RDS for Oracle DB インスタンスでの SSL の使用

Secure Sockets Layer (SSL) は、クライアントとサーバー間のネットワーク接続を安全に保つための業界標準のプロトコルです。SSL バージョン 3.0 以降は、名前が Transport Layer Security (TLS) と変更されていますが、いまだに SSL と呼称されることもよくあります。Amazon RDS は、Oracle DB インスタンス向けに SSL での暗号化をサポートしています。SSL を使用して、アプリケーションクライアントと Oracle DB インスタンス間の接続を暗号化できます。SSL は、Oracle 用のすべての AWS リージョンでサポートされています。

Oracle DB インスタンスの SSL 暗号化を有効するには、その DB インスタンスに関連付けられているオプショングループに、Oracle SSL オプションを追加します。Oracle からの要求があるため、Amazon RDS では SSL 接続のために 2 番目のポートを使用しています。こうすることで、クリアテキストと SSL 暗号化の両方の通信を、DB インスタンスと Oracle クライアント間で同時に実行できます。例えば、このポートで SSL 暗号化通信を使用して VPC 外部のリソースと通信する一方で、このポートでクリアテキスト通信を使用して VPC 内の他のリソースと通信できます。

詳細については、「Oracle Secure Sockets Layer」を参照してください。

> **Note**
>
> SSL と Oracle ネイティブネットワークの暗号化 (NNE) を同じ DB インスタンスで両方使用することはできません。SSL 暗号化を使用する前に、他のすべての接続の暗号化を無効にする必要があります。

新しい SSL/TLS 証明書を使用して Oracle DB インスタンスに接続するようにアプリケーションを更新する

2023 年 1 月 13 日に Amazon RDS は、Secure Socket Layer または Transport Layer Security (SSL/TLS) を使用して RDS DB インスタンスに接続するための新しい認証局 (CA) 証明書を公開しました。ここでは、新しい証明書を使用するためのアプリケーションの更新について説明します。

このトピックでは、クライアントアプリケーションが SSL/TLS を使用して DB インスタンスに接続されているかどうかを判断できます。

> **Important**
>
> Amazon RDS for Oracle DB インスタンスの証明書を変更すると、データベースリスナーのみが再起動されます。DB インスタンスは再起動されません。既存のデータベース接続は影響を受けませんが、新しい接続ではリスナーが再起動されるまで短期間だけエラーが発生します。
> 接続エラーを防ぐために Oracle DB を再起動することをお勧めします。

> **Note**
>
> DB インスタンスに接続するために SSL/TLS を使用するクライアントアプリケーションの場合、新しい CA 証明書を含めるためにクライアントアプリケーション信頼ストアを更新する必要があります。

クライアントアプリケーションの信頼ストアで CA 証明書を更新した後、DB インスタンスで証明書をローテーションできます。これらの手順を開発環境またはステージング環境でテストしてから、本番環境で実装することを強くお勧めします。

証明書のローテーションの詳細については、「SSL/TLS 証明書のローテーション」を参照してください。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。Oracle DB インスタンスで SSL/TLS を使用する方法については、「Oracle Secure Sockets Layer」を参照してください。

トピック

- アプリケーションが SSL を使用して接続しているかどうかを調べる
- アプリケーション信頼ストアの更新
- SSL 接続を確立するための Java コードの例

アプリケーションが SSL を使用して接続しているかどうかを調べる

Oracle DB インスタンスで SSL オプションが追加されたオプショングループを使用する場合、SSL を使用している可能性があります。オプショングループのオプションとオプション設定をリスト化する の手順に従ってこれをチェックします。SSL オプションの詳細については、「Oracle Secure Sockets Layer」を参照してください。

リスナーログをチェックして、SSL 接続があるかどうかを判断します。リスナーログのサンプル出力を次に示します。

```
date time * (CONNECT_DATA=(CID=(PROGRAM=program)
(HOST=host)(USER=user))(SID=sid)) *
(ADDRESS=(PROTOCOL=tcps)(HOST=host)(PORT=port)) * establish * ORCL * 0
```

PROTOCOL にエントリの値 tcps がある場合、SSL 接続を示します。ただし、HOST が 127.0.0.1 である場合は、エントリを無視できます。127.0.0.1 からの接続は、DB インスタンス上のローカル管理エージェントです。これらの接続は外部 SSL 接続ではありません。そのため、PROTOCOL が tcps であり、HOST が 127.0.0.1 ではないリスナーログエントリがある場合、SSL を使用して接続しているアプリケーションがあります。

リスナーログをチェックするには、ログを Amazon CloudWatch Logs に発行します。詳細については、「Amazon CloudWatch Logs への Oracle ログの発行」を参照してください。

アプリケーション信頼ストアの更新

SSL/TLS 接続に SQL*Plus または JDBC を使用するアプリケーションの信頼ストアを更新できます。

SQL*Plus のアプリケーション信頼ストアの更新

SSL/TLS 接続に SQL*Plus を使用するアプリケーションの信頼ストアを更新できます。

> **Note**
>
> 信頼ストアを更新するとき、新しい証明書を追加できるだけでなく、古い証明書を保持できます。

SQL*Plus アプリケーションの信頼ストアを更新するには

1. すべての AWS リージョンで動作する新しいルート証明書をダウンロードし、そのファイルを `ssl_wallet` ディレクトリに置きます。

   ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。
2. 以下のコマンドを実行して Oracle Wallet を更新します。

   ```
   prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
         $ORACLE_HOME/ssl_wallet/ssl-cert.pem -auto_login_only
   ```

   ファイル名を、ダウンロードしたファイル名に置き換えます。
3. 次のコマンドを実行して、ウォレットが正しく更新されていることを確認します。

   ```
   prompt>orapki wallet display -wallet $ORACLE_HOME/ssl_wallet
   ```

   出力には次の内容が含まれている必要があります。

   ```
   Trusted Certificates:
   Subject: CN=Amazon RDS Root 2019 CA,OU=Amazon RDS,O=Amazon Web Services\,
    Inc.,L=Seattle,ST=Washington,C=US
   ```

JDBC のアプリケーション信頼ストアの更新

SSL/TLS 接続に JDBC を使用するアプリケーションの信頼ストアを更新できます。

ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

証明書をインポートするサンプルスクリプトについては、証明書を信頼ストアにインポートするためのサンプルスクリプト を参照してください。

SSL 接続を確立するための Java コードの例

次のコード例は、JDBC を使用する SSL 接続のセットアップ方法を示します。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "<dns-name-provided-by-amazon-rds>";
    private static final String SSL_PORT = "<ssl-option-port-configured-in-option-
group>";
    private static final String DB_SID = "<oracle-sid>";
    private static final String DB_USER = "<user name>";
    private static final String DB_PASSWORD = "<password>";
    // This key store has only the prod root ca.
    private static final String KEY_STORE_FILE_PATH = "<file-path-to-keystore>";
    private static final String KEY_STORE_PASS = "<keystore-password>";

    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
                "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=
%s))(CONNECT_DATA=(SID=%s)))",
                DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
 properties);
        // If no exception, that means handshake has passed, and an SSL connection can
 be opened
    }
}
```

> **Important**
>
> データベース接続で SSL/TLS を使用することを決定し、アプリケーションの信頼ストアを更新したら、rds-ca-rsa2048-g1 証明書を使用するようにデータベースを更新できます。ステップについては、「DB インスタンスまたはクラスターの変更による CA 証明書の更新」のステップ 3 を参照してください。

RDS for Oracle DB インスタンスでネイティブネットワークの暗号化を使用する

Oracle Database には、ネットワークのデータを暗号化する方法として、ネイティブネットワーク暗号化 (NNE) と Transport Layer Security (TLS) の 2 つがあります。NNE は Oracle 独自のセキュリティ機能ですが、TLS は業界標準です。RDS for Oracle では、Oracle Database のすべてのエディションの NNE がサポートされています。

NNE には TLS と比較して次のような利点があります。

- NNE オプションの設定を使用して、クライアントとサーバーの NNE を制御できます。
  - `SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS` および `SQLNET.ALLOW_WEAK_CRYPTO`
  - `SQLNET.CRYPTO_CHECKSUM_CLIENT` および `SQLNET.CRYPTO_CHECKSUM_SERVER`
  - `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` および `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER`
  - `SQLNET.ENCRYPTION_CLIENT` および `SQLNET.ENCRYPTION_SERVER`
  - `SQLNET.ENCRYPTION_TYPES_CLIENT` および `SQLNET.ENCRYPTION_TYPES_SERVER`
- ほとんどの場合、クライアントやサーバーを設定する必要はありません。対照的に、TLS では、クライアントとサーバーの両方を設定する必要があります。
- 証明書は必要ありません。TLS では、サーバーには証明書が必要です (最終的に期限切れになります)。クライアントには、サーバーの証明書を発行した認証局の信頼できるルート証明書が必要です。

Oracle DB インスタンスの NNE 暗号化を有効にするには、その DB インスタンスに関連付けられているオプショングループに、Oracle NNE オプションを追加します。詳細については、「Oracle ネイティブネットワーク暗号化」を参照してください。

> **Note**
>
> NNE と TLS の両方を同じ DB インスタンスで使用することはできません。

Amazon RDS for Oracle の Kerberos 認証の設定

Kerberos 認証を使用して、ユーザーが Amazon RDS for Oracle DB インスタンスに接続する場合に、ユーザーを認証できます。この設定では、DB インスタンスは AWS Directory Service for Microsoft Active Directory と連携します (AWS Managed Microsoft AD とも言う)。ユーザーが、信頼性の高いドメインに接続された RDS for Oracle DB インスタンスを使用して認証を実行すると、Directory Service を使用して作成したディレクトリに認証リクエストが転送されます。

同じディレクトリにすべての認証情報を保持することで時間と労力を節約できます。複数のデータベースインスタンスの認証情報を保存し、管理する一元的な場所が用意されています。ディレクトリを使用することで、セキュリティプロファイル全体を向上することもできます。

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。Kerberos 認証を使用した RDS for Oracle のバージョンとリージョンの可用性の詳細については、「Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン」を参照してください。

> **Note**
>
> Kerberos 認証は、RDS for Oracle DB インスタンスに廃止された DB インスタンスクラスにはサポートされていません。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。

トピック

- Oracle DB インスタンス用に Kerberos を設定する
- ドメインの DB インスタンスの管理
- Oracle を Kerberos 認証に接続する

Oracle DB インスタンス用に Kerberos を設定する

AWS Directory Service for Microsoft Active Directory とも呼ばれる AWS Managed Microsoft AD を使用し、Oracle DB インスタンスに Kerberos 認証をセットアップします。Kerberos 認証をセットアップするには、以下のステップを完了します。

- ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する
- ステップ 2: 信頼関係を作成する
- ステップ 3: Amazon RDS の IAM のアクセス許可を設定する
- ステップ 4: ユーザーを作成して設定する
- ステップ 5: ディレクトリと DB インスタンスの間のクロス VPC トラフィックを有効にする
- ステップ 6: Oracle DB インスタンスを作成または変更する
- ステップ 7: Kerberos 認証 Oracle ログインを作成する
- ステップ 8: Oracle クライアントを設定する

> **Note**
>
> セットアップ中、RDS は、*managed_service_user*@*example.com* という名前の Oracle データベースユーザーを作成し、CREATE SESSION 権限を付与します (*example.com* はドメイン名です)。このユーザーは、Directory Service が管理されたアクティブディレクトリ内に作成するユーザーに対応します。RDS は定期的に、Directory Service によって提供される認証情報を使用して Oracle データベースにログインします。その後、RDS はすぐにチケットキャッシュを破棄します。

ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する

Directory Service はフルマネージド型の Active Directory を AWS クラウド内に作成します。AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって 2 つのドメインコントローラーと 2 つのドメインネームシステム (DNS) サーバーを作成します。ディレクトリサーバーは、VPC 内の異なるサブネットで作成されます。この冗長性によって、障害が発生してもディレクトリにアクセス可能な状態を維持できます。

AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service がユーザーに代わって自動的に以下のタスクを実行します。

- VPC 内に Active Directory を設定します。
- 「Admin」のユーザー名と指定されたパスワードを使用して、ディレクトリ管理者アカウントを作成します。このアカウントを使用してディレクトリを管理します。

> **Note**
>
> このパスワードは必ず保存してください。Directory Service は保存しません。パスワードはリセットできますが、取得することはできません。

- ディレクトリコントローラー用セキュリティグループを作成します。

AWS Managed Microsoft AD を起動すると、AWS は組織単位 (OU) を作成します。OU にはディレクトリのオブジェクトがすべて含まれています。この OU には、ディレクトリの作成時に入力した NetBIOS 名がドメインルートにあります。ドメインルートは AWS が所有し、管理します。

AWS Managed Microsoft AD ディレクトリに作成した管理者アカウントには、OU に関する以下の代表的な管理業務用のアクセス権限があります。

- ユーザーを作成、更新、削除する
- ファイルやプリントサーバーなどのドメインにリソースを追加して、追加したリソースへのアクセス許可を OU のユーザーとグループに割り当てる
- 追加の OU やコンテナを作成する
- 権限を委譲する
- 削除されたオブジェクトを Active Directory のごみ箱から元に戻す
- Active Directory Web Service で AD と DNS Windows PowerShell モジュールを実行する

管理者アカウントには、ドメイン全体に関する以下のアクティビティを実行する権限もあります。

- DNS 設定 (レコード、ゾーン、フォワーダーの追加、削除、更新) を管理する
- DNS イベントログを参照する
- セキュリティイベントログを参照する

AWS マネジメントコンソール、AWS CLI、Directory Service API を使用して、ディレクトリを作成します。ディレクトリが Oracle DB インスタンスと通信できるように、ディレクトリセキュリティグループで関連するアウトバウンドポートを必ず開いてください。

AWS Managed Microsoft AD でディレクトリを作成するには

1. AWS マネジメントコンソール にサインインし、Directory Service コンソール (https://console.aws.amazon.com/directoryservicev2/)を開きます。
2. ナビゲーションペインで、[Directories]、[Set up Directory] の順に選択します。
3. [AWS Managed Microsoft AD] を選択します。現状では、AWS Managed Microsoft AD が Amazon RDS で使用できる唯一のオプションです。
4. 次の情報を入力します。

   ディレクトリの DNS 名

   ディレクトリの完全修飾名 (例: **corp.example.com**)。

   ディレクトリの NetBIOS 名

   ディレクトリの短縮名 (例: **CORP**)。

   ディレクトリの説明

   (オプション) ディレクトリの説明。

   Admin パスワード

   ディレクトリ管理者のパスワードです。ディレクトリの作成プロセスでは、ユーザー名「Admin」とこのパスワードを使用して管理者アカウントが作成されます。

   ディレクトリ管理者のパスワードには、「admin」の単語を含めることはできません。パスワードは大文字と小文字を区別し、8-64 文字にします。また、以下の 4 つのカテゴリうち 3 つから少なくとも 1 文字を含める必要があります。

   - 小文字 (a ~ z)
   - 大文字 (A ~ Z)
   - 数字 (0 ~ 9)
   - アルファベット以外の文字 (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   パスワードを確認

   管理者のパスワードをもう一度入力します。
5. [Next (次へ)] を選択します。
6. [Networking] セクションに次の情報を入力し、[Next] を選択します。

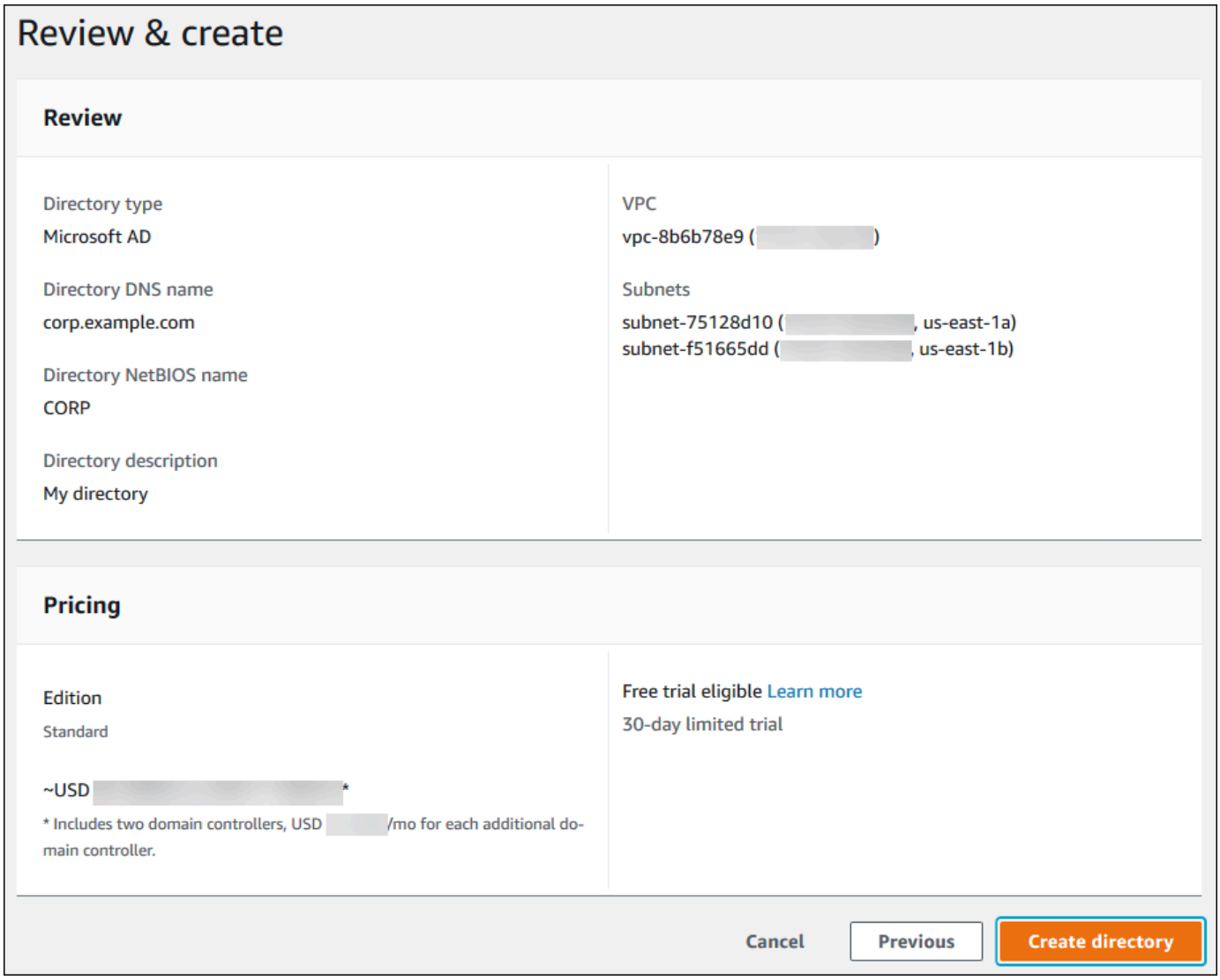
VPC

ディレクトリ用の VPC。この同じ VPC 内に Oracle DB インスタンスを作成します。

Subnets

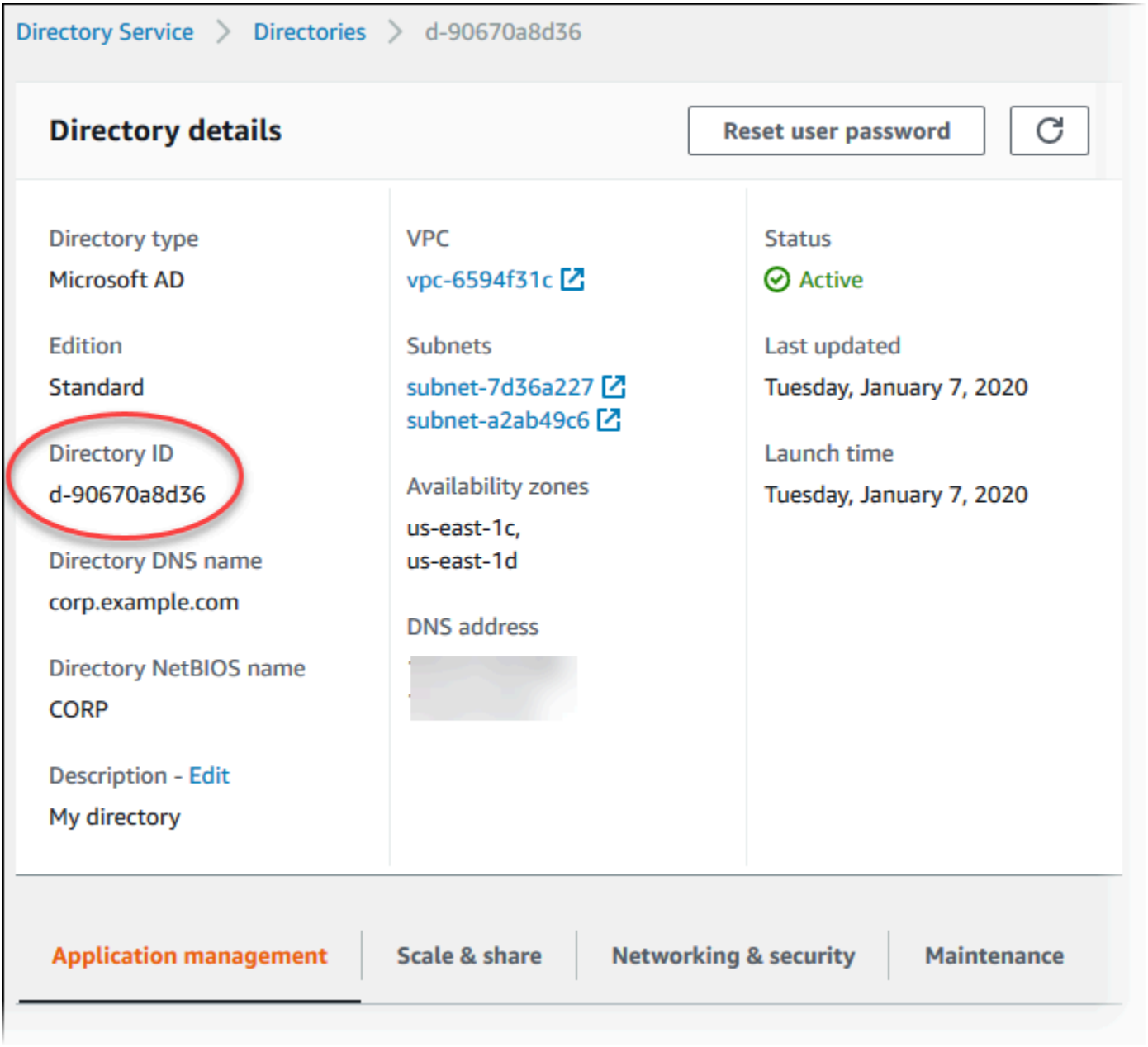
ディレクトリサーバーのサブネット。2 つのサブネットは、異なるアベイラビリティーゾーンに存在している必要があります。

7. ディレクトリ情報を確認し、必要な変更を加えます。情報が正しい場合は、[Create directory (ディレクトリの作成)] を選択します。



ディレクトリが作成されるまで、数分かかります。正常に作成されると、[Status] 値が [Active] に変わります。

ディレクトリに関する情報を表示するには、ディレクトリの一覧で、ディレクトリ名を選択します。[Directory ID] の値を書き留めます。この値は、Oracle DB インスタンスを作成または変更するときに必要になります。



ステップ 2: 信頼関係を作成する

AWS Managed Microsoft AD のみを使用する予定の場合は、ステップ 3: Amazon RDS の IAM のアクセス許可を設定する に進みます。

セルフマネージド Active Directory を使用して Kerberos 認証を有効にするには、セルフマネージド Active Directory と前の手順で作成した AWS Managed Microsoft AD との間にフォレスト信頼関

係を作成する必要があります。信頼は一方向で、AWS Managed Microsoft AD はセルフマネージド Active Directory を信頼します。信頼は、両方の Active Directory が相互に信頼する双方向にすることもできます。Directory Service を使用してフォレストの信頼関係を設定する方法の詳細については、Directory Service 管理ガイドの「信頼関係を作成する場合」を参照してください。

ステップ 3: Amazon RDS の IAM のアクセス許可を設定する

Directory Service を呼び出すには、Amazon RDS にマネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する IAM ロールが必要です。このロールにより、Amazon RDS は Directory Service への呼び出しを行うことができます。

> **Note**
>
> ロールによるアクセスを許可するには、AWS Security Token Service (AWS STS) エンドポイントを AWS アカウント の AWS リージョン でアクティベートする必要があります。AWS STS エンドポイントはすべての AWS リージョン でデフォルトでアクティブになっているため、他のアクションを実行せずに、エンドポイントを使用することができます。詳細については、「IAM ユーザーガイド」の「AWS リージョンでの AWS STS のアクティブ化と非アクティブ化」を参照してください。

IAM ロールの作成

AWS マネジメントコンソール を使用して DB インスタンスを作成し、コンソールユーザーが `iam:CreateRole` アクセス許可を持っている場合、コンソールは `rds-directoryservice-kerberos-access-role` を自動的に作成します。それ以外の場合は、IAM ロールを手動で作成する必要があります。IAM ロールを手動で作成する場合、[`Directory Service`] を選択し、それに AWS マネージドポリシー `AmazonRDSDirectoryServiceAccess` をアタッチします。

サービス用の IAM ロールを作成する方法の詳細については、IAM ユーザーガイドの「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Note**
>
> RDS for Microsoft SQL Server の RDS 用の Windows 認証に使用される IAM ロールは、Oracle 用の RDS には使用できません。

IAM 信頼ポリシーを手動で作成する

マネージド IAM ポリシー AmazonRDSDirectoryServiceAccess を使用する代わりに、必要なアクセス許可を使用してリソースポリシーを作成することもできます。プリンシパルとして directoryservice.rds.amazonaws.com と rds.amazonaws.com の両方を指定します。

特定のリソースへのアクセスについて、Amazon RDS が別のサービスに付与する許可を制限する場合は、リソースポリシー内で aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用することをお勧めします。混乱した代理問題から保護するための最も効果的な方法は、Amazon RDS リソースの完全な ARN を指定しながら、aws:SourceArn グローバル条件コンテキストキーを使用することです。詳細については、「サービス間での混乱した代理問題の防止」を参照してください。

次の例では、Amazon RDS で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、「混乱した代理」問題を回避する方法を示します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
```

```
}
```

オプトインリージョンの場合は、リージョンのサービスプリンシパルを directoryservice.rds.*region_name*.amazonaws.com の形式で含める必要もあります。例えば、アフリカ (ケープタウン) リージョンの場合、次の信頼ポリシーを使用します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "directoryservice.rds.af-south-1.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:af-south-1:123456789012:db:mydbinstance"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

また、ロールには、以下の IAM ポリシーも必要です。

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

ステップ 4: ユーザーを作成して設定する

アクティブディレクトリドメインサービスおよびアクティブディレクトリライトウェイトディレクトリサービスのツールの一部である「アクティブディレクトリユーザーとコンピュータ」ツールを使用して、ユーザーを作成できます。この場合、ユーザーは、ディレクトリにアクセスできる個別の人またはエンティティを表します。

Directory Service ディレクトリにユーザーを作成するには、Directory Service ディレクトリのメンバーである Windows ベースの Amazon EC2 インスタンスに接続している必要があります。同時に、ユーザーを作成する権限を持つユーザーとしてログインしていなければなりません。Active Directory にユーザーを作成する方法の詳細については、 AWS Managed Microsoft AD管理ガイドの「Directory Serviceでユーザーとグループを管理する」を参照してください。

ステップ 5: ディレクトリと DB インスタンスの間のクロス VPC トラフィックを有効にする

同じ VPC 内にディレクトリおよび DB インスタンスを配置する場合は、このステップをスキップして ステップ 6: Oracle DB インスタンスを作成または変更する に進みます。

ディレクトリと DB インスタンスを別の AWS アカウントまたは VPC に配置する場合は、VPC ピア接続または AWS Transit Gateway を使用してクロス VPC トラフィックを設定します。次の手順では、VPC ピア接続を使用して VPC 間のトラフィックを有効にします。Amazon Virtual Private Cloud ピアリング接続ガイドの「VPC ピア機能とは」の手順に従います。

VPC ピア接続を使用してクロス VPC トラフィックを有効にするには

1. 適切な VPC ルーティングを設定し、ネットワークトラフィックが双方向にフローするようにします。
2. DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからのインバウンドトラフィックを受信できることを確認します。詳細については、AWS Managed Microsoft AD 管理ガイドの「Directory Service のベストプラクティス」を参照してください。
3. トラフィックをブロックするネットワークのアクセスコントロールリスト (ACL) ルールがないことを確認します。

別の AWS アカウントがディレクトリを所有している場合は、ディレクトリを共有する必要があります。

AWS アカウント間でディレクトリを共有するには

1. DB インスタンスを作成する AWS アカウントとの間でディレクトリの共有を開始するには、AWS Managed Microsoft AD 管理ガイドの「チュートリアル: Directory Service ディレクトリを共有して、シームレスに EC2 ドメインを結合する」の手順を実行します。
2. DB インスタンスのアカウントを使用して、Directory Service コンソールにサインインし、続行する前にドメインが必ず SHARED ステータスであることを確認します。
3. DB インスタンスのアカウントを使用して Directory Service コンソールにサインインしている間に、[ディレクトリ ID] の値を書き留めておきます。このディレクトリ ID は、DB インスタンスをドメインに結合するために使用します。

ステップ 6: Oracle DB インスタンスを作成または変更する

ディレクトリで使用する Oracle DB インスタンスを作成または変更します。コンソール、CLI、RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これには以下の 2 つの方法があります。

- コンソール、create-db-instance CLI コマンド、CreateDBInstance RDS API オペレーションを使用して新しい Oracle DB インスタンスを作成します。

  手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance CLI コマンド、ModifyDBInstance RDS API オペレーションを使用して、既存の Oracle DB インスタンスを変更します。

手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。

- コンソール、restore-db-instance-from-db-snapshot CLI コマンド、RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから Oracle DB インスタンスを復元します。

  手順については、「DB インスタンスへの復元」を参照してください。

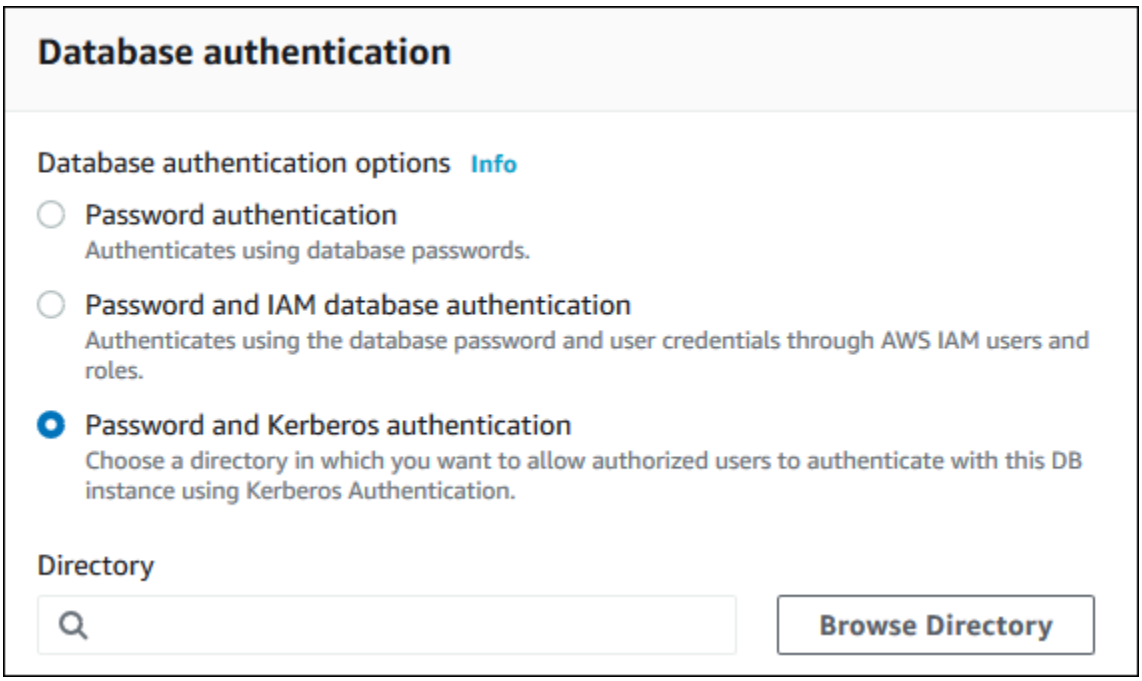
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、Oracle DB インスタンスをポイントインタイムに復元します。

  手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

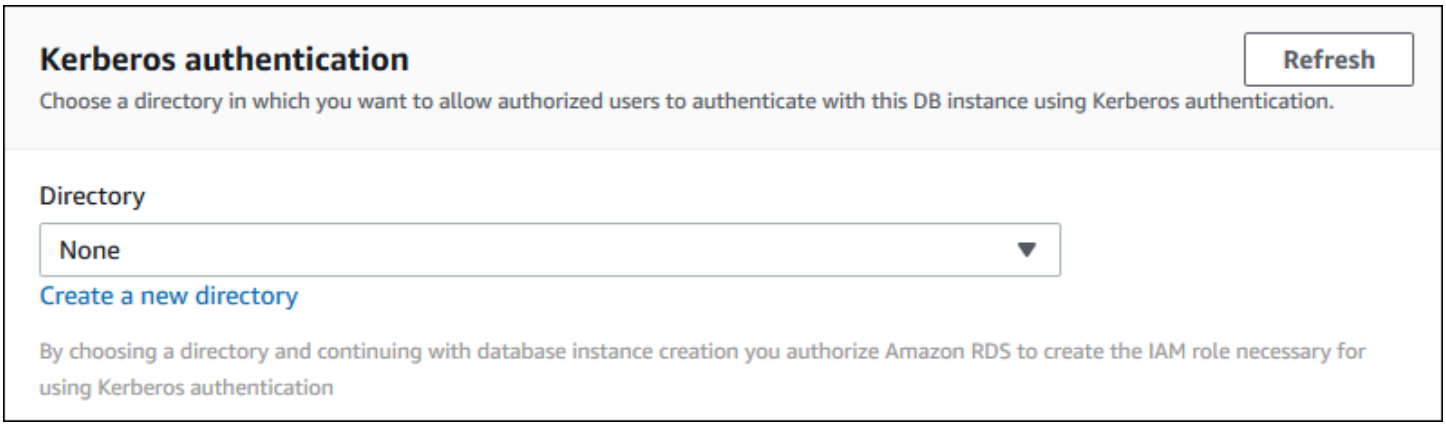
Kerberos 認証は、VPC 内の Oracle DB インスタンスにのみサポートされています。DB インスタンスは、ディレクトリと同じ VPC または異なる VPC 内にあります。DB インスタンスの作成または変更時に、次の手順を行います。

- ディレクトリの作成時に、生成されたドメイン識別子 (d-* 識別子) を指定します。
- 作成した IAM ロール名を指定します。
- DB インスタンスセキュリティグループがディレクトリセキュリティグループからインバウンドトラフィックを受信し、ディレクトリへのアウトバウンドトラフィックを送信できることを確認してください。

DB インスタンスを作成するためにコンソールを使用する場合は、データベースの認証 セクションの [パスワードと Kerberos 認証] を選択します。[ディレクトリの参照] を選択してディレクトリを選択するか、[新しいディレクトリの作成] を選択します。



コンソールを使用して DB インスタンスを変更または復元する場合は、[Kerberos 認証] セクションでディレクトリを選択するか、[Create a new directory (新しいディレクトリの作成)] を選択します。



AWS CLI を使用する場合は、DB インスタンスが、作成したディレクトリを使用できるように、以下のパラメータが必要です。

- `--domain` パラメータには、ディレクトリの作成時に生成されたドメイン識別子 ("d-*" 識別子) を使用します。
- `--domain-iam-role-name` パラメータには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する作成済みのロールを使用します。

例えば、以下の CLI コマンドはディレクトリを使用するように DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

Important

Kerberos 認証を有効化するために DB インスタンスを変更する場合、変更後 DB インスタンスを再起動します。

Note

MANAGED_SERVICE_USER は、RDS の Directory Service によってランダムに生成される名前を持つサービスアカウントです。Kerberos 認証のセットアップ時に、RDS for Oracle は同じ名前のユーザーを作成し、そのユーザーに CREATE SESSION 権限を割り当てます。Oracle DB ユーザーは、外部で *MANAGED_SERVICE_USER@EXAMPLE.COM* と識別されます。*EXAMPLE.COM* はドメインの名前です。RDS は定期的に、Directory Service によって提供される認証情報を使用して Oracle データベースにログインします。その後、RDS はすぐにチケットキャッシュを破棄します。

ドメイン参加が成功したことを確認するには、次のコマンドを実行します。

```
aws rds describe-db-instances --db-instance-identifier mydbinstance --query
 'DBInstances[0].DomainMemberships'
```

Status が joined と表示されていることを確認します。failed と表示される場合は、セキュリティグループが DB インスタンスとディレクトリサービス間のトラフィックを許可していることを確認します。

ステップ 7: Kerberos 認証 Oracle ログインを作成する

Amazon RDS マスターユーザーの認証情報を使用して、他の DB インスタンスと同様に Oracle DB インスタンスに接続します。DB インスタンスは、AWS Managed Microsoft AD ドメインに接続されています。このように、ドメインの Microsoft Active Directory ユーザーおよびから Oracle ログインおよびユーザーのプロビジョニングができます。標準の Oracle 権限をこれらのログインに付与したり、取り消したりしてデータベースの権限を管理します。

Microsoft Active Directory のユーザーに Oracle 認証を許可するには

1. Amazon RDS マスターユーザー認証情報を使用して、Oracle DB インスタンスに接続します。
2. 外部認証されたユーザーを Oracle データベース内に作成します。

   次の例では、*KRBUSER@CORP.EXAMPLE.COM* をユーザ名とドメイン名に置き換えます。

   ```
   CREATE USER "KRBUSER@CORP.EXAMPLE.COM" IDENTIFIED EXTERNALLY;
   GRANT CREATE SESSION TO "KRBUSER@CORP.EXAMPLE.COM";
   ```

   これでドメインのユーザー (人とアプリケーションの両方) は、Kerberos 認証を使用してドメインに参加しているクライアントマシンから Oracle DB インスタンスに接続できます。

ステップ 8: Oracle クライアントを設定する

Oracle クライアントを構成するには、次の要件を満たす必要があります。

- ドメインを指すように、krb5.conf (Linux) または krb5.ini (Windows) という名前の設定ファイルを作成します。この設定ファイルを使用する Oracle クライアントを設定します。
- TCP/UDP の 53 番ポート (DNS)、Kerberos ポート(マネージド型 Directory Service 向けの 88 番および 464 番)、および TCP の389 番 LDAP ポートを介して、クライアントホストと Directory Service の間でトラフィックが流れているかを確認します。
- データベースポートを介してクライアントホストと DB インスタンス間でトラフィックが流れることを確認します。

以下は、AWS Managed Microsoft AD のサンプルコンテンツです。

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
```

```
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

以下は、オンプレミス Microsoft AD のサンプルコンテンツです。krb5.conf ファイルまたは krb5.ini ファイルで、*on-prem-ad-server-name* をオンプレミス AD サーバー名に置き換えます。

```
[libdefaults]
 default_realm = ONPREM.COM
[realms]
 AWSAD.COM = {
  kdc = awsad.com
  admin_server = awsad.com
 }
 ONPREM.COM = {
  kdc = on-prem-ad-server-name
  admin_server = on-prem-ad-server-name
 }
[domain_realm]
 .awsad.com = AWSAD.COM
 awsad.com= AWSAD.COM
 .onprem.com = ONPREM.COM
 onprem.com= ONPREM.COM
```

> **Note**
>
> krb5.ini ファイル または krb5.conf ファイルを設定したら、サーバーを再起動することをお勧めします。

以下は、SQL*Plus 構成向けの sqlnet.ora の内容のサンプルです。

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5PRE,KERBEROS5)
SQLNET.KERBEROS5_CONF=path_to_krb5.conf_file
```

SQL 開発者向け構成の例に関しては、Oracle サポートの「ドキュメント 1609359.1」をご参照ください。

ドメインの DB インスタンスの管理

コンソール、CLI、RDS API を使用して、DB インスタンスと Microsoft Active Directory との関係を管理できます。例えば、Kerberos 認証を有効化するために、Microsoft Active Directory を関連付けることができます。また、Kerberos 認証を無効化するために、Microsoft Active Directory を関連付けを解除することができます。さらに、1 つの Microsoft Active Directory によって外部に認証される DB インスタンスをもう 1 つの Microsoft Active Directory に移動することもできます。

例えば、CLI を使用して次を実行できます。

- メンバーシップが失敗認めに Kerberos 認証を再度有効化することを試みるには、modify-db-instance CLI コマンドを使用し、`--domain` オプションの現在のメンバーシップのディレクトリ ID を指定します。
- DB インスタンスの Kerberos 認証を無効にするには、modify-db-instance CLI コマンドを使用して、`none` オプションに `--domain` を指定します。
- 1 つのドメインから他のドメインに DB インスタンスを移動するには、modify-db-instance CLI コマンドを使用して、`--domain` オプションに新しいドメインのドメイン識別子を指定します。

ドメインメンバーシップ状態の表示

DB インスタンスを作成または変更した後で、その DB インスタンスは、ドメインのメンバーになります。DB インスタンスのドメインメンバーシップのステータスは、コンソールで表示したり、describe-db-instances CLI コマンドを実行して表示したりすることができます。DB インスタンスのステータスは、以下のいずれかです。

- `kerberos-enabled` - DB インスタンスは Kerberos 認証を有効化しました。
- `enabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を有効化中です。
- `pending-enable-kerberos` - この DB インスタンスでは、Kerberos 認証の有効化が保留中になっています。
- `pending-maintenance-enable-kerberos` - AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の有効化を試みます。
- `pending-disable-kerberos` - この DB インスタンスでは、Kerberos 認証の無効化が保留中になっています。
- `pending-maintenance-disable-kerberos` – AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の無効化を試みます。

- `enable-kerberos-failed` – 設定の問題により、AWS は DB インスタンス上の Kerberos 認証を有効化できませんでした。DB インスタンスを変更するコマンドを再発行する前に、設定の問題を修正します。
- `disabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を無効化中です。

ネットワーク接続の問題や正しくない IAM ロールのために、Kerberos 認証を有効化するリクエストは失敗する可能性があります。DB インスタンスを作成または変更する際に Kerberos 認証を有効化する試みが失敗した場合は、正しい IAM ロールを使用していることを確認してください。次に、DB インスタンスを変更し、ドメインに接続します。

> **Note**
>
> Amazon RDS for Oracle を使用する Kerberos 認証でのみ、ドメインの DNS サーバーにトラフィックが送信されます。他のすべての DNS リクエストは、Oracle を実行している DB インスタンスでアウトバウンドのネットワークアクセスとして取り扱われます。Amazon RDS for Oracle を使用するアウトバウンドのネットワークアクセスに関する詳細は、「カスタム DNS サーバーのセットアップ」を参照してください。

Kerberos キーの強制更新

シークレットキーは、AWS Managed Microsoft AD と Amazon RDS for Oracle DB インスタンス間で共有されます。このキーは、45 日ごとに自動で更新されます。このキーを強制的に更新するには、次の Amazon RDS 手順を実行してください。

```
SELECT rdsadmin.rdsadmin_kerberos_auth_tasks.rotate_kerberos_keytab AS TASK_ID FROM
 DUAL;
```

> **Note**
>
> リードレプリカの設定では、ソース DB インスタンスのみでこの手順を実行可能で、リードレプリカでは実行できません。

SELECT ステートメントでは、データ型 VARCHAR2 のタスクの ID が返ります。実行中のタスクのステータスは bdump ファイルで確認できます。bdump ファイルは `/rdsdbdata/log/trace` ディレクトリにあります。bdump ファイルの名前形式は、以下のとおりです。

```
dbtask-task-id.log
```

タスクの出力ファイルを表示すると、結果を確認できます。

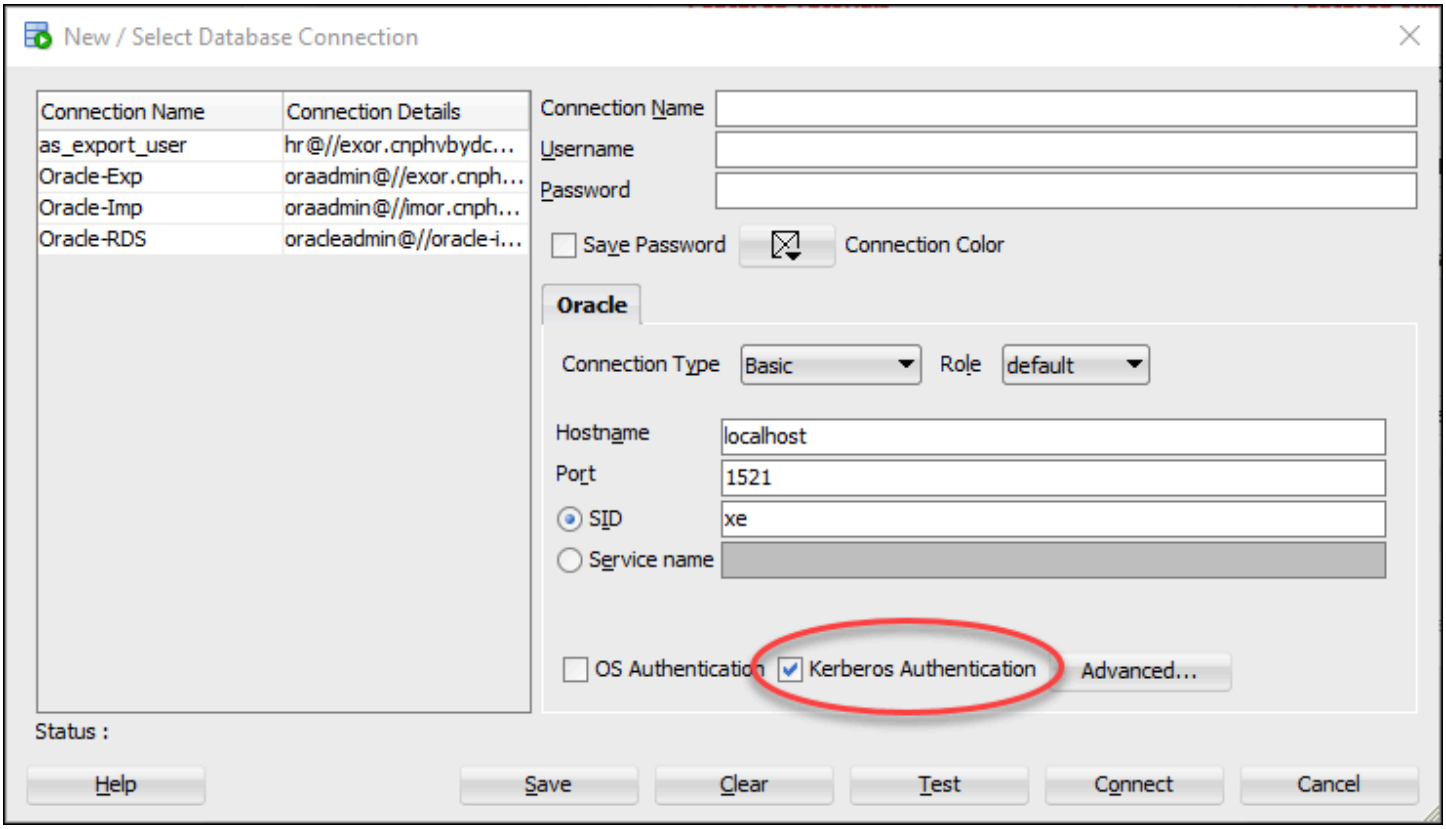
```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-
id.log'));
```

task-id は、この手順で返されたタスク ID に置き換えます。

> **Note**
>
> タスクは非同期的に実行されます。

Oracle を Kerberos 認証に接続する

このセクションは、「ステップ 8: Oracle クライアントを設定する」で説明されているように Oracle クライアントをセットアップ済みであることを前提としています。Kerberos 認証により Oracle DB に接続するには、Kerberos 認証タイプを使用してログインします。例えば、Oracle SQL Developer を起動した後で、以下の例に示すように、[Kerberos 認証] を認証タイプとして選択します。



SQL*Plus で Kerberos 認証を使用して Oracle に接続するには:

1. コマンドプロンプトで、次のコマンドを実行します。

   ```
   kinit username
   ```

   username をユーザー名で置き換え、プロンプトでユーザーに Microsoft Active Directory で保存されているパスワードを入力します。
2. SQL*Plus を開き、DNS の名前および Oracle DB インスタンスのポート番号を使用して接続します。

   SQL*Plus での Oracle DB インスタンスへの接続の詳細については、「SQL *Plus を使用した DB インスタンスへの接続」を参照してください。

> **Tip**
>
> ネイティブ Windows キャッシュを使用している場合は、sqlnet.ora ファイルで SQLNET.KERBEROS5_CC_NAME パラメータを OSMSFT:// または MSLSA に設定して、Microsoft Active Directory に保存されている認証情報を使用することもできます。

証明書と Oracle ウォレットを使用した、UTL_HTTP アクセスの設定

Amazon RDS は RDS for Oracle DB インスタンスでのアウトバウンドネットワークアクセスをサポートします。DB インスタンスをネットワークに接続するには、次の PL/SQL パッケージを使用できます。

UTL_HTTP

このパッケージでは、SQL および PL/SQL から HTTP 呼び出しを実行します。HTTP 経由でインターネット上のデータにアクセスするために使用できます。詳細については、Oracle ドキュメントの「UTL_MAIL」を参照してください。

UTL_TCP

このパッケージは、PL/SQL で TCP/IP クライアント側アクセス機能を提供します。このパッケージは、インターネットプロトコルと E メールを使用する PL/SQL アプリケーションに役立ちます。詳細については、Oracle ドキュメントの「UTL_TCP」を参照してください。

UTL_SMTP

このパッケージは、SMTP コマンドへのインターフェイスを提供します。これにより、クライアントが E メールを SMTP サーバーにディスパッチできるようにします。詳細については、Oracle ドキュメントの「UTL_SMTP」を参照してください。

次のタスクを完了すると、UTL_HTTP.REQUEST を設定できます。これにより、SSL ハンドシェイク中にクライアント認証の証明書を要求するウェブサイトを操作できます。また Oracle ウォレットの生成コマンドや、DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE の手順を修正して、ウェブサイトへの UTL_HTTP アクセスに使用するパスワード認証を設定することもできます。詳細については、Oracle Database のドキュメントの「DBMS_NETWORK_ACL_ADMIN」を参照してください。

> **Note**
>
> UTL_SMTP に合わせ次のタスクを変更して、SSL/TLS 経由でメールを送信できるようにします (Amazon Simple Email Service を含む)。

トピック

- UTL_HTTP アクセスを設定する際の考慮事項
- ステップ 1: ウェブサイトのルート証明書を取得する
- ステップ 2: Oracle ウォレットを作成する
- ステップ 3: RDS for Oracle インスタンスに、Oracle ウォレットをダウンロードする
- ステップ 4: ユーザーに Oracle ウォレットへのアクセス許可を付与する
- ステップ 5: DB インスタンスからウェブサイトへのアクセスを設定する
- ステップ 6: DB インスタンスからウェブサイトへの接続をテストする

UTL_HTTP アクセスを設定する際の考慮事項

アクセスを設定する前に、以下を考慮してください。

- UTL_MAIL オプションで SMTP を使用することができます。詳細については、「Oracle UTL_MAIL」を参照してください。
- リモートホストのドメインネームサーバー (DNS) の名は、以下のいずれかです:
  - パブリックに解決可能。

- Amazon RDS DB インスタンスのエンドポイント。
- 独自のDNS サーバーを介して解決可能。詳細については、「カスタム DNS サーバーのセットアップ」を参照してください。
- 同じ VPC またはピアリング接続先 VPC の Amazon EC2 インスタンスのプライベート DNS 名。この場合、名前がカスタム DNS サーバーを介して解決可能であることを確認してください。また、VPC 設定の `enableDnsSupport` 属性を有効にし、VPC ピア接続の DNS 解決サポートを有効にすることで Amazon が提供する DNS を使用することもできます。詳細については、「VPC の DNS サポート」および「VPC ピア接続の変更」を参照してください。
- リモート SSL/TLS リソースに安全に接続するには、カスタマイズされた Oracle ウォレット を作成し、アップロードすることをお勧めします。Amazon S3 を Amazon RDS for Oracle 機能と統合することで、Amazon S3 から Oracle DB インスタンスにウォレットをダウンロードします。Oracle 用の Amazon S3 の統合については、「「Amazon S3 統合」」を参照してください。

- Oracle SSL オプションがインスタンスごとに構成されている場合は、SSL/TLS エンドポイントを介して Oracle DB インスタンス間にデータベースリンクを確立することができます。必要な設定はこれだけです。詳細については、「Oracle Secure Sockets Layer」を参照してください。

ステップ 1: ウェブサイトのルート証明書を取得する

RDS for Oracle DB インスタンスがウェブサイトへのセキュアな接続を確立するには、ルート CA 証明書を追加します。Amazon RDS はルート証明書を使用して、Oracle ウォレットにウェブサイトの証明書を署名します。

ルート証明書は、さまざまな方法で取得できます。例えば、次のオペレーションを実行できます。

1. ウェブサーバーを使用して、証明書で保護されたウェブサイトにアクセスします。
2. 署名に使われたルート証明書をダウンロードします。

AWS のサービスの場合、ルート証明書は通常、Amazon Trust Services リポジトリにあります。

ステップ 2: Oracle ウォレットを作成する

ウェブサーバー証明書とクライアント認証証明書の両方を含む Oracle ウォレットを作成します。RDS Oracle インスタンスは、ウェブサーバー証明書を使用して、ウェブサイトへの安全な接続を確立します。ウェブサイトでは、Oracle のデータベースユーザーを認証するために、クライアント証明書が必要です。

認証にクライアント証明書を使用せずに、セキュアな接続を構成したい場合があります。この場合、次の手順で Java キーストアのステップを省略できます。

Oracle ウォレットを作成するには

1. ルート証明書とクライアント証明書を 1 つのディレクトリに配置してから、このディレクトリに移動します。
2. .p12 クライアント証明書を、Java キーストアに変換します。

   > **Note**
   >
   > 認証にクライアント証明書を使用していない場合は、このステップを省略できます。

   次の例では、*client_certificate.p12* という名前のクライアント証明書を、*client_keystore.jks* という名前の Java キーストアに変換します。その後、キーストアは Oracle ウォレットに入ります。キーストアのパスワードは *P12PASSWORD* です。

   ```
   orapki wallet pkcs12_to_jks -wallet ./client_certificate.p12 -
   jksKeyStoreLoc ./client_keystore.jks -jksKeyStorepwd P12PASSWORD
   ```
3. 証明書のディレクトリとは別に、Oracle ウォレット用のディレクトリを作成します。

   例えば、次の例では /tmp/wallet ディレクトリを作成します。

   ```
   mkdir -p /tmp/wallet
   ```
4. ウォレットディレクトリに、Oracle ウォレットを作成します。

   次の例では、Oracle ウォレット パスワードを *P12PASSWORD* に設定します。これは、前のステップで Java キーストアで使用したのと同じパスワードです。同じパスワードを使用すると便利ですが、必須ではありません。-auto_login パラメータが自動ログイン機能をオンにするため、アクセスするたびにパスワードを指定する必要はありません。

   > **Note**
   >
   > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

```
orapki wallet create -wallet /tmp/wallet -pwd P12PASSWORD -auto_login
```

5. Java キーストアを、Oracle ウォレットに追加します。

> **Note**
>
> 認証にクライアント証明書を使用していない場合は、このステップを省略できます。

次の例では、*/tmp/wallet* という Oracle ウォレットに、*client_keystore.jks* キーストアを追加します。この例では、Java キーストアと Oracle ウォレットに、同じパスワードを指定します。

```
orapki wallet jks_to_pkcs12 -wallet /tmp/wallet -pwd P12PASSWORD -
keystore ./client_keystore.jks -jkspwd P12PASSWORD
```

6. Oracle ウォレットに、対象のウェブサイトのルート証明書を追加します。

次の例では、*Root_CA.cer* という名前の証明書を追加します。

```
orapki wallet add -wallet /tmp/wallet -trusted_cert -cert ./Root_CA.cer -
pwd P12PASSWORD
```

7. 中間証明書を追加します。

次の例では、*Intermediate.cer* という名前の証明書を追加します。中間証明書をすべてロードするまで、この手順を必要な回数繰り返します。

```
orapki wallet add -wallet /tmp/wallet -trusted_cert -cert ./Intermediate.cer -
pwd P12PASSWORD
```

8. 新しく作成した Oracle ウォレットに、必要な証明書があることを確認します。

```
orapki wallet display -wallet /tmp/wallet -pwd P12PASSWORD
```

ステップ 3: RDS for Oracle インスタンスに、Oracle ウォレットをダウンロードする

このステップでは、まず Oracle ウォレットを Amazon S3 にアップロードし、次に Amazon S3 から RDS for Oracle インスタンスにウォレットをダウンロードします。

RDS for Oracle DB インスタンスに Oracle ウォレットをダウンロードするには

1. Oracle との Amazon S3 統合の前提条件を満たしたら、S3_INTEGRATION オプションを Oracle DB インスタンスに追加します。使用している Amazon S3 バケットへのアクセス権が、オプションの IAM ルールにあることを確認します。

   詳細については、「「Amazon S3 統合」」を参照してください。
2. マスターユーザーとして DB インスタンスにログインし、Oracle ウォレットを保持しておく Oracle ディレクトリを作成します。

   次の例では、*WALLET_DIR* という名前で Oracle のディレクトリを作成しています。

   ```
   EXEC rdsadmin.rdsadmin_util.create_directory('WALLET_DIR');
   ```

   詳細については、「主要データストレージ領域でのディレクトリの作成と削除」を参照してください。
3. 作成した Oracle ウォレットを Amazon S3 バケットにアップロードします。

   サポートされているアップロード方法は、いずれも使用できます。
4. Oracle ウォレットを再度アップロードする場合は、既存のウォレットを削除します。それ以外の場合は、次のステップに進みます。

   次の例では、*cwallet.sso* という名前の、既存のウォレットを削除します。

   ```
   EXEC UTL_FILE.FREMOVE ('WALLET_DIR','cwallet.sso');
   ```
5. Amazon S3 バケットから Oracle DB インスタンスに Oracle ウォレットをダウンロードします。

   次の例では、*cwallet.sso* という名前のウォレットを、*my_s3_bucket* という名前の Amazon S3 バケットから、*WALLET_DIR* という名前の DB インスタンスディレクトリにダウンロードしています。

   ```
   SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
   ```

```
      p_bucket_name    =>  'my_s3_bucket',
      p_s3_prefix      =>  'cwallet.sso',
      p_directory_name =>  'WALLET_DIR')
   AS TASK_ID FROM DUAL;
```

6. (オプション) パスワードで保護された Oracle ウォレットをダウンロードします。

   このウォレットは、ウォレットを使用するたびにパスワードを要求する場合にのみ、ダウンロードしてください。次の例では、パスワードで保護された *ewallet.p12* ウォレットをダウンロードしています。

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'my_s3_bucket',
      p_s3_prefix      =>  'ewallet.p12',
      p_directory_name =>  'WALLET_DIR')
   AS TASK_ID FROM DUAL;
```

7. DB タスクのステータスを確認します。

   次の例では、前のステップで返されたタスク ID を、*dbtask-1234567890123-4567.log* に代入します。

```
SELECT TEXT FROM
 TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1234567890123-4567.log'));
```

8. Oracle ウォレットの保存に使用している、ディレクトリの内容を確認します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'WALLET_DIR'));
```

   詳細については、「DB インスタンスディレクトリ内のファイルのリスト化」を参照してください。

ステップ 4: ユーザーに Oracle ウォレットへのアクセス許可を付与する

データベースのユーザーを新しく作成するか、既存のユーザーを設定できます。いずれの場合も、証明書を使用したセキュアな接続や、クライアント認証のために、Oracle ウォレットにアクセスするようにユーザーを設定する必要があります。

Oracle ウォレットのユーザーアクセス許可を付与するには

1. RDS for Oracle DB インスタンスに、マスターユーザーとしてログインします。

2. 既存のデータベースユーザーを設定しない場合は、新しいユーザーを作成します。それ以外の場合は、次のステップに進みます。

   次の例では、*my-user* という名前のデータベースユーザーを作成します。

   ```
   CREATE USER my-user IDENTIFIED BY my-user-pwd;
   GRANT CONNECT TO my-user;
   ```

3. データベースユーザーに、Oracle ウォレットのあるディレクトリに対するアクセス許可を付与します。

   次の例では、*my-user* ユーザーに対して、*WALLET_DIR* ディレクトリへの読み取りアクセス権限を付与します。

   ```
   GRANT READ ON DIRECTORY WALLET_DIR TO my-user;
   ```

4. データベースユーザーに対して、UTL_HTTP パッケージを使用するアクセス許可を付与します。

   次の PL/SQL プログラムは、*my-user* ユーザーに対して、UTL_HTTP へのアクセス許可を付与します。

   ```
   BEGIN
     rdsadmin.rdsadmin_util.grant_sys_object('UTL_HTTP', UPPER('my-user'));
     END;
   /
   ```

5. データベースユーザーに対して、UTL_FILE パッケージを使用するアクセス許可を付与します。

   次の PL/SQL プログラムは、*my-user* ユーザーに対して、UTL_FILE へのアクセス許可を付与します。

   ```
   BEGIN
     rdsadmin.rdsadmin_util.grant_sys_object('UTL_FILE', UPPER('my-user'));
     END;
   /
   ```

ステップ 5: DB インスタンスからウェブサイトへのアクセスを設定する

このステップでは、UTL_HTTP やアップロードした Oracle ウォレット、クライアント証明書を使用して、対象のウェブサイトに接続できるように、Oracle データベースユーザーを設定します。詳細については、Oracle Database のドキュメントの「Configuring Access Control to an Oracle Wallet」(Oracle Wallet に対するアクセス制御の設定) を参照してください。

RDS for Oracle DB インスタンスからウェブサイトへのアクセスを設定するには

1. RDS for Oracle DB インスタンスに、マスターユーザーとしてログインします。
2. セキュアなポートで、ユーザーと対象のウェブサイトの Host Access Control Entry (ACE) を作成します。

   次の例では、*my-user* ユーザーが、セキュアなポート 443 経由で *secret.encrypted-website.com* にアクセスできるように設定します。

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host       => 'secret.encrypted-website.com',
    lower_port => 443,
    upper_port => 443,
    ace        => xs$ace_type(privilege_list => xs$name_list('http'),
                              principal_name => 'my-user',
                              principal_type => xs_acl.ptype_db));
                           -- If the program unit results in PLS-00201, set
                           -- the principal_type parameter to 2 as follows:
                           -- principal_type => 2));
END;
/
```

> **Important**
>
> 前述のプログラムユニットでは、次のエラーが発生する可能性があります: PLS-00201: identifier 'XS_ACL' must be declared。このエラーが返された場合は、principal_type に値を割り当てる行を次の行に置き換えてから、プログラムユニットを再実行します。
>
> ```
> principal_type => 2));
> ```

PL/SQL パッケージ XS_ACL の定数の詳細については、Oracle Database ドキュメントの「Real Application Security Administrator's and Developer's Guide」を参照してください。

詳細については、Oracle Database のドキュメントの「Configure Access Control Lists (ACLs) to External Network Services」(外部ネットワークサービスへのアクセス制御リスト (ACL) の構成) を参照してください。

3. (オプション) 標準のポートで、ユーザーと対象のウェブサイトの ACE を作成します。

   一部のウェブページが、セキュアポート (443) ではなく標準のウェブサーバーのポート (80) から提供される場合は、標準のポートを使用する必要がある場合があります。

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host       => 'secret.encrypted-website.com',
    lower_port => 80,
    upper_port => 80,
    ace        => xs$ace_type(privilege_list => xs$name_list('http'),
                              principal_name => 'my-user',
                              principal_type => xs_acl.ptype_db));
                           -- If the program unit results in PLS-00201, set
                           -- the principal_type parameter to 2 as follows:
                           -- principal_type => 2));
END;
/
```

4. アクセスコントロールエントリが存在することを確認します。

```
SET LINESIZE 150
COLUMN HOST FORMAT A40
COLUMN ACL FORMAT A50

SELECT HOST, LOWER_PORT, UPPER_PORT, ACL
  FROM DBA_NETWORK_ACLS
ORDER BY HOST;
```

5. データベースユーザーに対して、UTL_HTTP パッケージを使用するアクセス許可を付与します。

次の PL/SQL プログラムは、*my-user* ユーザーに対して、UTL_HTTP へのアクセス許可を付与します。

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_HTTP', UPPER('my-user'));
  END;
/
```

6. 関連するアクセスコントロールリストが存在することを確認します。

```
SET LINESIZE 150
COLUMN ACL FORMAT A50
COLUMN PRINCIPAL FORMAT A20
COLUMN PRIVILEGE FORMAT A10

SELECT ACL, PRINCIPAL, PRIVILEGE, IS_GRANT,
       TO_CHAR(START_DATE, 'DD-MON-YYYY') AS START_DATE,
       TO_CHAR(END_DATE, 'DD-MON-YYYY') AS END_DATE
  FROM DBA_NETWORK_ACL_PRIVILEGES
ORDER BY ACL, PRINCIPAL, PRIVILEGE;
```

7. クライアント認証で証明書を使用し、接続で Oracle ウォレットを使用するためのアクセス許可を、データベースユーザーに付与します。

> **Note**
>
> 認証にクライアント証明書を使用していない場合は、このステップを省略できます。

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
BEGIN
  SELECT DIRECTORY_PATH
    INTO l_wallet_path
    FROM ALL_DIRECTORIES
   WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
  DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE(
    wallet_path => 'file:/' || l_wallet_path,
    ace         =>  xs$ace_type(privilege_list => xs
$name_list('use_client_certificates'),
                                principal_name => 'my-user',
```

```
                                    principal_type => xs_acl.ptype_db));
END;
/
```

ステップ 6: DB インスタンスからウェブサイトへの接続をテストする

このステップでは、UTL_HTTP やアップロードした Oracle ウォレット、クライアント証明書を使用してウェブサイトに接続できるように、データベースユーザーを設定します。

RDS for Oracle DB インスタンスからウェブサイトへのアクセスを設定するには

1. RDS for Oracle DB インスタンスに、UTL_HTTP へのアクセス許可を持つデータベースユーザーとしてログインします。
2. 対象のウェブサイトへ接続する際に、ホストアドレスを解決できることを確認します。

   次の例では、*secret.encrypted-website.com* からホストアドレスを取得します。

   ```
   SELECT UTL_INADDR.GET_HOST_ADDRESS(host => 'secret.encrypted-website.com')
     FROM DUAL;
   ```
3. 失敗した接続をテストします。

   UTL_HTTP は、証明書とともに Oracle ウォレットの場所を要求するため、次のクエリは失敗します。

   ```
   SELECT UTL_HTTP.REQUEST('https://secret.encrypted-website.com') FROM DUAL;
   ```
4. UTL_HTTP.SET_WALLET を使用し、DUAL から選択した場合の、ウェブサイトへのアクセスをテストします。

   ```
   DECLARE
     l_wallet_path all_directories.directory_path%type;
   BEGIN
     SELECT DIRECTORY_PATH
       INTO l_wallet_path
       FROM ALL_DIRECTORIES
      WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
     UTL_HTTP.SET_WALLET('file:/' || l_wallet_path);
   END;
   /
   ```

```
SELECT UTL_HTTP.REQUEST('https://secret.encrypted-website.com') FROM DUAL;
```

5. (オプション) クエリを変数に保存し、EXECUTE IMMEDIATE を使用した場合の、ウェブサイトへのアクセスをテストします。

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
  v_webpage_sql VARCHAR2(1000);
  v_results     VARCHAR2(32767);
BEGIN
  SELECT DIRECTORY_PATH
    INTO l_wallet_path
    FROM ALL_DIRECTORIES
   WHERE UPPER(DIRECTORY_NAME)='WALLET_DIR';
  v_webpage_sql := 'SELECT UTL_HTTP.REQUEST(''secret.encrypted-website.com'', '''',
 ''file:/' ||l_wallet_path||''') FROM DUAL';
  DBMS_OUTPUT.PUT_LINE(v_webpage_sql);
  EXECUTE IMMEDIATE v_webpage_sql INTO v_results;
  DBMS_OUTPUT.PUT_LINE(v_results);
END;
/
```

6. (オプション) ファイルシステム上で、Oracle ウォレットディレクトリの場所を検索します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'WALLET_DIR'));
```

前のコマンドからの出力を使用して、HTTP リクエストを作成します。例えば、ディレクトリが *rdsdbdata/userdirs/01* である場合、次のクエリを実行します。

```
SELECT UTL_HTTP.REQUEST('https://secret.encrypted-website.com/', '',
 'file://rdsdbdata/userdirs/01')
FROM   DUAL;
```

RDS for Oracle で CDB を使用する

Oracle のマルチテナントアーキテクチャでは、コンテナデータベース (CDB) にカスタマーが作成したプラグイン可能なデータベース (PDB) を含めることができます。CDB の詳細については、Oracle Database ドキュメントの「マルチテナントアーキテクチャの概要」を参照してください。

トピック

- RDS for Oracle CDB の概要
- RDS for Oracle CDB の設定
- CDB のバックアップと復元
- RDS for Oracle の 非 CDB から CDB への変換
- シングルテナント設定からマルチテナント設定への変換
- RDS for Oracle テナントデータベースを CDB インスタンスに追加する
- RDS for Oracle テナントデータベースの変更
- RDS for Oracle テナントデータベースを CDB から削除する
- テナントデータベースの詳細を表示する
- CDB のアップグレード

RDS for Oracle CDB の概要

Oracle Database 19c 以降を実行している場合、RDS for Oracle DB インスタンスをコンテナデータベース (CDB) として作成できます。Oracle Database 21c 以降では、すべてのデータベースは CDB です。CDB は、RDS for Oracle でテナントデータベースと呼ばれるプラガブルデータベース (PDB) を格納できるという点で、非 CDB とは異なります。PDB は、スキーマとオブジェクトのポータブルコレクションであり、アプリケーションには別のデータベースとして表示されます。

CDB インスタンスを作成するときには、初期テナントデータベース (PDB) を作成します。RDS for Oracle では、クライアントアプリケーションは CDB ではなく PDB とやり取りします。PDB でのエクスペリエンスは、非 CDB でのエクスペリエンスとほぼ同じです。

トピック

- CDB アーキテクチャのマルチテナント設定
- CDB アーキテクチャのシングルテナント設定
- CDB の作成と変換のオプション

- CDB のユーザーアカウントと権限
- CDB のパラメータグループファミリー
- RDS for Oracle CDB の制限事項

CDB アーキテクチャのマルチテナント設定

RDS for Oracle は、Oracle マルチテナントアーキテクチャのマルチテナント設定 (CDB アーキテクチャとも呼ばれる) をサポートしています。この設定では、RDS for Oracle CDB インスタンスに、データベースのエディションと必要なオプションライセンスに応じて、1～30 個のテナントデータベースを含めることができます。Oracle データベースでは、テナントデータベースは PDB です。DB インスタンスが 19.0.0.0.0.ru-2022-01.r1 以降の Oracle Database を使用する必要があります。

> **Note**
>
> Amazon RDS 設定は、単なる Oracle DB エンジンではなく、Amazon RDS の機能であるため、「multitenant」ではなく「multi-tenant」と呼ばれています。同様に、RDS の用語「テナント」は、Oracle PDB だけでなく、RDS 設定のすべてのテナントを指します。RDS のドキュメントの「Oracle マルチテナント」という用語は、オンプレミスデプロイと RDS デプロイの両方に対応する Oracle データベース CDB アーキテクチャのみを指します。

以下の設定を設定できます。

- テナントデータベース名
- テナントデータベースのマスターユーザー名
- テナントデータベースのマスターパスワード (オプションで Secrets Manager と統合)
- テナントデータベース文字セット
- テナントデータベース各国語文字セット

テナントデータベース文字セットは CDB の文字セットと異なる場合があります。同じことが各国語文字セットにも当てはまります。最初のテナントデータベースを作成したら、RDS API を使用してテナントデータベースを作成、変更、または削除できます。CDB 名はデフォルトで RDSCDB に設定されており、変更できません。詳細については、「DB インスタンスの設定」および「RDS for Oracle テナントデータベースの変更」を参照してください。

CDB アーキテクチャのシングルテナント設定

RDS for Oracle は、シングルテナント設定と呼ばれる Oracle マルチテナントアーキテクチャのレガシー設定をサポートしています。この設定では、RDS for Oracle CDB インスタンスには 1 つのテナント (PDB) しか格納できません。後にさらに PDB を作成することはできません。

CDB の作成と変換のオプション

Oracle Database 21c と Oracle Database 26ai は CDB のみをサポートしていますが、Oracle Database 19c は CDB と非 CDB の両方をサポートしています。RDS for Oracle CDB インスタンスはすべて、マルチテナント設定とシングルテナント設定の両方をサポートしています。

Oracle データベースアーキテクチャの作成、変換、アップグレードのオプション

次の表に、RDS for Oracle Database の作成とアップグレードのさまざまなアーキテクチャオプションを示します。

リリース	データベース作成オプション	アーキテクチャ変換オプション	メジャーバージョンアップグレードターゲット
Oracle Database 26ai	CDB アーキテクチャのみ	該当なし	該当なし
Oracle Database 21c	CDB アーキテクチャのみ	該当なし	Oracle Database 26ai CDB
Oracle Database 19c	CDB または非 CDB アーキテクチャ	非 CDB から CDB アーキテクチャへ (April 2021 RU 以降)	Oracle Database 21c CDB、Oracle Database 26ai CDB

前の表に示すように、非 CDB を新しいメジャーデータベースバージョンの CDB に直接アップグレードすることはできません。ただし、Oracle Database 19c の非 CDB を Oracle Database 19c CDB に変換してから、Oracle Database 19c CDB を Oracle Database 21c または Oracle Database 26ai CDB にアップグレードすることはできます。詳細については、「RDS for Oracle の 非 CDB から CDB への変換」を参照してください。

CDB アーキテクチャ設定の変換オプション

次の表は、RDS for Oracle DB インスタンスのアーキテクチャ設定を変換するためのさまざまなオプションを示しています。

現在のアーキテクチャと設定	CDB アーキテクチャのシングルテナント設定への変換	CDB アーキテクチャのマルチテナント設定への変換	非 CDB アーキテクチャへの変換
非 CDB	サポート対象	サポート対象*	該当なし
シングルテナント設定を使用した CDB	該当なし	サポート対象	サポートされていません
マルチテナント設定を使用した CDB	サポートされていません	該当なし	サポートされていません

* 非 CDB を 1 回の操作でマルチテナント設定に変換することはできません。非 CDB を CDB に変換すると、CDB はシングルテナント設定になります。その後、別の操作でシングルテナントをマルチテナント設定に変換できます。

CDB のユーザーアカウントと権限

Oracle マルチテナントアーキテクチャでは、すべてのユーザーアカウントが共通ユーザーまたはローカルユーザーのいずれかになります。CDB 共通ユーザーとは、単一のアイデンティティとパスワードが CDB ルート、ならびに既存および将来のすべての PDB で認識されているデータベースユーザーです。対照的に、ローカルユーザーは 1 つの PDB にのみ存在します。

RDS マスターユーザーは PDB 内のローカルユーザーアカウントで、DB インスタンスを作成するときに名前を付けます。新しいユーザーアカウントを作成すると、これらのユーザーも PDB に常駐するローカルユーザーとなります。ユーザーアカウントを使用して新しい PDB を作成したり、既存の PDB の状態を変更したりすることはできません。

`rdsadmin` ユーザーは共通のユーザーアカウントです。このアカウントに存在する RDS for Oracle パッケージを実行できますが、`rdsadmin` としてログインすることはできません。詳細については、Oracle のドキュメントの About Common Users and Local Users を参照してください。

マルチテナント設定とシングルテナント設定の両方のマスターユーザーの場合、セルフマネージドまたは AWS Secrets Manager によって管理される認証情報を使用できます。シングルテナント設定では、マネージドマスターパスワードに `create-db-instance` などのインスタンスレベルの CLI コマンドを使用します。マルチテナント設定では、マネージドマスターパスワードに `create-tenant-database` などのテナントデータベースコマンドを使用します。Secrets Manager との統合の詳細については、「Secrets Manager を使用した RDS for Oracle テナントデータベースのマスターユーザーパスワードの管理」を参照してください。

CDB のパラメータグループファミリー

CDB には、独自のパラメータグループファミリーおよびデフォルトパラメータ値があります。CDB パラメータグループファミリーは次のとおりです。

- oracle-ee-cdb-26
- oracle-ee-cdb-21
- oracle-se2-cdb-21
- oracle-ee-cdb-19
- oracle-se2-cdb-19

RDS for Oracle CDB の制限事項

RDS for Oracle は、オンプレミス CDB で使用できる機能のサブセットをサポートしています。

CDB の制限事項

以下の制限は、RDS for Oracle に CDB レベルで適用されます。

- CDB に接続することはできません。常に CDB ではなくテナントデータベース (PDB) に接続します。非 CDB の場合と同様に、PDB のエンドポイントを指定します。唯一の違いは、データベース名に pdb_name を指定することです。ここで、pdb_name は PDB のために選択した名前です。
- マルチテナント設定の CDB をシングルテナント設定の CDB に変換することはできません。マルチテナント設定への変換は一方向で、元に戻すことはできません。
- DB インスタンスが 19.0.0.0.ru-2022-01.r1 より前の Oracle データベースのリリースを使用している場合は、マルチテナント設定を有効にしたり、マルチテナント設定に変換したりすることはできません。
- CDB でデータベースアクティビティストリームを使用することはできません。

- CDB$ROOT 内から監査を有効にすることはできません。各 PDB 内で個別に監査を有効にする必要があります。

テナントデータベース (PDB) の制限事項

RDS for Oracle マルチテナント設定のテナントデータベースには、次の制限が適用されます。

- テナントデータベースの操作をメンテナンス期間まで延期することはできません。すべての変更はすぐに反映されます。
- シングルテナント設定を使用する CDB には、テナントデータベースを追加できません。
- 1 回の操作で、複数のテナントデータベースを追加または変更することはできません。一度に追加または変更できる項目は 1 つのみです。
- テナントデータベースを CDB$ROOT または PDB$SEED という名前に変更することはできません。
- CDB 内のテナントが 1 つのみの場合、そのテナントデータベースは削除できません。
- すべての DB インスタンスクラスタイプが、RDS for Oracle CDB インスタンスで複数の PDB をサポートするのに十分なリソースを備えているわけではありません。PDB 数が増えると、小規模なインスタンスクラスのパフォーマンスと安定性に影響し、データベースのアップグレードなど、ほとんどのインスタンスレベルの操作にかかる時間が長くなります。
- rdsadmin.rdsadmin_util.rename_global_name を使用して PDB の名前を変更することはできません。代わりに modify-tenant-database API を使用する必要があります。
- 同じ CDB に複数の AWS アカウント を使用して PDB を作成することはできません。PDB は、PDB がホストされている DB インスタンスと同じアカウントが所有しているものである必要があります。
- CDB 内のすべての PDB は、同じエンドポイントとデータベースリスナーを使用します。
- 以下の操作は PDB レベルではサポートされていませんが、CDB レベルではサポートされています。
  - バックアップとリカバリ
  - データベースアップグレード
  - メンテナンスアクション
- 以下の機能は PDB レベルではサポートされていませんが、CDB レベルではサポートされています。
  - オプショングループ (オプションは CDB インスタンスのすべての PDB にインストールされます)

  - パラメータグループ (すべてのパラメータは CDB インスタンスに関連付けられたパラメータグループから導出されます)
- オンプレミスの CDB アーキテクチャではサポートされているが、RDS for Oracle CDB ではサポートされていない PDB レベルの操作には、次のようなものがあります。

> **Note**
>
> 以下のリストは完全なものではありません。

  - Application ELB
  - Proxy PDBs
  - PDB の起動または停止
  - PDB のアンプラグとプラギング

    CDB へ、または CDB からデータを移動するには、非 CDB と同じ手法を使用します。データ移行に関する詳細については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。
  - PDB レベルでのオプション設定

    PDB は CDB オプショングループからオプション設定を継承します。設定オプションの詳細については、「Oracle DB インスタンスへのオプションの追加」を参照してください。ベストプラクティスについては、DB パラメータグループを使用するを参照してください。
  - PDB のパラメータの設定

    PDB は CDB からパラメータ設定を継承します。パラメータの設定の詳細については、Amazon RDS のパラメータグループ を参照してください。
  - 同じ CDB 内の PDB に異なるリスナーを設定する
  - Oracle Flashback 機能

RDS for Oracle CDB の設定

CDB の設定は、非 CDB の設定と似ています。

トピック

- RDS for Oracle CDB インスタンスの作成

- RDS for Oracle CDB での PDB への接続

RDS for Oracle CDB インスタンスの作成

RDS for Oracle では、CDB インスタンスの作成は非 CDB の作成とほとんど同じです。その違いは、DB インスタンスを作成するときに Oracle マルチテナントアーキテクチャを選択し、アーキテクチャ設定 (マルチテナントまたはシングルテナント) を選択することです。マルチテナント設定で CDB を作成するときにタグを作成すると、RDS はそのタグを初期テナントデータベースに伝達します。CDB を作成するには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用します。

コンソール

CDB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、CDB インスタンスを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。
5. [Choose a database creation method (データベース作成方法を選択)] で [Standard Create (スタンダード作成)] を選択します。
6. [エンジンのオプション] で [Oracle] を選択します。
7. [データベース管理のタイプ] で、[Amazon RDS] を選択します。
8. [アーキテクチャ設定] では、[Oracle マルチテナントアーキテクチャ] を選択します。
9. [アーキテクチャ設定] で、次のいずれかを実行します。
   - [マルチテナント設定] を選択し、次のステップに進みます。
   - [シングルテナント設定] を選択し、ステップ 11 に進みます。
10. (マルチテナント設定) [テナントデータベース設定] で、次の変更を行います。
    - [テナントデータベース名] には、データベースの名前を入力します。PDB 名は CDB 名 (デフォルトは RDSCDB) と異なるものである必要があります。
    - [テナントデータベースのマスターユーザー名] には、PDB のマスターユーザー名を入力します。テナントデータベースマスターユーザー名を使用して CDB 自体にログインすることはできません。

- [認証情報管理] では、次のいずれかの認証情報管理オプションを選択します。
  - で管理AWS Secrets Manager

    マネージドパスワードは、インスタンス用ではなく、初期テナントデータベース用です。[暗号化キーを選択] で、Secrets Manager によって作成される KMS キーと自分で作成したキーのいずれかを選択します。

    > **Note**
    >
    > 認証情報を管理するための最も安全な手法として、AWS Secrets Manager をお勧めします。別途 料金がかかります。詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。

  - セルフマネージド

    パスワードを指定するには、[パスワードの自動生成] チェックボックスが選択されている場合はクリアします。[マスターパスワード] と [マスターパスワードを確認] に同じパスワードを入力します。
- [テナントデータベースの文字セット] では、PDB の文字セットを選択します。テナントデータベース文字セットとは異なる PDB 文字セットを選択できます。

  デフォルトの PDB 文字セットは AL32UTF8 です。デフォルト以外の PDB 文字セットを選択すると、CDB の作成が遅くなる可能性があります。

> **Note**
>
> 作成オペレーションで複数のテナントデータベースを指定することはできません。CDB の作成時に PDB が 1 つあります。PDB は、別のオペレーションで既存の CDB に追加できます。

11. (シングルテナント設定) DB インスタンスの設定 に記載されているオプションに基づいて、必要な設定を選択します。

    - [設定] セクションで、[認証情報の設定] を開きます。次に、以下の操作を実行します。

      i. [マスターユーザー名] には、PDB のローカルユーザーの名前を入力します。マスターユーザー名を使用して CDB ルートにログインすることはできません。

      ii. [認証情報管理] では、次のいずれかの認証情報管理オプションを選択します。

- で管理AWS Secrets Manager

  [暗号化キーを選択] で、Secrets Manager によって作成される KMS キーと自分で作成したキーのいずれかを選択します。

  > **Note**
  >
  > 認証情報を管理するための最も安全な手法として、AWS Secrets Manager をお勧めします。別途 料金がかかります。詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。

- セルフマネージド

  パスワードを指定するには、[パスワードの自動生成] チェックボックスが選択されている場合はクリアします。[マスターパスワード] と [マスターパスワードを確認] に同じパスワードを入力します。

12. 残りのセクションで、DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。
13. [データベースの作成] を選択します。

AWS CLI

マルチテナント設定で CDB を作成するには、以下のパラメータを指定して create-db-instance コマンドを使用します。

- `--db-instance-identifier`
- `--db-instance-class`
- `--engine { oracle-ee-cdb | oracle-se2-cdb }`
- `--master-username`
- `--master-user-password`-または`---manage-master-user-password`
- `--multi-tenant`(シングルテナント設定の場合は、`multi-tenant` も `--no-multi-tenant` も指定しないでください)
- `--allocated-storage`
- `--backup-retention-period`

各設定の詳細については、「DB インスタンスの設定」を参照してください。

次の例では、マルチテナント設定で *my-cdb-inst* という名前の RDS for Oracle DB インスタンスを作成します。--no-multi-tenant を指定するか --multi-tenant を指定しなかった場合、デフォルトの CDB 設定はシングルテナントです。エンジンは、oracle-ee-cdb: oracle-ee および --multi-tenant を指定するとエラーで失敗するコマンドです。初期テナントデータベースの名前は *mypdb* です。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --engine oracle-ee-cdb \
    --db-instance-identifier my-cdb-inst \
    --multi-tenant \
    --db-name mypdb \
    --allocated-storage 250 \
    --db-instance-class db.m5.large \
    --master-username pdb_admin \
    --manage-master-user-password \
    --backup-retention-period 3
```

Windows の場合:

```
aws rds create-db-instance ^
    --engine oracle-ee-cdb ^
    --db-instance-identifier my-cdb-inst ^
    --multi-tenant ^
    --db-name mypdb ^
    --allocated-storage 250 ^
    --db-instance-class db.m5.large ^
    --master-username pdb_admin ^
    --manage-master-user-password ^
    --backup-retention-period 3
```

このコマンドでは、次のような出力が生成されます。データベース名、文字セット、各国語文字セット、マスターユーザー、マスターユーザーシークレットは出力に含まれません。この情報は CLI コマンド describe-tenant-databases を使用して表示できます。

```
{
    "DBInstance": {
```

```
        "DBInstanceIdentifier": "my-cdb-inst",
        "DBInstanceClass": "db.m5.large",
        "MultiTenant": true,
        "Engine": "oracle-ee-cdb",
        "DBResourceId": "db-ABCDEFGJIJKLMNOPQRSTUVWXYZ",
        "DBInstanceStatus": "creating",
        "AllocatedStorage": 250,
        "PreferredBackupWindow": "04:59-05:29",
        "BackupRetentionPeriod": 3,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-0a1bcd2e",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.oracle-ee-cdb-19",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-1234567a",
            "SubnetGroupStatus": "Complete",
            ...
```

RDS API

Amazon RDS API を使用して DB インスタンスを作成するには、CreateDBInstance オペレーションを呼び出します。

各設定の詳細については、「DB インスタンスの設定」を参照してください。

RDS for Oracle CDB での PDB への接続

SQL*Plus などのユーティリティを使用して PDB に接続できます。SQL*Plus のスタンドアロンバージョンを含む Oracle Instant Client をダウンロードするには、Oracle Instant Client Downloads を参照してください。

SQL*Plus を PDB に接続するには以下の情報が必要です。

- PDB 名
- データベースユーザー名とパスワード
- DB インスタンスのエンドポイント
- ポート番号

前述の情報を見つける方法については、「RDS for Oracle DB インスタンスのエンドポイントを見つける」を参照してください。

Example SQL*Plus を使用して PDB に接続するには

次の例では、*master_user_name* をマスターユーザーに置き換えてください。また、エンドポイントを DB インスタンスに置き換えて、ポート番号および PDB サービス名を含めます。サービス名は DB インスタンスの作成時に指定した PDB の名前であり、DB インスタンス識別子ではありません。

Linux、macOS、Unix の場合:

```
sqlplus 'master_user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint)
(PORT=port))(CONNECT_DATA=(SERVICE_NAME=pdb_name)))'
```

Windows の場合:

```
sqlplus master_user_name@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint)
(PORT=port))(CONNECT_DATA=(SERVICE_NAME=pdb_name)))
```

次のような出力が表示されます。

```
SQL*Plus: Release 19.0.0.0.0 Production on Mon Aug 21 09:42:20 2021
```

ユーザーのパスワードを入力すると、SQL プロンプトが表示されます。

```
SQL>
```

> **Note**
>
> sqlplus *username*/*password*@*LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE*:1521/*database-identifier* のような短い形式の接続文字列 (Easy connect または EZCONNECT) は、最大文字数制限に達する可能性があるため、接続には使用しないでください。

CDB のバックアップと復元

RDS DB スナップショットまたは Recovery Manager (RMAN) を使用して CDB をバックアップおよび復元できます。

DB スナップショットを使用した CDB のバックアップと復元

DB スナップショットは CDB アーキテクチャでも非 CDB アーキテクチャでも同様に機能します。主な違いは次のとおりです。

- CDB の DB スナップショットを復元する場合、CDB の名前を変更することはできません。CDB 名は RDSCDB であり、変更できません。
- CDB の DB スナップショットを復元する場合、PDB の名前を変更することはできません。PDB 名は modify-tenant-database コマンドを使用して変更できます。
- スナップショット内のテナントデータベースを検索するには、CLI コマンド describe-db-snapshot-tenant-databases を使用します。
- マルチテナントアーキテクチャ設定を使用する CDB スナップショットでは、テナントデータベースを直接操作することはできません。DB スナップショットを復元すると、そのテナントデータベースもすべて復元されます。
- RDS for Oracle は、テナントデータベースのタグを DB スナップショット内のテナントデータベースに暗黙的にコピーします。テナントデータベースを復元すると、復元されたデータベースにタグが表示されます。
- DB スナップショットを復元し、--tags パラメータを使用して新しいタグを指定すると、新しいタグが既存のすべてのタグを上書きします。
- タグが付いている CDB インスタンスの DB スナップショットを作成し、--copy-tags-to-snapshot を指定すると、RDS for Oracle はテナントデータベースからスナップショット内のテナントデータベースにタグをコピーします。

詳細については、「Oracle データベースに関する考慮事項」を参照してください。

RMAN を使った CDB のバックアップと復元

RMAN を使用して CDB または個々のテナントデータベースをバックアップおよび復元する方法については、「Oracle DB インスタンスの一般的な RMAN タスクの実行」を参照してください。

RDS for Oracle の 非 CDB から CDB への変換

Oracle データベースのアーキテクチャは、非 CDB アーキテクチャから Oracle マルチテナントアーキテクチャ (CDB アーキテクチャとも呼ばれます) に `modify-db-instance` コマンドで変更できます。ほとんどの場合、新しい CDB を作成してデータをインポートするよりも、この手法を使用することをお勧めします。変換操作にはダウンタイムが発生します。

データベースエンジンのバージョンをアップグレードする場合、同じオペレーションでデータベースアーキテクチャを変更することはできません。したがって、Oracle Database 19c 非 CDB を Oracle Database 21c または Oracle Database 26ai CDB にアップグレードするには、まず 1 つのステップで非 CDB を CDB に変換してから、別のステップで 19c CDB を 21c または 26ai CDB にアップグレードする必要があります。

非 CDB 変換オペレーションには次の要件があります。

- DB エンジンタイプに対して `oracle-ee-cdb` または `oracle-se2-cdb` を指定する必要があります。これらの値のみがサポートされています。
- お使いの DB エンジンが、April 2021 以降のリリースアップデート (RU) で Oracle Database 19c を使用している必要があります。

このオペレーションには次の制約事項があります。

- CDB を非 CDB に変換することはできません。非 CDB を CDB に変換することしかできません。
- 非 CDB を 1 回の `modify-db-instance` 呼び出しでマルチテナント設定に変換することはできません。非 CDB を CDB に変換した後、CDB はシングルテナント設定になります。シングルテナント設定をマルチテナント設定に変換するには、`modify-db-instance` をもう一度実行してください。詳細については、「シングルテナント設定からマルチテナント設定への変換」を参照してください。
- Oracle Data Guardが有効になっているプライマリデータベースまたはレプリカデータベースは変換できません。リードレプリカのある非 CDB を変換するには、まずリードレプリカをすべて削除します。
- 同じオペレーションで DB エンジンのバージョンをアップグレードし、非 CDB を CDB に変換することはできません。

非 CDB を変換する前に、次の点を考慮してください。

- オプションとパラメータグループの考慮事項は、DB エンジンをアップグレードする場合と同じです。詳細については、「Oracle のデータベースアップグレードに関する考慮事項」を参照してください。
- マネージドマスターパスワードを使用する既存の非 CDB インスタンスは、1 回のオペレーションでシングルテナントインスタンスに変換できます。シングルテナントインスタンスはマネージドパスワードを継承します。
- DB インスタンスに OEMAGENT オプションがインストールされている場合は、非 CDB を変換する前にこのオプションを削除することをお勧めします。非 CDB が CDB に変換されたら、オプションを再インストールします。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。
- 変換プロセス中、RDS はオンライン REDO ログサイズをデフォルトの 128M にリセットします。

コンソール

非 CDB を CDB に変換するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスが存在する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択し、CDB インスタンスに変更する 非 CDB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. [アーキテクチャ設定] では、[Oracle マルチテナントアーキテクチャ] を選択します。変換後、CDB はシングルテナント設定になります。
6. (オプション) [DB パラメータグループ] には、CDB インスタンスの新しいパラメータグループを選択します。DB インスタンスを変換するときも、DB インスタンスをアップグレードするときと同じパラメータグループの考慮事項が適用されます。詳細については、「パラメータグループに関する考慮事項」を参照してください。
7. (オプション) [オプショングループ] では、CDB インスタンスの新しいオプショングループを選択します。DB インスタンスを変換するときも、DB インスタンスをアップグレードするときと同じオプショングループの考慮事項が適用されます。詳細については、「オプショングループに関する考慮事項」を参照してください。
8. (オプション) [認証情報管理] では、[AWS Secrets Manager で管理] または [セルフマネージド] を選択します。詳細については、「Secrets Manager による DB インスタンスのマスターユーザーパスワードの管理」を参照してください。

9. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
10. (省略可能) 変更をすぐに適用するには、[すぐに適用] を選択します。このオプションを選択すると、ダウンタイムを発生させる場合があります。詳細については、「スケジュール変更設定の使用」を参照してください。
11. 確認ページで、変更内容を確認します。正しい場合は、[DB インスタンスを変更] を選択します。

    または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

AWS CLI

シングルテナント設定で DB インスタンス上の非 CDB を CDB に変換するには、AWS CLI コマンド modify-db-instance で `--engine` を `oracle-ee-cdb` または `oracle-se2-cdb` に設定します。詳細については、「DB インスタンスの設定」を参照してください。

次の例では、*my-non-cdb* という名前の DB インスタンスを変換し、カスタムオプショングループとパラメータグループを指定します。コマンドは、Secrets Manager を使用したパスワード管理も有効にします。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my-non-cdb \
    --engine oracle-ee-cdb \
    --option-group-name custom-option-group \
    --db-parameter-group-name custom-parameter-group \
    --manage-master-user-password
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my-non-cdb ^
    --engine oracle-ee-cdb ^
    --option-group-name custom-option-group ^
    --db-parameter-group-name custom-parameter-group ^
    --manage-master-user-password
```

RDS API

非 CDB を CDB に変換するには、RDS API オペレーション ModifyDBInstance で `Engine` を指定します。

シングルテナント設定からマルチテナント設定への変換

RDS for Oracle CDB のアーキテクチャは、シングルテナント設定からマルチテナント設定に変更できます。変換前と変換後、CDB には 1 つのテナントデータベース (PDB) が含まれます。DB インスタンスのタグは、変換中に作成された最初のテナントデータベースに伝達されます。

開始する前に、IAM ポリシーにテナントデータベースを作成するアクセス許可があることを確認します。変換中、RDS for Oracle は次のメタデータを新しいテナントデータベースに移行します。

- マスターユーザー名
- マネージドマスターパスワード (ソース CDB が Secrets Manager と統合されている場合)
- データベース名
- 文字セット
- 各国語文字セット

変換前に、`describe-db-instances` コマンドを使用して上記の情報を表示します。変換前は、`describe-tenant-database` コマンドを使用して上記の情報を表示できます。

シングルテナントからマルチテナントへの変換には、次の制限事項があります。

- マルチテナント設定に変換した後、シングルテナント設定に戻すことはできません。この変換は元に戻せません。
- Oracle Data Guardが有効になっているプライマリデータベースまたはレプリカデータベースは変換できません。
- 同じオペレーションで DB エンジンのバージョンをアップグレードし、非 CDB を CDB に変換することはできません。
- 変換中にマネージドマスターユーザーのパスワードを有効化または無効化することはできません。

コンソール

シングルテナント設定を使用して CDB をマルチテナント設定に変換するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスが存在する AWS リージョンを選択します。
3. ナビゲーションペインで、[データベース] を選択し、マルチテナント設定に変更する CDB インスタンスを選択します。
4. [Modify] (変更) を選択します。
5. [アーキテクチャ設定] では、[Oracle マルチテナントアーキテクチャ] を選択します。
6. [アーキテクチャ設定] で、[マルチテナント設定] を選択します。
7. (オプション) [DB パラメータグループ] には、CDB インスタンスの新しいパラメータグループを選択します。DB インスタンスを変換するときも、DB インスタンスをアップグレードするときと同じパラメータグループの考慮事項が適用されます。
8. (オプション) [オプショングループ] では、CDB インスタンスの新しいオプショングループを選択します。DB インスタンスを変換するときも、DB インスタンスをアップグレードするときと同じオプショングループの考慮事項が適用されます。
9. すべての変更が正しいことを確認したら、[Continue] を選択して変更の概要を確認します。
10. [すぐに適用] を選択します。このオプションは、マルチテナント設定に切り替えるときに必要です。このオプションを選択すると、ダウンタイムを発生させる場合があることに注意してください。
11. 確認ページで、変更内容を確認します。正しい場合は、[DB インスタンスを変更] を選択します。

    または、[戻る] を選択して変更を編集するか、キャンセルを選択して変更をキャンセルします。

AWS CLI

シングルテナント設定を使用して CDB をマルチテナント設定に変換するには、AWS CLI コマンド modify-db-instance で `--multi-tenant` を指定します。

次の例では、`my-st-cdb` という名前の DB インスタンスをシングルテナント設定からマルチテナント設定に変換します。`--apply-immediately` オプションは必須です。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance --region us-east-1 \
    --db-instance-identifier my-st-cdb \
    --multi-tenant \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance --region us-east-1 ^
    --db-instance-identifier my-st-cdb ^
    --multi-tenant ^
    --apply-immediately
```

出力は次のようになります。

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-st-cdb",
        "DBInstanceClass": "db.r5.large",
        "MultiTenant": false,
        "Engine": "oracle-ee-cdb",
        "DBResourceId": "db-AB1CDE2FGHIJK34LMNOPRLXTXU",
        "DBInstanceStatus": "modifying",
        "MasterUsername": "admin",
        "DBName": "ORCL",
        ...
        "EngineVersion": "19.0.0.0.ru-2022-01.rur-2022-01.r1",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "bring-your-own-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:oracle-ee-cdb-19",
                "Status": "in-sync"
            }
        ],
        ...
        "PendingModifiedValues": {
            "MultiTenant": "true"
```

```
        }
    }
}
```

RDS for Oracle テナントデータベースを CDB インスタンスに追加する

RDS for Oracle のマルチテナント設定で、テナントデータベースは PDB です。テナントデータベースを追加するには、次の前提条件を満たしていることを確認します。

- CDB ではマルチテナント設定が有効になっています。詳細については、「CDB アーキテクチャのマルチテナント設定」を参照してください。
- テナントデータベースを作成するのに必要な IAM アクセス権限があります。

AWS マネジメントコンソール、AWS CLI、または RDS API を使用してテナントデータベースを追加できます。1 回の操作で複数のテナントデータベースを追加することはできません。一度に 1 つずつ追加する必要があります。CDB でバックアップ保持が有効になっている場合、Amazon RDS は新しいテナントデータベースを追加する前後に DB インスタンスをバックアップします。CDB にリードレプリカがある場合は、プライマリ DB インスタンスにのみテナントデータベースを追加できます。Amazon RDS は、レプリカにテナントデータベースを自動的に作成します。レプリケーションの正常性も検証され、テナントが作成される前に、すべてのレプリカが利用可能であり、レプリケーションの遅延が 5 分未満であることが確認されます。

コンソール

DB インスタンスにテナントデータベースを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、テナントデータベースを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[Databases] (データベース) を選択します。
4. テナントデータベースを追加する CDB インスタンスを選択します。DB インスタンスは CDB アーキテクチャのマルチテナント設定を使用する必要があります。
5. [アクション] を選択し、[テナントデータベースを追加] を選択します。
6. [テナントデータベース設定] では、以下の操作を行います。
    - [テナントデータベース名] には、新しいデータベースの名前を入力します。

- [テナントデータベースのマスターユーザー名] には、PDB のマスターユーザー名を入力します。
- 次のいずれかの認証情報管理オプションを選択します。
  - で管理AWS Secrets Manager

    [暗号化キーを選択] で、Secrets Manager によって作成される KMS キーと自分で作成したキーのいずれかを選択します。

    > **Note**
    >
    > 認証情報を管理するための最も安全な手法として、AWS Secrets Manager をお勧めします。追加料金が適用されます。AWS Secrets Manager は、リードレプリカを使用するインスタンスではサポートされていません。詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。

  - セルフマネージド

    パスワードを指定するには、[パスワードの自動生成] チェックボックスが選択されている場合はクリアします。[マスターパスワード] と [マスターパスワードを確認] に同じパスワードを入力します。

  - [追加設定] で、[初期データベース名] に PDB の名前を入力します。デフォルト名の `RDSCDB` がある CDB には名前を付けることはできません。
- [テナントデータベースの文字セット] では、PDB の文字セットを選択します。デフォルトは AL32UTF8 です。CDB 文字セットとは異なる PDB 文字セットを選択できます。インスタンスにリードレプリカがある場合、カスタム文字セットを使用してテナントを作成することはできません。必要に応じて、リードレプリカを作成する前に、カスタム文字セットを使用してテナントを作成できます。
- [テナント全国データベースの文字セット] では、PDB の全国文字セットを選択します。デフォルトは AL16UTF16 です。各国文字セットは、`NCHAR` データ型 (`NCHAR`、`NVARCHAR2` および `NCLOB`) を使用する列にのみエンコーディングを指定し、データベースのメタデータには影響しません。

前の設定の詳細については、「DB インスタンスの設定」をご参照ください。

7. [テナントの追加] を選択します。

AWS CLI

AWS CLI を使用して CDB にテナントデータベースを追加するには、create-tenant-database コマンドを使用し、次の必須パラメータを指定します。

- `--db-instance-identifier`
- `--tenant-db-name`
- `--master-username`
- `--master-user-password`

次の例では、RDS for Oracle CDB インスタンス *my-cdb-inst* に *mypdb2* という名前のテナントデータベースを作成します。PDB 文字セットは UTF-16 です。

Example

Linux、macOS、Unix の場合:

```
aws rds create-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name mypdb2 \
    --master-username mypdb2-admin \
    --master-user-password mypdb2-pwd \
    --character-set-name UTF-16
```

Windows の場合:

```
aws rds create-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name mypdb2 ^
    --master-username mypdb2-admin ^
    --master-user-password mypdb2-pwd ^
    --character-set-name UTF-16
```

出力は次の例のようになります。

```
...}
    "TenantDatabase" :
         {
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac567",
```

```
            "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:mypdb2",
            "DBInstanceIdentifier" : "my-cdb-inst",
            "TenantDBName" : "mypdb2",
            "Status" : "creating",
            "MasterUsername" : "mypdb2",
            "CharacterSetName" : "UTF-16",
            ...
        }
}...
```

RDS for Oracle テナントデータベースの変更

CDB 内のテナントデータベースは、PDB 名とマスターユーザーパスワードのみを変更できます。次の要件と制限事項に注意してください。

- DB インスタンス内のテナントデータベースの設定を変更するには、テナントデータベースが存在していなければなりません。
- 1 回の操作で、複数のテナントデータベースを変更することはできません。一度に変更できるテナントデータベースは 1 つのみです。
- テナントデータベースの名前を CDB$ROOT または PDB$SEED に変更することはできません。
- DB インスタンスにリードレプリカがある場合は、プライマリ DB インスタンスのテナントのみ変更できます。レプリケーションの正常性も検証され、テナントが変更される前にレプリカが利用可能で、レプリケーションの遅延が 5 分未満であることが確認されます。

PDBは、AWS マネジメントコンソール、AWS CLI、または RDS API を使用して変更できます。

コンソール

テナントデータベースの PDB 名またはマスターパスワードを変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、テナントデータベースを作成する AWS リージョンを選択します。
3. ナビゲーションペインで、[Databases] (データベース) を選択します。
4. データベース名またはマスターユーザーパスワードを変更したいテナントデータベースを選択します。

5. [Modify] (変更) を選択します。
6. [テナントデータベース設定] では、以下のいずれかの操作を行います。
   - [テナントデータベース名] には、新しいデータベースの新しい名前を入力します。
   - [テナントデータベースのマスターパスワード] に、新しいパスワードを入力します。
7. [テナントの変更] を選択します。

AWS CLI

AWS CLI を使用してテナントデータベースを変更するには、次のパラメータを指定して modify-tenant-database コマンドを呼び出します。

- `--db-instance-identifier #`
- `--tenant-db-name value`
- `[--new-tenant-db-name value]`
- `[--master-user-password value]`

次の例では、テナントデータベースの名前を DB インスタンス my-cdb-inst で pdb1 から pdb-hr に変更します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name pdb1 \
    --new-tenant-db-name pdb-hr
```

Windows の場合:

```
aws rds modify-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name pdb1 ^
    --new-tenant-db-name pdb-hr
```

このコマンドでは、次のような出力が生成されます。

```
{
    "TenantDatabase" : {
        "DbiResourceId" : "db-abc123",
        "TenantDatabaseResourceId" : "tdb-bac567",
        "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-inst:pdb1",
        "DBInstanceIdentifier" : "my-cdb-inst",
        "TenantDBName" : "pdb1",
        "Status" : "modifying",
        "MasterUsername" : "tenant-admin-user",
        "Port" : "6555",
        "CharacterSetName" : "UTF-16",
        "MaxAllocatedStorage" : "1000",
        "ParameterGroups": [
            {
                "ParameterGroupName": "pdb1-params",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "pdb1-options",
                "Status": "in-sync"
            }
        ],
        "PendingModifiedValues": {
            "TenantDBName": "pdb-hr"
        }
    }
}
```

RDS for Oracle テナントデータベースを CDB から削除する

AWS マネジメントコンソール、AWS CLI、または RDS API を使用してテナントデータベース (PDB) を削除できます。次の前提条件と制限事項を検討してください。

- テナントデータベースと DB インスタンスが存在している必要があります。
- 削除を成功させるには、以下のいずれかの状況が存在している必要があります。
  - テナントデータベースと DB インスタンスが使用可能である。

> **Note**
>
> 最終スナップショットを作成できますが、`delete-tenant-database` コマンドを発行する前にテナントデータベースと DB インスタンスが使用可能な状態であった場合に限ります。このスナップショットは、DB インスタンスにリードレプリカがある場合にのみ、プライマリインスタンスで作成されます。

  - テナントデータベースを作成しています。
  - DB インスタンスはテナントデータベースを変更しています。
  - DB インスタンスにリードレプリカがある場合、これらの制約はすべてのレプリカに適用されます。
- 1 回の操作で、複数のテナントデータベースを削除することはできません。
- CDB 内のテナントが 1 つのみの場合、そのテナントデータベースは削除できません。
- リードレプリカのテナントデータベースを削除することはできません。プライマリ DB インスタンスのテナントのみを削除できます。レプリケーションの正常性も検証され、テナントが削除される前にレプリケーションの遅延が 5 分未満であることが確認されます。

コンソール

テナントデータベースを削除するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択して、削除するテナントデータベースを選択します。
3. [アクション] で、[削除] を選択します。
4. DB インスタンスの最終 DB スナップショットを作成するには、[最終スナップショットを作成しますか?] を選択します。
5. 最終スナップショットの作成を選択した場合は、[Final snapshot name (最終スナップショット名)] を入力します。
6. ボックスに「**`delete me`**」と入力します。
7. [削除] を選択します。

AWS CLI

AWS CLI を使用してテナントデータベースを削除するには、次のパラメータを指定して delete-tenant-database コマンドを呼び出します。

- `--db-instance-identifier value`
- `--tenant-db-name value`
- `[--skip-final-snapshot | --no-skip-final-snapshot]`
- `[--final-snapshot-identifier value]`

*######my-cdb-inst ###### CDB ## pdb-test ######################*デフォルトでは、この操作により最終スナップショットが作成されます。

Example

Linux、macOS、Unix の場合:

```
aws rds delete-tenant-database --region us-east-1 \
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name pdb-test \
    --final-snapshot-identifier final-snap-pdb-test
```

Windows の場合:

```
aws rds delete-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name pdb-test ^
    --final-snapshot-identifier final-snap-pdb-test
```

このコマンドでは、次のような出力が生成されます。

```
{
    "TenantDatabase" : {
        "DbiResourceId" : "db-abc123",
        "TenantDatabaseResourceId" : "tdb-bac456",
        "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-inst:pdb-
test",
        "DBInstanceIdentifier" : "my-cdb-inst",
        "TenantDBName" : "pdb-test",
        "Status" : "deleting",
```

```
        "MasterUsername" : "pdb-test-admin",
        "Port" : "6555",
        "CharacterSetName" : "UTF-16",
        "MaxAllocatedStorage" : "1000",
        "ParameterGroups": [
            {
                "ParameterGroupName": "tenant-1-params",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "tenant-1-options",
                "Status": "in-sync"
            }
        ]
    }
}
```

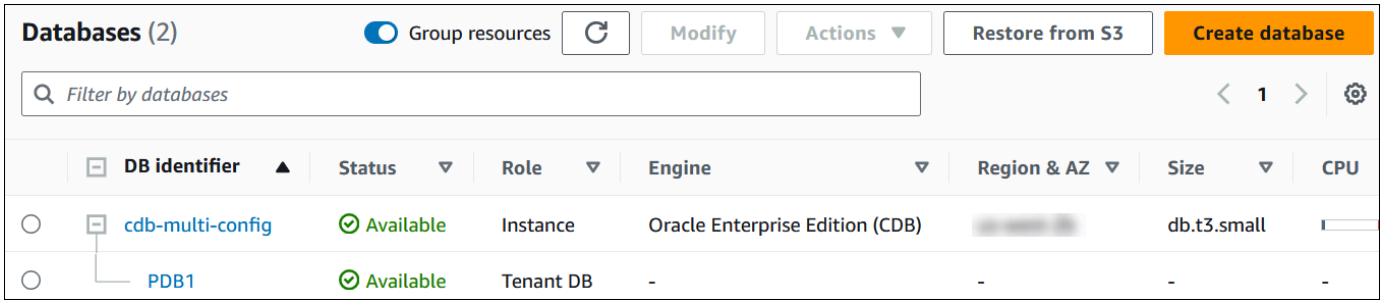
テナントデータベースの詳細を表示する

テナントデータベースの詳細は、非 CDB や CDB の場合と同じ方法で表示できます。

コンソール

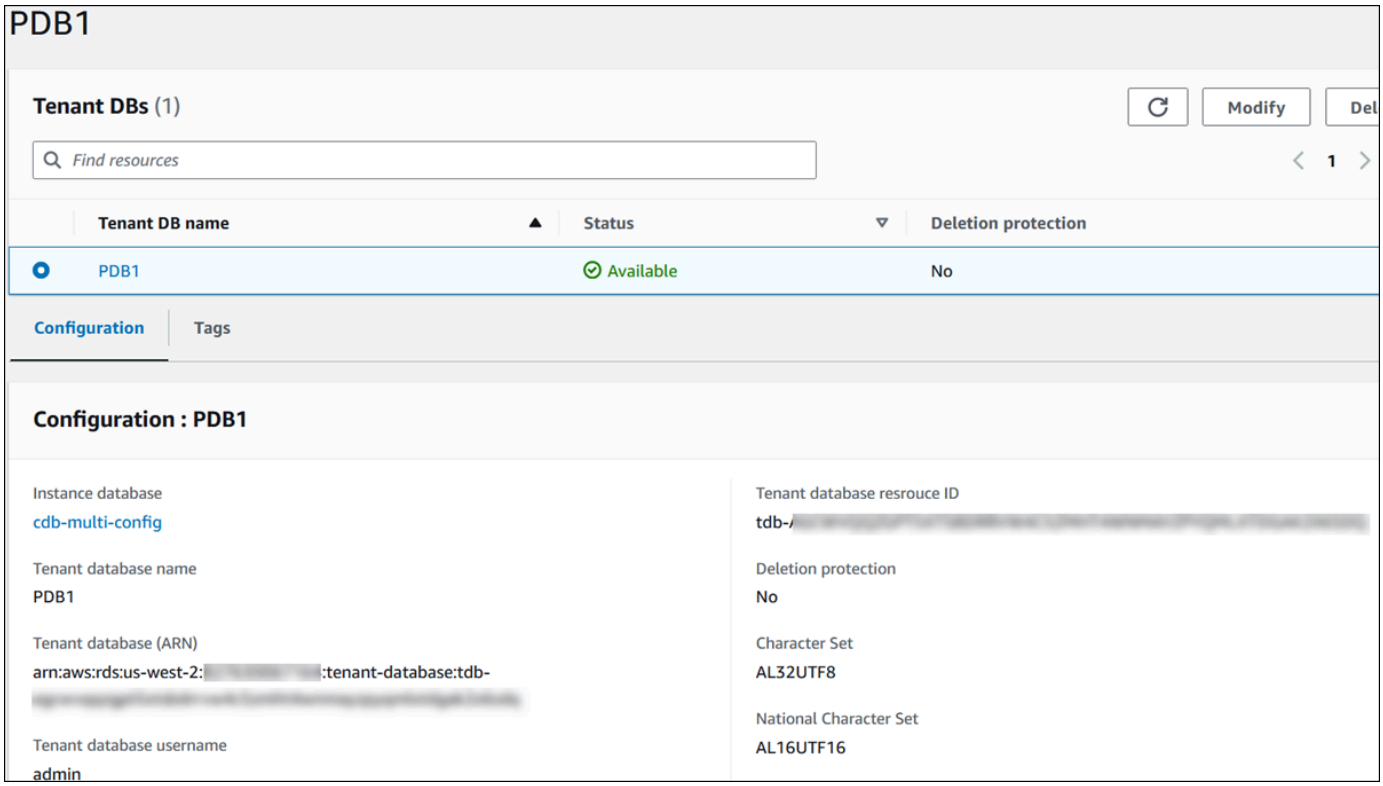
テナントデータベースの詳細情報を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、DB インスタンスが存在する AWS リージョンを選択します。
3. ナビゲーションペインで、[Databases] (データベース) を選択します。



上の画像では、唯一のテナントデータベース (PDB) は DB インスタンスの子として表示されています。

4. テナントデータベースの名前を選択します。



AWS CLI

PDB に関する詳細情報を表示するには、AWS CLI コマンド describe-tenant-databases を使用してください。

次の例では、指定したリージョンのすべてのテナントデータベースについて説明しています。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-tenant-databases --region us-east-1
```

Windows の場合:

```
aws rds describe-tenant-databases --region us-east-1
```

このコマンドでは、次のような出力が生成されます。

```
    "TenantDatabases" : [
```

```
         {
            "DBInstanceIdentifier" : "my-cdb-inst",
            "TenantDBName" : "pdb-test",
            "Status" : "available",
            "MasterUsername" : "pdb-test-admin",
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac456",
            "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:pdb-test",
            "CharacterSetName": "AL32UTF8",
            "NcharCharacterSetName": "AL16UTF16",
            "DeletionProtection": false,
            "PendingModifiedValues": {
                 "MasterUserPassword": "****"
            },
            "TagList": []
         },
         {

            "DBInstanceIdentifier" : "my-cdb-inst2",
            "TenantDBName" : "pdb-dev",
            "Status" : "modifying",
            "MasterUsername" : "masterrdsuser",
            "DbiResourceId" : "db-xyz789",
            "TenantDatabaseResourceId" : "tdb-ghp890",
            "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst2:pdb-dev",
            "CharacterSetName": "AL32UTF8",
            "NcharCharacterSetName": "AL16UTF16",
            "DeletionProtection": false,
            "PendingModifiedValues": {
                 "MasterUserPassword": "****"
            },
            "TagList": []
         },
         ... other truncated data
```

次の例では、2 番目のリージョンの DB インスタンス my-cdb-inst のテナントデータベースについて説明しています。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-tenant-databases --region us-east-1 \
    --db-instance-identifier my-cdb-inst
```

Windows の場合:

```
aws rds describe-tenant-databases --region us-east-1 ^
    --db-instance-identifier my-cdb-inst
```

このコマンドでは、次のような出力が生成されます。

```
{
    "TenantDatabase": {
        "TenantDatabaseCreateTime": "2023-10-19T23:55:30.046Z",
        "DBInstanceIdentifier": "my-cdb-inst",
        "TenantDBName": "pdb-hr",
        "Status": "creating",
        "MasterUsername": "tenant-admin-user",
        "DbiResourceId": "db-abc123",
        "TenantDatabaseResourceId": "tdb-bac567",
        "TenantDatabaseARN": "arn:aws:rds:us-west-2:579508833180:pdb-hr:tdb-
abcdefghi1jklmno2p3qrst4uvw5xy6zabc7defghi8jklmn90op",
        "CharacterSetName": "AL32UTF8",
        "NcharCharacterSetName": "AL16UTF16",
        "DeletionProtection": false,
        "PendingModifiedValues": {
            "MasterUserPassword": "****"
        },
        "TagList": [
            {
                "Key": "TEST",
                "Value": "testValue"
            }
        ]
    }
}
```

次の例では、米国東部 (バージニア北部)リージョンの DB インスタンス pdb1 のテナントデータベースについて説明しています。

Example

Linux、macOS、Unix の場合:

```
aws rds describe-tenant-databases --region us-east-1 \
--db-instance-identifier my-cdb-inst \
--tenant-db-name pdb1
```

Windows の場合:

```
aws rds describe-tenant-databases --region us-east-1 ^
--db-instance-identifier my-cdb-inst ^
--tenant-db-name pdb1
```

このコマンドでは、次のような出力が生成されます。

```
{
    "TenantDatabases" : [
        {
            "DbiResourceId" : "db-abc123",
            "TenantDatabaseResourceId" : "tdb-bac567",
            "TenantDatabaseARN" : "arn:aws:rds:us-east-1:123456789012:db:my-cdb-
inst:pdb1",
            "DBInstanceIdentifier" : "my-cdb-inst",
            "TenantDBName" : "pdb1",
            "Status" : "available",
            "MasterUsername" : "masterawsuser",
            "Port" : "1234",
            "CharacterSetName": "UTF-8",
            "ParameterGroups": [
                {
                    "ParameterGroupName": "tenant-custom-pg",
                    "ParameterApplyStatus": "in-sync"
                }
            ],
            "OptionGroupMemberships": [
                {
                    "OptionGroupName": "tenant-custom-og",
                    "Status": "in-sync"
                }
            ]
         }
    ]
}
```

CDB のアップグレード

CDB を別の Oracle Database リリースにアップグレードできます。例えば、Oracle Database 19c CDB から Oracle Database 21c CDB にアップグレードできます。アップグレード中にデータベースアーキテクチャを変更することはできません。そのため、非 CDB を CDB にアップグレードしたり、CDB を非 CDB にアップグレードすることはできません。

CDB を CDB にアップグレードする手順は、非 CDB を非 CDB にアップグレードする手順と同じです。詳細については、「RDS for Oracle DB エンジンのアップグレード」を参照してください。

RDS for Oracle DB インスタンスの管理

以下は、RDS for Oracle DB インスタンスで実行する一般的な管理タスクです。一部のタスクは、すべての RDS DB インスタンスで同じです。その他のタスクは、RDS for Oracle に固有です。

次のタスクはすべての RDS データベースに共通ですが、Oracle データベースには特別な考慮事項があります。例えば、Oracle クライアント SQL*Plus および SQL Developer を使用して Oracle Database に接続します。

タスク領域	関連資料
インスタンスクラス、ストレージ、PIOPS 本稼働インスタンスを作成する場合は、Amazon RDS でインスタンスクラス、ストレージタイプ、プロビジョンド IOPS がどのように機能するかを学習します。	RDS for Oracle DB インスタンスクラス Amazon RDS ストレージタイプ
マルチ AZ 配置 本稼働 DB インスタンスは、マルチ AZ 配置を使用する必要があります。マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。	Amazon RDS でのマルチ AZ 配置の設定と管理
Amazon VPC AWS アカウントにデフォルト 仮想プライベートクラウド (VPC) がある場合、DB インスタンスがデフォルト (VPC) 内に自動的に作成されます。アカウントにデフォルト VPC がなく、DB インスタンスを VPC に作成する必要がある場合は、インスタンスを作成する前に VPC とサブネットグループを作成する場合があります。	VPC 内の DB インスタンスの使用
セキュリティグループ: デフォルトでは、DB インスタンスはアクセスを防止するファイアウォールを使用します。DB インスタンスにアクセスするために、正しい IP アドレスとネットワーク構成を備えたセキュリティグループを作成する必要があります。	セキュリティグループによるアクセス制御

タスク領域	関連資料
パラメータグループ DB インスタンスに特定のデータベースパラメータが必要になる場合は、DB インスタンスを作成する前にパラメータグループを作成します。	Amazon RDS のパラメータグループ
オプショングループ DB インスタンスに特定のデータベースオプションが必要になる場合は、DB インスタンスを作成する前にオプショングループを作成します。	Oracle DB インスタンスへのオプションの追加
DB インスタンスへの接続 セキュリティグループを作成し、それを DB インスタンスに関連付けると、Oracle SQL *Plus などのスタンダード的な SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。	Oracle DB インスタンスへの接続
バックアップと復元 バックアップが自動的に作成されるように DB インスタンスを設定したり、スナップショットを手動で作成したりできます。そうすることで後で、そのバックアップまたはスナップショットからインスタンスを復元できます。	データのバックアップ、復元、エクスポート
モニタリング: CloudWatch Amazon RDS メトリクス、イベント、および拡張モニタリングを使用することで、Oracle DB インスタンスをモニタリングできます。	Amazon RDS コンソールでのメトリクスの表示 Amazon RDS イベントの表示
ログファイル Oracle DB インスタンスのログファイルにアクセスできます。	Amazon RDS ログファイルのモニタリング

以下では、RDS Oracle 共通の DBA タスクでの、Amazon RDS 専用の実装に関する説明を行います。マネージドサービスエクスペリエンスを提供するうえで、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、RDS では、アドバンストの特権を必要とする特定のシステムの手順やテーブルへのアクセスも制限しています。多くのタスクでは、データベースの管理のための Amazon RDS 専用のツールとして、`rdsadmin` パッケージを実行します。

Oracle を実行する DB インスタンスのための一般的な DBA タスクを次に示します。

- システムタスク

セッションの切断	Amazon RDS 方法: `rdsadmin.rdsadmin_util.disconnect` Oracle での方法: `alter system disconnect session`
セッションの終了	Amazon RDS 方法: `rdsadmin.rdsadmin_util.kill` Oracle での方法: `alter system kill session`
セッションでの SQL ステートメントのキャンセル	Amazon RDS 方法: `rdsadmin.rdsadmin_util.cancel` Oracle での方法: `alter system cancel sql`
制限セッションの有効化と無効化	Amazon RDS 方法: `rdsadmin.rdsadmin_util.restricted_session` Oracle での方法: `alter system enable restricted session`
共有プールのフラッシュ	Amazon RDS 方法: `rdsadmin.rdsadmin_util.flush_shared_pool` Oracle での方法: `alter system flush shared_pool`
バッファキャッシュのフラッシュ	Amazon RDS 方法: `rdsadmin.rdsadmin_util.flush_buffer_cache` Oracle での方法: `alter system flush buffer_cache`
SYS オブジェクトへの SELECT または	Amazon RDS 方法: `rdsadmin.rdsadmin_util.grant_sys_object` Oracle での方法: `grant`

EXECUTE 権限の付与	
SYS オブジェクトに対する SELECT または EXECUTE 権限の取り消し	Amazon RDS 方法: `rdsadmin.rdsadmin_util.revoke_sys_object` Oracle での方法: `revoke`
Oracle DB インスタンスの RDS_X$ ビューの管理	Amazon RDS 方法: `rdsadmin.rdsadmin_util.create_sys_x$_view` Oracle での方法: `CREATE VIEW`
非マスターユーザーへの権限の付与	Amazon RDS 方法: `grant`
パスワードを検証するためのカスタム関数の作成	Amazon RDS 方法: `rdsadmin.rdsadmin_password_verify.create_verify_function` Amazon RDS 方法: `rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn`
カスタム DNS サーバーのセットアップ	—
許可されたシステム診断イベントのリスト化	Amazon RDS 方法: `rdsadmin.rdsadmin_util.list_allowed_system_events` Oracle での方法: —
システム診断イベントの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.set_allowed_system_events` Oracle での方法: `ALTER SYSTEM SET EVENTS 'set_event_clause'`

設定されているシステム診断イベントのリスト化	Amazon RDS 方法: `rdsadmin.rdsadmin_util.list_set_system_events` Oracle での方法: `ALTER SESSION SET EVENTS 'IMMEDIATE EVENTDUMP(SYSTEM)'`
システム診断イベントの設定解除	Amazon RDS 方法: `rdsadmin.rdsadmin_util.unset_system_event` Oracle での方法: `ALTER SYSTEM SET EVENTS 'unset_event_clause'`

- データベースタスク

データベースのグローバル名の変更	Amazon RDS 方法: `rdsadmin.rdsadmin_util.rename_global_name` Oracle での方法: `alter database rename`
RDS for Oracle でのテーブルスペースの作成とサイズ設定	Amazon RDS 方法: `create tablespace` Oracle での方法: `alter database`
RDS for Oracle でのデフォルトのテーブルスペースの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.alter_default_tablespace` Oracle での方法: `alter database default tablespace`
RDS for Oracle でのデフォルトの一時テーブルスペースの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.alter_default_temp_tablespace` Oracle での方法: `alter database default temporary tablespace`

インスタンスストアに一時テーブルスペースを作成する	Amazon RDS 方法: `rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace` Oracle での方法: `create temporary tablespace`
データベースのチェックポイント機能	Amazon RDS 方法: `rdsadmin.rdsadmin_util.checkpoint` Oracle での方法: `alter system checkpoint`
分散復旧の設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.enable_distr_recovery` Oracle での方法: `alter system enable distributed recovery`
データベースタイムゾーンの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.alter_db_time_zone` Oracle での方法: `alter database set time_zone`
RDS for Oracle での外部テーブルの使用	—
自動ワークロードリポジトリ (AWR) を使用したパフォーマンスレポートの生成	Amazon RDS での方法: `rdsadmin.rdsadmin_diagnostic_util` プロシージャ Oracle での方法 : `dbms_workload_repository` パッケージ
VPC の DB インスタンスで使用するデータベースリンクの調整	—
DB インスタンスのデフォルトエディションの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.alter_default_edition` Oracle での方法: `alter database default edition`

SYS.AUD$ テーブルの監査を有効にする	Amazon RDS 方法: `rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table` Oracle での方法: `audit`
SYS.AUD$ テーブルの監査を無効にする	Amazon RDS 方法: `rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table` Oracle での方法: `noaudit`
中断したオンラインインデックス構築のクリーンアップ	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_repair.online_index_clean` Oracle での方法: `dbms_repair.online_index_clean`
破損ブロックのスキップ	Amazon RDS 方法 : いくつかの `rdsadmin.rdsadmin_dbms_repair` 手順 Oracle での方法 : `dbms_repair` パッケージ
RDS for Oracle のテーブルスペース、データファイル、一時ファイルのサイズ変更	Amazon RDS の方法: `rdsadmin.rdsadmin_util.resize_temp_tablespace` 、`rdsadmin.rdsadmin_util.resize_tempfile` 、または `rdsadmin.rdsadmin_util.autoextend_tempfile` の手順 `rdsadmin.rdsadmin_util.resize_datafile` または `rdsadmin.rdsadmin_util.autoextend_datafile` の手順 Oracle での方法: —
ごみ箱を空にする	Amazon RDS 方法: `EXEC rdsadmin.rdsadmin_util.purge_dba_recyclebin` Oracle での方法: `purge dba_recyclebin`

フルリダクションのデフォルト表示値の設定	Amazon RDS 方法: `EXEC rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val` Oracle での方法: `exec dbms_redact.UPDATE_FULL_REDACTION_VALUES`

- ログタスク

強制ログ作成の設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.force_logging` Oracle での方法: `alter database force logging`
サプリメンタルロギングの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.alter_supplemental_logging` Oracle での方法: `alter database add supplemental log`
オンラインログファイルを切り替える	Amazon RDS 方法: `rdsadmin.rdsadmin_util.switch_logfile` Oracle での方法: `alter system switch logfile`
オンライン REDO ログの追加	Amazon RDS 方法: `rdsadmin.rdsadmin_util.add_logfile`

オンライン REDO ログの削除	Amazon RDS 方法: `rdsadmin.rdsadmin_util.drop_logfile`
オンライン REDO ログのサイズ変更	—
アーカイブ REDO ログの保持	Amazon RDS 方法: `rdsadmin.rdsadmin_util.set_configuration`
Amazon S3 からのアーカイブ REDO ログのダウンロード	Amazon RDS 方法: `rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum` Amazon RDS 方法: `rdsadmin.rdsadmin_archive_log_download.download_logs_in_seqnum_range`
オンライン およびアーカイブ REDO ログへのアクセス	Amazon RDS 方法: `rdsadmin.rdsadmin_master_util.create_archivelog_dir` Amazon RDS 方法: `rdsadmin.rdsadmin_master_util.create_onlinelog_dir`

- RMAN タスク

RDS for Oracle でのデータベースファイルの検証	Amazon RDS 方法: `rdsadmin_rman_util . procedure` Oracle での方法: `RMAN VALIDATE`
ブロック変更追跡の有効化/無効化	Amazon RDS 方法: `rdsadmin_rman_util . procedure` Oracle での方法: `ALTER DATABASE`
アーカイブ REDO ログのクロスチェック	Amazon RDS 方法: `rdsadmin_rman_util .crosscheck_archivelog` Oracle での方法: `RMAN BACKUP`
アーカイブ REDO ログファイルのバックアップ	Amazon RDS 方法: `rdsadmin_rman_util . procedure` Oracle での方法: `RMAN BACKUP`
データベースの完全バックアップの実行	Amazon RDS 方法: `rdsadmin_rman_util .backup_database_full` Oracle での方法: `RMAN BACKUP`

データベースの増分バックアップの実行	Amazon RDS 方法: `rdsadmin_rman_util.backup_database_incremental` Oracle での方法: `RMAN BACKUP`
テーブルスペースのバックアップ	Amazon RDS 方法: `rdsadmin_rman_util.backup_database_tablespace` Oracle での方法: `RMAN BACKUP`

- Oracle Scheduler タスク

DBMS_SCHEDULER ジョブの変更	Amazon RDS 方法: `dbms_scheduler.set_attribute` Oracle での方法: `dbms_scheduler.set_attribute`
自動タスクメンテナンスウィンドウの変更	Amazon RDS 方法: `dbms_scheduler.set_attribute` Oracle での方法: `dbms_scheduler.set_attribute`

Oracle Scheduler ジョブのタイムゾーンの設定	Amazon RDS 方法: `dbms_scheduler.set_scheduler_attribute` Oracle での方法: `dbms_scheduler.set_scheduler_attribute`
SYS が所有する Oracle Scheduler ジョブの無効化	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_scheduler.disable` Oracle での方法: `dbms_scheduler.disable`
SYS が所有する Oracle Scheduler ジョブを有効にする	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_scheduler.enable` Oracle での方法: `dbms_scheduler.enable`
CALENDAR タイプのジョブに対する Oracle Scheduler の繰り返し間隔の変更	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_scheduler.set_attribute` Oracle での方法: `dbms_scheduler.set_attribute`

NAMED タイプのジョブに対する Oracle Scheduler の繰り返し間隔の変更	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_scheduler.set_attribute` Oracle での方法: `dbms_scheduler.set_attribute`
Oracle Scheduler ジョブ作成のオートコミットをオフにする	Amazon RDS 方法: `rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag` Oracle での方法: `dbms_isched.set_no_commit_flag`

- 問題の診断

インシデントのリスト化	Amazon RDS 方法: `rdsadmin.rdsadmin_adrci_util.list_adrci_incidents` Oracle での方法: ADRCI コマンド `show incident`
問題のリスト化	Amazon RDS 方法: `rdsadmin.rdsadmin_adrci_util.list_adrci_problem` Oracle での方法: ADRCI コマンド `show problem`

インシデントパッケージの作成	Amazon RDS 方法: `rdsadmin.rdsadmin_adrci_util.create_adrci_package` Oracle での方法: ADRCI コマンド `ips create package`
トレースファイルの表示	Amazon RDS 方法: `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile` Oracle での方法: ADRCI コマンド `show tracefile`

- その他のタスク

主要データストレージ領域でのディレクトリの作成と削除	Amazon RDS 方法: `rdsadmin.rdsadmin_util.create_directory` Oracle での方法: `CREATE DIRECTORY` Amazon RDS 方法: `rdsadmin.rdsadmin_util.drop_directory` Oracle での方法: `DROP DIRECTORY`

DB インスタンスディレクトリ内のファイルのリスト化	Amazon RDS 方法: `rdsadmin.rds_file_util.listdir` Oracle での方法: —
DB インスタンスディレクトリ内のファイルの読み取り	Amazon RDS 方法: `rdsadmin.rds_file_util.read_text_file` Oracle での方法: —
Opatch ファイルへのアクセス	Amazon RDS での方法: `rdsadmin.rds_file_util.read_text_file` または `rdsadmin.tracefile_listing` Oracle での方法: `opatch`
アドバイザータスクのパラメータの設定	Amazon RDS 方法: `rdsadmin.rdsadmin_util.advisor_task_set_parameter` Oracle での方法: さまざまなストアドパッケージプロシージャ
AUTO_STATS_ADVISOR_TASK を無効にする	Amazon RDS 方法: `rdsadmin.rdsadmin_util.advisor_task_drop` Oracle での方法: —

AUTO_STATS_ADVISOR_TASK を再度有効にする	Amazon RDS 方法: `rdsadmin.rdsadmin_util.dbms_stats_init` Oracle での方法: —

Oracle との Amazon S3 統合および、OEM Management Agent データベースタスクを実行するため Amazon RDS の手順を使用することも可能です。詳細については、「「Amazon S3 統合」」および「Management Agent を使用したデータベースタスクの実行」を参照してください。

Oracle DB インスタンスの一般的なシステムタスクの実行

次に、Oracle を実行している Amazon RDS DB インスタンスで、システムに関連する特定の一般的な DBA タスクを実行する方法を示します。マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスは提供していません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。

トピック

- セッションの切断
- セッションの終了
- セッションでの SQL ステートメントのキャンセル
- 制限セッションの有効化と無効化
- 共有プールのフラッシュ
- SYS オブジェクトへの SELECT または EXECUTE 権限の付与
- SYS オブジェクトに対する SELECT または EXECUTE 権限の取り消し
- Oracle DB インスタンスの RDS_X$ ビューの管理
- 非マスターユーザーへの権限の付与
- パスワードを検証するためのカスタム関数の作成
- カスタム DNS サーバーのセットアップ
- システム診断イベントの設定と設定の解除

セッションの切断

専用サーバープロセスを終了して現在のセッションを切断するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.disconnect を使用します。disconnect プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
sid	number	—	はい	セッション識別子。
serial	number	—	はい	セッションのシリアル番号。
method	varchar	'即時'	いいえ	有効な値は 'IMMEDIATE' または 'POST_TRANSACTION' です。

次の例では、セッションを切断します。

```
begin
    rdsadmin.rdsadmin_util.disconnect(
        sid    => sid,
        serial => serial_number);
end;
/
```

セッション識別子とセッションのシリアル番号を取得するには、V$SESSION ビューを照会します。次の例では、ユーザー AWSUSER のすべてのセッションを取得します。

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION WHERE USERNAME = 'AWSUSER';
```

このメソッドを使用するにはデータベースが開いている必要があります。セッションの切断の詳細については、Oracle ドキュメントの「ALTER SYSTEM」を参照してください。

セッションの終了

セッションを終了するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.kill を使用します。kill プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
sid	number	—	はい	セッション識別子。
serial	number	—	はい	セッションのシリアル番号。
method	varchar	null	いいえ	有効な値は 'IMMEDIATE' または 'PROCESS' です。IMMEDIATE を指定した場合、次のステートメントを実行した場合と同じ結果になります。 `ALTER SYSTEM KILL SESSION 'sid,serial#' IMMEDIATE` PROCESS を指定した場合、セッションに関連付けられたプロセスを終了します。IMMEDIATE を使用したセッションの終了に失敗した場合のみ、PROCESS を指定してください。

セッション識別子とセッションのシリアル番号を取得するには、V$SESSION ビューを照会します。次の例では、ユーザー *AWSUSER* のすべてのセッションを取得します。

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION WHERE USERNAME = 'AWSUSER';
```

次の例では、セッションを終了します。

```
BEGIN
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
```

```
        serial => serial_number,
        method => 'IMMEDIATE');
END;
/
```

次の例では、セッションに関連付けられたプロセスを終了します。

```
BEGIN
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
        serial => serial_number,
        method => 'PROCESS');
END;
/
```

セッションでの SQL ステートメントのキャンセル

セッション内の SQL ステートメントをキャンセルするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.cancel` を使用します。

> **Note**
>
> この手順は Oracle Database 19c (19.0.0)、および RDS for Oracle のすべての上位メジャーバージョンおよびマイナーバージョンでサポートされています。

`cancel` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`sid`	number	—	はい	セッション識別子。
`serial`	number	—	はい	セッションのシリアル番号。
`sql_id`	varchar2	null	いいえ	SQL ステートメントの SQL 識別子。

次の例では、セッション内の SQL ステートメントをキャンセルします。

```
begin
    rdsadmin.rdsadmin_util.cancel(
        sid    => sid,
        serial => serial_number,
        sql_id => sql_id);
end;
/
```

セッション識別子、セッションのシリアル番号、および SQL ステートメントの SQL 識別子を取得するには、V$SESSION ビューを照会します。次の例では、ユーザー AWSUSER のすべてのセッションと SQL 識別子を取得します。

```
select SID, SERIAL#, SQL_ID, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

制限セッションの有効化と無効化

制限セッションを有効または無効にするには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.restricted_session を使用します。restricted_session プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_enable	boolean	true	いいえ	制限セッションを有効にするには true、制限セッションを無効にするには false に設定します。

次の例では、制限セッションを有効化および無効化する方法を示します。

```
/* Verify that the database is currently unrestricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED

/* Enable restricted sessions */
```

```
EXEC rdsadmin.rdsadmin_util.restricted_session(p_enable => true);


/* Verify that the database is now restricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
----------
RESTRICTED


/* Disable restricted sessions */

EXEC rdsadmin.rdsadmin_util.restricted_session(p_enable => false);


/* Verify that the database is now unrestricted again. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED
```

共有プールのフラッシュ

共用プールをフラッシュするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.flush_shared_pool` を使用します。`flush_shared_pool` プロシージャにはパラメータはありません。

次の例では、共有プールをフラッシュします。

```
EXEC rdsadmin.rdsadmin_util.flush_shared_pool;
```

バッファキャッシュのフラッシュ

バッファキャッシュをフラッシュするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.flush_buffer_cache` を使用します。`flush_buffer_cache` プロシージャにはパラメータはありません。

次の例では、バッファキャッシュをフラッシュします。

```
EXEC rdsadmin.rdsadmin_util.flush_buffer_cache;
```

データベースのスマートフラッシュキャッシュのフラッシュ

データベースのスマートフラッシュキャッシュをフラッシュするには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.flush_flash_cache を使用します。flush_flash_cache プロシージャにはパラメータはありません。次の例では、データベースのスマートフラッシュキャッシュをフラッシュします。

```
EXEC rdsadmin.rdsadmin_util.flush_flash_cache;
```

RDS for Oracle でのデータベーススマートフラッシュキャッシュの使用に関する詳細は、「RDS for Oracle インスタンスストアへの一時データの保存」を参照してください。

SYS オブジェクトへの SELECT または EXECUTE 権限の付与

通常多くのオブジェクトを含めることができるロールを使用して権限を転送します。1 つのオブジェクトへ権限を付与するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.grant_sys_object を使用します。このプロシージャは、ロールまたは直接付与によってマスターユーザーに既に付与されている権限のみを付与します。

grant_sys_object プロシージャには以下のパラメータがあります。

> **Important**
>
> 大文字と小文字を区別する識別子を使用してユーザーを作成した場合を除き、すべてのパラメータ値に大文字を使用します。例えば、CREATE USER myuser または CREATE USER MYUSER を実行すると、データディクショナリに MYUSER が保存されます。ただし、CREATE USER "MyUser" で二重引用符を使用すると、データディクショナリには MyUser が保存されます。

パラメータ名	データ型	デフォルト	必須	説明
p_obj_name	varchar2	—	はい	権限を付与する元のオブジェクトの名前。オブジェクトとして、ディ

パラメータ名	データ型	デフォルト	必須	説明
				レクトリ、ファンクション、パッケージ、プロシージャ、シーケンス、テーブル、またはビューを指定できます。オブジェクト名のスペルは DBA_OBJECTS に表示されているとおりに正確に入力する必要があります。ほとんどのシステムオブジェクトが大文字で定義されるため、初めにこれを試すことをお勧めします。
p_grantee	varchar2	—	はい	権限を付与する先のユーザーまたはロールの名前。
p_privilege	varchar2	null	はい	付与する権限。有効な値は、SELECT、EXECUTE、および ALL です。
p_grant_option	ブール値	false	いいえ	付与オプションで使用するには、true に設定します。

次の例では、オブジェクト V_$SESSION に対する選択権限をユーザー USER1 に付与します。

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_grantee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

次の例では、付与オプションを使用して、オブジェクト V_$SESSION に対する選択権限をユーザー USER1 に付与します。

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name     => 'V_$SESSION',
        p_grantee      => 'USER1',
        p_privilege    => 'SELECT',
        p_grant_option => true);
end;
/
```

オブジェクトに対して権限を付与するには、付与オプションまたは with admin option を使用して付与されたロールを通じてアカウントにこれらの権限が直接付与されている必要があります。最も一般的なケースとして、SELECT ロールに付与された DBA ビューでの SELECT_CATALOG_ROLE 権限の付与があります。このロールが with admin option を通じてユーザーにまだ直接付与されていない場合、権限を転送することはできません。DBA 権限がある場合は、そのロールを他のユーザーに直接付与できます。

次の例では、SELECT_CATALOG_ROLE と EXECUTE_CATALOG_ROLE を USER1 に付与します。with admin option が使用されているため、USER1 は、SELECT_CATALOG_ROLE に許可された SYS オブジェクトへのアクセスを許可できるようになりました。

```
GRANT SELECT_CATALOG_ROLE TO USER1 WITH ADMIN OPTION;
GRANT EXECUTE_CATALOG_ROLE to USER1 WITH ADMIN OPTION;
```

既に PUBLIC として許可されたオブジェクトへのアクセスは再許可する必要はありません。grant_sys_object プロシージャを使用してアクセスを再許可すると、プロシージャの呼び出しは成功します。

SYS オブジェクトに対する SELECT または EXECUTE 権限の取り消し

1 つのオブジェクトに対する権限を取り消すには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.revoke_sys_object を使用します。このプロシージャは、ロールまたは直接付与によってマスターユーザーに既に付与されている権限のみを取り消します。

revoke_sys_object プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_obj_name	varchar2	—	はい	権限を取り消す対象のオブジェクトの名前。オブジェクトとして、ディレクトリ、ファンクション、パッケージ、プロシージャ、シーケンス、テーブル、またはビューを指定できます。オブジェクト名のスペルは DBA_OBJECTS に表示されているとおりに正確に入力する必要があります。ほとんどのシステムオブジェクトが大文字で定義されるため、初めにこれを試すことをお勧めします。
p_revokee	varchar2	—	はい	権限を取り消すユーザーまたはロールの名前。
p_privilege	varchar2	null	はい	取り消す権限。有効な値は、SELECT、EXECUTE、および ALL です。

次の例では、オブジェクト V_$SESSION に対する選択権限をユーザー USER1 から取り消します。

```
begin
    rdsadmin.rdsadmin_util.revoke_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_revokee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

Oracle DB インスタンスの RDS_X$ ビューの管理

SYS によってのみアクセス可能な SYS.X$ 固定テーブルへのアクセスが必要になる場合があります。対象の X$ テーブルに SYS.RDS_X$ ビューを作成するには、rdsadmin.rdsadmin_util パッケージの手順を使用します。マスターユーザーには、RDS_X$ ビューに対する SELECT … WITH GRANT OPTION 権限が自動的に付与されます。

この rdsadmin.rdsadmin_util 手順は、以下の場合に利用できます。

- まだアップグレードされたことがなく、以下のリリースを使用している既存の DB インスタンス:
  - 21.0.0.0.ru-2023-10.rur-2023-10.r1 以降の 21c リリース
  - 19.0.0.0.ru-2023-10.rur-2023-10.r1 以降の 19c リリース
- 作成した新しい DB インスタンス
- アップグレードした既存の DB インスタンス

> **Important**
>
> 内部的には、rdsadmin.rdsadmin_util パッケージは X$ テーブルにビューを作成します。X$ テーブルは内部システムオブジェクトであり、Oracle Database のドキュメントでは説明されていません。本番用以外のデータベースで特定のビューをテストし、Oracle サポートのガイダンスに従って、本番用のデータベースにのみビューを作成することをお勧めしています。

RDS_X$ ビューで使用できる X$ 固定テーブルを一覧表示する

RDS_X$ ビューで使用できる X$ テーブルを一覧表示するには、RDS プロシージャ rdsadmin.rdsadmin_util.list_allowed_sys_x$_views を使用します。この手順はパラメータを受け付けません。次のステートメントは、対象となるすべての X$ テーブル (サンプル出力を含む) を一覧表示します。

```
SQL> SET SERVEROUTPUT ON
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_util.list_allowed_sys_x$_views);

'X$BH'
'X$K2GTE'
'X$KCBWBPD'
```

```
'X$KCBWDS'
'X$KGLLK'
'X$KGLOB'
'X$KGLPN'
'X$KSLHOT'
'X$KSMSP'
'X$KSPPCV'
'X$KSPPI'
'X$KSPPSV'
'X$KSQEQ'
'X$KSQRS'
'X$KTUXE'
'X$KQRFP'
```

対象となる X$ テーブルの一覧は、時間の経過とともに変化する可能性があります。対象となる X$ 固定テーブルのリストが最新であることを確認するには、定期的に list_allowed_sys_x$_views を再実行してください。

SYS.RDS_X$ ビューの作成

対象となる X$ テーブルに RDS_X$ ビューを作成するには、RDS プロシージャ rdsadmin.rdsadmin_util.create_sys_x$_view を使用します。rdsadmin.rdsadmin_util.list_allowed_sys_x$_views の出力に一覧表示されているテーブルのビューのみを作成できます。create_sys_x$_view 手順は、次のパラメータを受け付けます。

パラメータ名	データ型	デフォルト	必須	説明
p_x$_tbl	varchar2	Null	はい	有効な X$ テーブル名。値は、list_allowed_sys_x$_views によって報告される X$ テーブルの 1 つである必要があります。
p_force_creation	ブール値	FALSE	いいえ	X$ テーブルに既に存在する RDS_X$ ビューを強制的に作成するかどうかを示す値。デフォルトでは、ビューが既に存在す

パラメータ名	データ型	デフォルト	必須	説明
				る場合、RDS はビューを作成しません。強制的に作成するには、このパラメータを TRUE に設定します。

次の例では、テーブル X$KGLOB で SYS.RDS_X$KGLOB ビューを作成します。ビュー名の形式は RDS_*X$tablename* です。

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.create_sys_x$_view('X$KGLOB');

PL/SQL procedure successfully completed.
```

次のデータディクショナリクエリは、ビュー SYS.RDS_X$KGLOB とそのステータスを表示します。マスターユーザーには、このビューに対する SELECT ... WITH GRANT OPTION 権限が自動的に付与されます。

```
SQL> SET SERVEROUTPUT ON
SQL> COL OWNER FORMAT A30
SQL> COL OBJECT_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT OWNER, OBJECT_NAME, STATUS
FROM DBA_OBJECTS
WHERE OWNER = 'SYS' AND OBJECT_NAME = 'RDS_X$KGLOB';

OWNER                          OBJECT_NAME                    STATUS
------------------------------ ------------------------------
 ------------------------------
SYS                            RDS_X$KGLOB                    VALID
```

Important

X$ テーブルは、アップグレード前とアップグレード後で同じままである保証はありません。RDS for Oracle は、エンジンのアップグレード中に X$ テーブルで RDS_X$ ビューを削除して再作成します。次に、マスターユーザーに SELECT ... WITH GRANT OPTION 権

限を付与します。アップグレード後、必要に応じて対応する RDS_X$ ビューでデータベースユーザーに権限を付与します。

SYS.RDS_X$ ビューの一覧表示

既存の RDS_X$ ビューを一覧表示するには、RDS プロシージャ rdsadmin.rdsadmin_util.list_created_sys_x$_views を使用します。このプロシージャでは、プロシージャ create_sys_x$_view によって作成されたビューのみが一覧表示されます。次の例では、対応する RDS_X$ ビュー (サンプル出力を含む) を持つ X$ テーブルを一覧表示します。

```
SQL> SET SERVEROUTPUT ON
SQL> COL XD_TBL_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_util.list_created_sys_x$_views);

XD_TBL_NAME                    STATUS
------------------------------ ------------------------------
X$BH                           VALID
X$K2GTE                        VALID
X$KCBWBPD                      VALID

3 rows selected.
```

RDS_X$ ビューの削除

SYS.RDS_X$ ビューを削除するには、RDS プロシージャ rdsadmin.rdsadmin_util.drop_sys_x$_view を使用します。rdsadmin.rdsadmin_util.list_allowed_sys_x$_views の出力に一覧表示されているビューだけを削除できます。drop_sys_x$_view 手順は、次のパラメータを受け付けます。

パラメータ名	データ型	デフォルト	必須	説明
p_x$_tbl	varchar2	Null	はい	有効な X$ 固定テーブル名。この値は、list_created_sys_x$_views

パラメータ名	データ型	デフォルト	必須	説明
				によって報告される X$ 固定テーブルの 1 つである必要があります。

次の例では、テーブル X$KGLOB で作成された RDS_X$KGLOB ビューを削除します。

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.drop_sys_x$_view('X$KGLOB');

PL/SQL procedure successfully completed.
```

次の例は、ビュー SYS.RDS_X$KGLOB が削除されたことを示しています (サンプル出力を含む)。

```
SQL> SET SERVEROUTPUT ON
SQL> COL OWNER FORMAT A30
SQL> COL OBJECT_NAME FORMAT A30
SQL> COL STATUS FORMAT A30
SQL> SET LINESIZE 200
SQL> SELECT OWNER, OBJECT_NAME, STATUS
FROM DBA_OBJECTS
WHERE OWNER = 'SYS' AND OBJECT_NAME = 'RDS_X$KGLOB';

no rows selected
```

非マスターユーザーへの権限の付与

SYS ロールを使用すると、SELECT_CATALOG_ROLE スキーマ内の多くのオブジェクトに対する選択権限を付与できます。SELECT_CATALOG_ROLE ロールは、データディクショナリビューに対する SELECT 権限をユーザーに付与します。次の例では、ユーザー SELECT_CATALOG_ROLE にロール user1 を付与します。

```
GRANT SELECT_CATALOG_ROLE TO user1;
```

EXECUTE ロールを使用すると、SYS スキーマ内の多くのオブジェクトに対する EXECUTE_CATALOG_ROLE 権限を付与できます。EXECUTE_CATALOG_ROLE ロールは、データディクショナリのパッケージとプロシージャに対する EXECUTE 権限をユーザーに付与します。次の例では、ユーザー user1 にロール EXECUTE_CATALOG_ROLE を付与します。

```
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

次の例では、ロール SELECT_CATALOG_ROLE とロール EXECUTE_CATALOG_ROLE が許可するアクセス権限を取得します。

```
  SELECT *
    FROM ROLE_TAB_PRIVS
   WHERE ROLE IN ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

次の例では、非マスターユーザー user1 を作成し、CREATE SESSION 権限を付与します。また、テーブル sh.sales に対する SELECT 権限を付与します。

```
CREATE USER user1 IDENTIFIED BY PASSWORD;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

パスワードを検証するためのカスタム関数の作成

カスタムパスワード検証関数は、以下の方法で作成できます。

- スタンダード検証ロジックを使用するには、または SYS スキーマに関数を格納するには、create_verify_function の手順を使用します。
- カスタム検証ロジックを使用するには、または SYS スキーマに関数を格納しないようにするには、create_passthrough_verify_fcn の手順を使用します。

create_verify_function プロシージャ

Amazon RDS プロシージャ rdsadmin.rdsadmin_password_verify.create_verify_function を使用してパスワードを検証するには、カスタム関数を作成できます。create_verify_function プロシージャは、RDS for Oracle のすべてのバージョンでサポートされています。

create_verify_function プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_verify_function_name	varchar2	—	はい	カスタム関数の名前。この関数は、SYS スキーマ

パラメータ名	データ型	デフォルト	必須	説明
				に作成されます。この関数をユーザープロファイルに割り当てます。
p_min_length	number	8	いいえ	必要な文字の最小数。
p_max_length	number	256	いいえ	許容された文字の最大数。
p_min_letters	number	1	いいえ	必要な文字の最小数。
p_min_uppercase	number	0	いいえ	必要な大文字の最小数。
p_min_lowercase	number	0	いいえ	必要な小文字の最小数。
p_min_digits	number	1	いいえ	必要な数字の最小数。
p_min_special	number	0	いいえ	必要な特殊文字の最小数。
p_min_different_chars	number	3	いいえ	古いパスワードと新しいパスワードの間で必要な異なる文字の最小数。
p_disallow_username	boolean	true	いいえ	パスワードでユーザー名を禁止するには、true に設定します。
p_disallow_reverse	boolean	true	いいえ	パスワードで反転したユーザー名を禁止するには、true に設定します。
p_disallow_db_name	boolean	true	いいえ	パスワードでデータベースまたはサーバー名を禁止するには、true に設定します。

パラメータ名	データ型	デフォルト	必須	説明
p_disallow_simple_strings	boolean	true	いいえ	パスワードで単純な文字列を禁止するには、true に設定します。
p_disallow_whitespace	boolean	false	いいえ	パスワードで空白文字を禁止するには、true に設定します。
p_disallow_at_sign	boolean	false	いいえ	パスワードで「@」文字の使用を禁止するには、true に設定します。

複数のパスワード検証関数を作成できます。

カスタム関数名に制限があります。カスタム関数名を既存のシステムオブジェクト名と同じにすることはできません。30 文字を超える名前は使用できません。また、名前には文字列として PASSWORD、VERIFY、COMPLEXITY、ENFORCE、STRENGTH のいずれかを含める必要があります。

次の例では、CUSTOM_PASSWORD_FUNCTION という名前の関数を作成します。この関数のパスワードは、最低で 2 つの大文字、1 つの数字、1 つの特殊文字を含む、12 文字以上で構成する必要があります。また、「@」文字は使用できません。

```
begin
    rdsadmin.rdsadmin_password_verify.create_verify_function(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_min_length           => 12,
        p_min_uppercase        => 2,
        p_min_digits           => 1,
        p_min_special          => 1,
        p_disallow_at_sign     => true);
end;
/
```

検証関数のテキストを表示するには、DBA_SOURCE を照会します。次の例では、カスタムパスワード関数 CUSTOM_PASSWORD_FUNCTION のテキストを取得します。

```
COL TEXT FORMAT a150

  SELECT TEXT
    FROM DBA_SOURCE
   WHERE OWNER = 'SYS'
     AND NAME = 'CUSTOM_PASSWORD_FUNCTION'
ORDER BY LINE;
```

検証関数をユーザープロファイルに関連付けるには、ALTER PROFILE を使用します。次の例では、CUSTOM_PASSWORD_FUNCTION という名前の検証 PL/SQL 関数を DEFAULT ユーザープロファイルに関連付けます。PASSWORD_VERIFY_FUNCTION は Oracle プロファイルリソース名です。

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

どのユーザープロファイルがどの照合機能に関連付けられているかを確認するには、DBA_PROFILES を照会します。次の例では、カスタム検証関数 CUSTOM_PASSWORD_FUNCTION に関連付けられたプロファイルを取得します。

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION' AND LIMIT =
 'CUSTOM_PASSWORD_FUNCTION';


PROFILE                    RESOURCE_NAME                    RESOURCE  LIMIT
-------------------------  -------------------------------  --------
 ------------------------
DEFAULT                    PASSWORD_VERIFY_FUNCTION         PASSWORD
 CUSTOM_PASSWORD_FUNCTION
```

次の例では、すべてのプロシージャとそれらに関連付けられたパスワード検証関数を取得します。

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION';

PROFILE                    RESOURCE_NAME                    RESOURCE  LIMIT
-------------------------  -------------------------------  --------
 ------------------------
DEFAULT                    PASSWORD_VERIFY_FUNCTION         PASSWORD
 CUSTOM_PASSWORD_FUNCTION
RDSADMIN                   PASSWORD_VERIFY_FUNCTION         PASSWORD  NULL
```

create_passthrough_verify_fcn プロシージャ

create_passthrough_verify_fcn プロシージャは、RDS for Oracle のすべてのバージョンでサポートされています。

Amazon RDS 手順

rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn を使用してパスワードを検証するには、カスタム関数を作成できます。create_passthrough_verify_fcn プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_verify_function_name	varchar2	—	はい	カスタム検証関数の名前。これは、SYS 関数で作成されるラッパー関数です。検証ロジックは含まれていません。この関数をユーザープロファイルに割り当てます。
p_target_owner	varchar2	—	はい	カスタム検証関数のスキーマ所有者。
p_target_function_name	varchar2	—	はい	検証ロジックを含む既存のカスタム関数の名前。カスタム関数はブール値を返します。パスワードが有効の場合は true、無効の場合は false が関数より返ります。

次の例では、PASSWORD_LOGIC_EXTRA_STRONG という名前の関数のロジックを使用するパスワード検証関数を作成します。

```
begin
    rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_target_owner         => 'TEST_USER',
```

```
        p_target_function_name => 'PASSWORD_LOGIC_EXTRA_STRONG');
end;
/
```

検証関数をユーザープロファイルに関連付けるには、alter profile を使用します。次の例では、検証関数を DEFAULT ユーザープロファイルに関連付けています。

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

カスタム DNS サーバーのセットアップ

Amazon RDS は、Oracle を実行している DB インスタンスでのアウトバウンドのネットワークアクセスをサポートします。前提条件など、アウトバウンドのネットワークアクセスの詳細については、「証明書と Oracle ウォレットを使用した、UTL_HTTP アクセスの設定」を参照してください

Amazon RDS Oracle は、顧客所有のカスタム DNS サーバーでドメイン名サービス (DNS) 解決を許可します。Amazon RDS DB インスタンスからカスタム DNS サーバーを介して完全修飾ドメイン名のみを解決することができます。

カスタム DNS ネームサーバーを設定後、変更を DB インスタンスに反映させるまで約 30 分ほどかかります。DB インスタンスへの変更が反映されたら、すべてのアウトバウンドネットワークトラフィックのポート 53 の DNS サーバーにおいて DNS ルックアップクエリを行う必要があります。

Amazon RDS for Oracle DB インスタンスでカスタム DNS サーバーをセットアップするには、次を実行します。

- Virtual Private Cloud (VPC) の DHCP オプションで domain-name-servers を DNS ネームサーバーの IP アドレスに設定します。詳細については、「DHCP オプションセット」を参照してください。

  > **Note**
  >
  > domain-name-servers オプションが許可する値は 4 つまでになりますが、Amazon RDS DB インスタンスが使用するのは初期の値のみです。

- DNS サーバーが、パブリック DNS 名、Amazon EC2 プライベート DNS 名、ユーザー固有の DNS 名を含むすべてのルックアップクエリを解決できることを確認します。DNS サーバーが処理できない DNS ルックアップがアウトバウンドネットワークトラフィックにある場合は、状況に適したアップストリーミング DNS プロバイダを必ず設定してください。

- 512 バイト以下の User Datagram Protocol (UDP) レスポンスを生成するように DNS サーバーを設定します。
- 1024 バイト以下の Transmission Control Protocol (TCP) レスポンスを生成するように DNS サーバーを設定します。
- ポート 53 で Amazon RDS DB インスタンスからのインバウンドトラフィックを許可するように DNS サーバーを設定します。DNS サーバーが Amazon VPC にある場合、VPC にはポート 53 で UDP と TCP トラフィックを許可するインバウンドルールを含むセキュリティグループが必要になります。DNS サーバーが Amazon VPC にない場合は、ポート 53 で UDP と TCP インバウンドトラフィックを許可できるように、適切なファイアウォールの allow-listing が必要になります。

  詳細については、「VPC のセキュリティグループ」と「ルールの追加と削除」を参照してください。
- ポート 53 でアウトバウンドトラフィックを許可するため、Amazon RDS DB インスタンスの VPC を設定します。VPC には、ポート 53 で UDP および TCP トラフィックを許可するアウトバウンドルールを含むセキュリティグループが必要になります。

  詳細については、「VPC のセキュリティグループ」と「ルールの追加と削除」を参照してください。
- Amazon RDS DB インスタンスと DNS サーバー間のルーティングパスは、DNS トラフィックを許可できるよう適切に設定してください。
  - Amazon RDS DB インスタンスと DNS サーバーが同じ VPC にない場合は、両者の間でピア接続をセットアップする必要があります。詳細については、「VPC ピア機能とは」を参照してください。

システム診断イベントの設定と設定の解除

セッションレベルで診断イベントを設定および設定解除するには、Oracle SQL ステートメントを使用できます `ALTER SESSION SET EVENTS`。ただし、システムレベルでイベントを設定するには、Oracle SQL を使用できません。代わりに、`rdsadmin.rdsadmin_util` パッケージ内のシステムイベント手順を使用します。システムイベント手順は、次のエンジンバージョンで使用できます。

- すべての Oracle Database 26ai バージョン
- すべての Oracle Database 21c バージョン
- 19.0.0.0.ru-2020-10.rur-2020-10.r1 以降の Oracle Database 19c バージョン

詳細については、「Amazon RDS for Oracle リリースノート」の「バージョン 19.0.0.0.ru-2020-10.rur-2020-10.r1」を参照してください。

> **Important**
>
> 内部的には、rdsadmin.rdsadmin_util パッケージは ALTER SYSTEM SET EVENTS ステートメントを使用してイベントを設定します。ALTER SYSTEM ステートメントは、Oracle データベースのドキュメントには記載されていません。システム診断イベントによっては、大量のトレース情報を生成したり、競合を引き起こしたり、データベースの可用性に影響を与えることがあります。非稼動のデータベースで特定の診断イベントをテストし、Oracleサポートのガイダンスに従って、本番稼動用のデータベースにのみイベントを設定することをお勧めしています。

許可されたシステム診断イベントのリスト化

設定できるシステムイベントをリスト化するには、Amazon RDS の手順 rdsadmin.rdsadmin_util.list_allowed_system_eventsを使用します。この手順はパラメータを受け付けません。

次の例では、設定可能なすべてのシステムイベントをリスト化します。

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_allowed_system_events;
```

次の出力サンプルは、イベント番号とその説明をリスト化しています。set_system_eventこれらのイベントを設定し、unset_system_event設定解除するには、Amazon RDS の手順を使用します。

```
604   - error occurred at recursive SQL level
942   - table or view does not exist
1401  - inserted value too large for column
1403  - no data found
1410  - invalid ROWID
1422  - exact fetch returns more than requested number of rows
1426  - numeric overflow
1427  - single-row subquery returns more than one row
1476  - divisor is equal to zero
```

```
1483  - invalid length for DATE or NUMBER bind variable
1489  - result of string concatenation is too long
1652  - unable to extend temp segment by  in tablespace
1858  - a non-numeric character was found where a numeric was expected
4031  - unable to allocate  bytes of shared memory ("","","","")
6502  - PL/SQL: numeric or value error
10027 - Specify Deadlock Trace Information to be Dumped
10046 - enable SQL statement timing
10053 - CBO Enable optimizer trace
10173 - Dynamic Sampling time-out error
10442 - enable trace of kst for ORA-01555 diagnostics
12008 - error in materialized view refresh path
12012 - error on auto execute of job
12504 - TNS:listener was not given the SERVICE_NAME in CONNECT_DATA
14400 - inserted partition key does not map to any partition
31693 - Table data object  failed to load/unload and is being skipped due to error:
```

> **Note**
>
> 許可されるシステムイベントのリストは、時間の経過とともに変化する可能性があります。対象イベントの最新のリストがあることを確認するには、`rdsadmin.rdsadmin_util.list_allowed_system_events` を使用します。

システム診断イベントの設定

システムイベントを設定するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.set_system_event` を使用します。`rdsadmin.rdsadmin_util.list_allowed_system_events` の出力にリストされているイベントだけを設定できます。`set_system_event` 手順は、次のパラメータを受け付けます。

パラメータ名	データ型	デフォルト	必須	説明
p_event	number	—	はい	システムイベント番号。値は、`list_allowed_system_events` によって報告されるイベント番号の 1 つである必要があります。

パラメータ名	データ型	デフォルト	必須	説明
p_level	number	—	はい	イベントレベル。異なるレベル値の説明については、Oracle データベースのドキュメントまたは Oracle サポートを参照してください。

手順 set_system_eventは、次の原則に従って必要なALTER SYSTEM SET EVENTS ステートメントを構築および実行します。

- イベントタイプ (context または errorstack) は自動的に決定されます。
- フォーム内のステートメントは、ALTER SYSTEM SET EVENTS '*event* LEVEL *event_level*'コンテキストイベントを設定します。この表記は ALTER SYSTEM SET EVENTS '*event* TRACE NAME CONTEXT FOREVER, LEVEL *event_level*' と同等です。
- フォーム内のステートメントは、ALTER SYSTEM SET EVENTS '*event* ERRORSTACK (*event_level*)'エラースタックイベントを設定します。この表記は ALTER SYSTEM SET EVENTS '*event* TRACE NAME ERRORSTACK LEVEL *event_level*' と同等です。

次の例では、イベント 942 をレベル 3 に設定し、イベント 10442 をレベル 10 に設定します。サンプル出力が含まれています。

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(942,3);
Setting system event 942 with: alter system set events '942 errorstack (3)'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(10442,10);
Setting system event 10442 with: alter system set events '10442 level 10'

PL/SQL procedure successfully completed.
```

設定されているシステム診断イベントのリスト化

現在設定されているシステムイベントをリスト化するには、Amazon RDS の手順を使用します rdsadmin.rdsadmin_util.list_set_system_events。この手順では、set_system_event によってシステムレベルで設定されたイベントだけが報告されます。

次の例は、アクティブなシステムイベントをリスト化します。

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_set_system_events;
```

次のサンプル出力は、イベントのリスト、イベントタイプ、イベントが現在設定されているレベル、およびイベントが設定された時刻を示しています。

```
942 errorstack (3) - set at 2020-11-03 11:42:27
10442 level 10 - set at 2020-11-03 11:42:41

PL/SQL procedure successfully completed.
```

システム診断イベントの設定解除

システムイベントの設定を解除するには、Amazon RDS の手順を使用します rdsadmin.rdsadmin_util.unset_system_event。設定を解除できるのは、rdsadmin.rdsadmin_util.list_allowed_system_events の出力にリスト化されたイベントだけです 。unset_system_event 手順は、次のパラメータを受け付けます。

パラメータ名	データ型	デフォルト	必須	説明
p_event	number	—	はい	システムイベント番号。値は、list_allowed_system_events によって報告されるイベント番号の 1 つである必要があります。

次の例では、イベント 942 と 10442 の設定を解除します。サンプル出力が含まれています。

```
SQL> SET SERVEROUTPUT ON
```

```
SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(942);
Unsetting system event 942 with: alter system set events '942 off'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(10442);
Unsetting system event 10442 with: alter system set events '10442 off'

PL/SQL procedure successfully completed.
```

Oracle DB インスタンスの一般的なデータベースタスクの実行

次に、Oracle を実行している Amazon RDS DB インスタンスのデータベースに関連する特定の一般的な DBA タスクを実行する方法を示します。マネージドサービスエクスペリエンスを提供するために、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、Amazon RDS では、高度な特権を必要とする、一部のシステムプロシージャやテーブルへのアクセスが制限されます。

トピック

- データベースのグローバル名の変更
- RDS for Oracle でテーブルスペースを使用する
- RDS for Oracle で一時ファイルを使用する
- RDS for Oracle のテーブルスペース、データファイル、一時ファイルのサイズ変更
- RDS for Oracle のストレージボリューム間のデータの移動
- RDS for Oracle での外部テーブルの使用

データベースのグローバル名の変更

データベースのグローバル名を変更するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.rename_global_name` を使用します。`rename_global_name` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_new_global_name	varchar2	—	はい	データベースの新しいグローバル名。

名前の変更を実行するには、データベースが開いている必要があります。データベースのグローバル名の変更の詳細については、Oracle ドキュメントの「ALTER DATABASE」を参照してください。

次の例では、データベースのグローバル名を new_global_name に変更します。

```
EXEC rdsadmin.rdsadmin_util.rename_global_name(p_new_global_name => 'new_global_name');
```

RDS for Oracle でテーブルスペースを使用する

RDS for Oracle では、データベースのデータを保存する論理ストレージユニットであるテーブルスペースを使用できます。

> **Important**
>
> DB インスタンスにレプリカがある場合は、セッションレベルの変更ではなくパラメータグループ設定を使用してデフォルトのファイルの場所を管理することをお勧めします。プライマリインスタンスのデフォルトファイルの場所に対するセッションレベルの変更は、レプリカに自動的に反映されません。パラメータグループ設定を使用すると、プライマリインスタンスとレプリカインスタンス間で一貫したファイルの場所が確保されます。

トピック

- RDS for Oracle でのデータベースファイルの場所の指定
- RDS for Oracle でのテーブルスペースの作成とサイズ設定
- RDS for Oracle で追加のストレージボリュームにテーブルスペースを作成する
- RDS for Oracle でのデフォルトのテーブルスペースの設定
- RDS for Oracle でのデフォルトの一時テーブルスペースの設定
- インスタンスストアに一時テーブルスペースを作成する

RDS for Oracle でのデータベースファイルの場所の指定

RDS for Oracle は、Oracle Managed Files (OMF) を使用してデータベースファイルに名前を付けます。データベースファイルを作成すると、データベースは DB_CREATE_FILE_DEST 初期化パラメータの現在の設定に基づいて設定を取得します。

DB_CREATE_FILE_DEST 初期化パラメータのデフォルト値は、/rdsdbdata/db スタンドアロンデータベースとコンテナ化された (CDB/MT) アーキテクチャ用 /rdsdbdata/db/pdb で

す。DB インスタンスに追加のストレージボリュームがある場合は、ボリューム口ケーションに DB_CREATE_FILE_DEST を設定できます。例えば、インスタンスに /rdsdbdata/db にマウントされたボリュームがある場合、DB_CREATE_FILE_DEST をこの値に設定できます。

DB_CREATE_FILE_DEST パラメータは、セッションレベルまたは Oracle データベースインスタンスレベルで変更できます。

インスタンスレベルで DB_CREATE_FILE_DEST を変更する

インスタンスレベルでパラメータを変更するには、DB インスタンスに割り当てられたパラメータグループのパラメータを更新して適用します。詳細については、「RDS for Oracle 初期化パラメータ」および「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

セッションレベルで DB_CREATE_FILE_DEST を変更する

ALTER SESSION ステートメントを実行して、セッションレベルでパラメータを変更できます。このアプローチは、インスタンス全体に影響を与えることなく、特定のセッションの特定の場所にデータベースファイルを作成する場合に便利です。

次の例は、現在のパラメータ値を確認し、セッションで変更する方法を示しています。

```
SHOW PARAMETER db_create_file_dest

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_create_file_dest                  string      /rdsdbdata/db

ALTER SESSION SET db_create_file_dest = '/rdsdbdata2/db';

Session altered.

SHOW PARAMETER db_create_file_dest

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_create_file_dest                  string      /rdsdbdata2/db
```

RDS for Oracle でのテーブルスペースの作成とサイズ設定

テーブルスペースを作成すると、データベースは作成時に DB_CREATE_FILE_DEST 初期化パラメータで指定されたストレージボリュームにデータファイルを作成します。データファイルのサイズを指

定しない場合、デフォルトの AUTOEXTEND ON でテーブルスペースが作成され。最大サイズはありません。次の例では、テーブルスペース *users1* は自動拡張可能です。

```
CREATE TABLESPACE users1;
```

これらのデフォルト設定のため、テーブルスペースはすべての割り当てられたストレージを消費するまで大きくなります。永続テーブルスペースとテンポラリテーブルスペースに適切な最大サイズを指定し、リージョンの使用状況を注意深くモニタリングすることをお勧めします。

次の例では、スタートサイズが 1 ギガバイトのテーブルスペース *users2* を作成します。データファイルのサイズが指定されていても、AUTOEXTEND ON が指定されていないため、テーブルスペースは自動拡張できません。

```
CREATE TABLESPACE users2 DATAFILE SIZE 1G;
```

次の例では、スタートサイズが 1 ギガバイト、自動拡張がオン、最大サイズが 10 ギガバイトのテーブルスペース *users3* を作成します。

```
CREATE TABLESPACE users3 DATAFILE SIZE 1G AUTOEXTEND ON MAXSIZE 10G;
```

次の例では、一時テーブルスペース *temp01* を作成します。

```
CREATE TEMPORARY TABLESPACE temp01;
```

ALTER TABLESPACE を使用すると、bigfile テーブルスペースをサイズ変更できます。サイズは、キロバイト (K)、メガバイト (M)、ギガバイト (G)、またはテラバイト (T) で指定できます。次の例では、bigfile テーブルスペース *users_bf* のサイズを 200 MB に変更します。

```
ALTER TABLESPACE users_bf RESIZE 200M;
```

次の例では、smallfile テーブルスペース *users_sf* に追加のデータファイルを追加します。

```
ALTER TABLESPACE users_sf ADD DATAFILE SIZE 100000M AUTOEXTEND ON NEXT 250m
 MAXSIZE UNLIMITED;
```

RDS for Oracle で追加のストレージボリュームにテーブルスペースを作成する

追加のストレージボリュームにテーブルスペースを作成するには、DB_CREATE_FILE_DEST パラメータをボリュームの場所に変更します。次の例では、ファイルの場所を /rdsdbdata2/db に設定します。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata2/db';

Session altered.
```

次の例では、追加のボリューム /rdsdbdata2/db にテーブルスペースを作成します。

```
CREATE TABLESPACE new_tablespace DATAFILE SIZE 10G;

Tablespace created.

SELECT tablespace_name,file_id,file_name FROM dba_data_files
WHERE tablespace_name = 'NEW_TABLESPACE';

TABLESPACE_NAME              FILE_ID FILE_NAME
------------------------- ----------
 --------------------------------------------------------------------------------
NEW_TABLESPACE                      7 /rdsdbdata2/db/ORCL_A/datafile/
o1_mf_newtable_a123b4c5_.dbf
```

smallfile テーブルスペースを作成し、そのデータファイルを異なるストレージボリュームに分散するには、作成後にテーブルスペースにデータファイルを追加します。次の例では、/rdsdbdata/db のデフォルトの場所にデータファイルを含むテーブルスペースを作成します。次に、デフォルトの送信先を /rdsdbdata2/db に設定します。新しく作成したテーブルスペースにデータファイルを追加すると、データベースはそのファイルを /rdsdbdata2/db に保存します。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata/db';

Session altered.

CREATE SMALLFILE TABLESPACE smalltbs DATAFILE SIZE 10G;

Tablespace created.

SELECT tablespace_name,file_id,file_name FROM dba_data_files
WHERE tablespace_name = 'SMALLTBS';
```

```
TABLESPACE_NAME              FILE_ID FILE_NAME
------------------------- ----------
 --------------------------------------------------------------------------------
SMALLTBS                           8 /rdsdbdata/db/ORCL_A/datafile/
o1_mf_smalltbs_n563yryk_.dbf

ALTER SESSION SET db_create_file_dest = '/rdsdbdata2/db';

Session altered.

ALTER TABLESPACE smalltbs ADD DATAFILE SIZE 10G;

Tablespace altered.

SELECT tablespace_name,file_id,file_name FROM dba_data_files
WHERE tablespace_name = 'SMALLTBS';

TABLESPACE_NAME              FILE_ID FILE_NAME
------------------------- ----------
 --------------------------------------------------------------------------------
SMALLTBS                           8 /rdsdbdata/db/ORCL_A/datafile/
o1_mf_smalltbs_n563yryk_.dbf
SMALLTBS                           9 /rdsdbdata2/db/ORCL_A/datafile/
o1_mf_smalltbs_n564004g_.dbf
```

RDS for Oracle でのデフォルトのテーブルスペースの設定

デフォルトのテーブルスペースを設定するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.alter_default_tablespace` を使用します。`alter_default_tablespace` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`tablespace_name`	varchar	—	はい	デフォルトのテーブルスペースの名前。

次の例では、デフォルトのテーブルスペースを *`users2`* に設定します。

```
EXEC rdsadmin.rdsadmin_util.alter_default_tablespace(tablespace_name => 'users2');
```

デフォルトのテーブルスペース設定を確認するには、次のクエリを実行します。

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE PROPERTY_NAME =
 'DEFAULT_PERMANENT_TABLESPACE';
```

RDS for Oracle でのデフォルトの一時テーブルスペースの設定

デフォルトのテンポラリテーブルスペースを設定するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.alter_default_temp_tablespace` を使用します。`alter_default_temp_tablespace` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`tablespace_name`	varchar	—	はい	デフォルトのテンポラリテーブルスペースの名前。

次の例では、デフォルトのテンポラリテーブルスペースを *`temp01`* に設定します。

```
EXEC rdsadmin.rdsadmin_util.alter_default_temp_tablespace(tablespace_name => 'temp01');
```

インスタンスストアに一時テーブルスペースを作成する

インスタンスストアに一時テーブルスペースを作成するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace` を使用します。`create_inst_store_tmp_tblspace` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_tablespace_name`	varchar	—	はい	テンポラリテーブルスペースの名前。

次の例では、インスタンスストアに一時テーブルスペース *`temp01`* を作成します。

```
EXEC rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace(p_tablespace_name =>
 'temp01');
```

> **Important**
>
> `rdsadmin_util.create_inst_store_tmp_tblspace` を実行しても、新しく作成された一時テーブルスペースは、デフォルトの一時テーブルスペースとして自動的に設定されません。これをデフォルトとして設定するには、「RDS for Oracle でのデフォルトの一時テーブルスペースの設定」を参照してください。

詳細については、「RDS for Oracle インスタンスストアへの一時データの保存」を参照してください。

RDS for Oracle で一時ファイルを使用する

リードレプリカのインスタンスストアへの一時ファイルの追加

プライマリ DB インスタンスで一時テーブルスペースを作成すると、リードレプリカの一時ファイルは作成されません。次のいずれかの理由により、リードレプリカに空の一時テーブルスペースが存在するとします。

- リードレプリカのテーブルスペースから一時ファイルを削除しました。詳細については、「リードレプリカの一時ファイルの削除」を参照してください。
- プライマリ DB インスタンスに新しい一時テーブルスペースを作成しました。この場合、RDS for Oracle はメタデータをリードレプリカに同期します。

一時ファイルを空の一時テーブルスペースに追加し、その一時ファイルをインスタンスストアに保存できます。インスタンスストアに一時ファイルを作成するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.add_inst_store_tempfile` を使用します。この手順はリードレプリカでのみ使用できます。 プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_tablespace_name`	varchar	—	はい	リードレプリカの一時テーブルスペースの名前。

次の例では、空の一時テーブルスペース *`temp01`* がリードレプリカに存在します。次のコマンドを実行して、このテーブルスペースの一時ファイルを作成し、インスタンスストアに保存します。

```
EXEC rdsadmin.rdsadmin_util.add_inst_store_tempfile(p_tablespace_name => 'temp01');
```

詳細については、「RDS for Oracle インスタンスストアへの一時データの保存」を参照してください。

リードレプリカの一時ファイルの削除

既存の一時テーブルスペースをリードレプリカから削除することはできません。リードレプリカの一時ファイルストレージを Amazon EBS からインスタンスストアに、またはインスタンスストアから Amazon EBS に変更できます。これらの目標を達成するには、次の操作を行います。

1. リードレプリカの一時テーブルスペースにある現在の一時ファイルを削除します。
2. 別のストレージに新しい一時ファイルを作成します。

一時ファイルを削除するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.drop_replica_tempfiles` を使用します。この手順はリードレプリカでのみ使用できます。`drop_replica_tempfiles` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_tablespace_name	varchar	—	はい	リードレプリカの一時テーブルスペースの名前。

temp01 という名前の一時テーブルスペースがリードレプリカのインスタンスストアにあると仮定します。次のコマンドを実行して、このテーブルスペースのすべての一時ファイルを削除します。

```
EXEC rdsadmin.rdsadmin_util.drop_replica_tempfiles(p_tablespace_name => 'temp01');
```

詳細については、「RDS for Oracle インスタンスストアへの一時データの保存」を参照してください。

RDS for Oracle のテーブルスペース、データファイル、一時ファイルのサイズ変更

デフォルトでは、Oracle テーブルスペースは自動エクステンションをオンにして作成され、最大サイズでは作成されません。これらのデフォルト設定のため、場合によってはテーブルスペースが大き

くなりすぎる可能性があります。永続テーブルスペースとテンポラリテーブルスペースに適切な最大サイズを指定し、リージョンの使用状況を注意深くモニタリングすることをお勧めします。

永続テーブルスペースの変更

RDS for Oracle DB インスタンスの永続テーブルスペースのサイズを変更するには、以下の Amazon RDS 手順のいずれかを使用してください。

- `rdsadmin.rdsadmin_util.resize_datafile`
- `rdsadmin.rdsadmin_util.autoextend_datafile`

`resize_datafile` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_data_file_id`	number	—	はい	サイズ変更するデータファイルの識別子。
`p_size`	varchar2	—	はい	データファイルのサイズ。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。

`autoextend_datafile` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_data_file_id`	number	—	はい	サイズ変更するデータファイルの識別子。
`p_autoextend_state`	varchar2	—	はい	自動拡張機能の状態。`ON` を指定するとデータファイルは自動的に拡張し、`OFF` を指定すると自

パラメータ名	データ型	デフォルト	必須	説明
				動拡張はオフになります。
p_next	varchar2	—	いいえ	次のデータファイル増分のサイズ。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。
p_maxsize	varchar2	—	いいえ	自動拡張に許可される最大ディスク容量。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。UNLIMITED を指定するとファイルサイズ制限を削除できます。

次の例では、データファイル 4 のサイズを 500 MB に変更します。

```
EXEC rdsadmin.rdsadmin_util.resize_datafile(4,'500M');
```

次の例では、データファイル 4 の自動拡張をオフにします。また、データファイル 5 の自動拡張はオンになり、増分は 128 MB で、最大サイズはありません。

```
EXEC rdsadmin.rdsadmin_util.autoextend_datafile(4,'OFF');
EXEC rdsadmin.rdsadmin_util.autoextend_datafile(5,'ON','128M','UNLIMITED');
```

一時テーブルスペースのサイズ変更

RDS for Oracle DB インスタンスの一時テーブルスペース (リードレプリカを含む) のサイズを変更するには、以下の Amazon RDS 手順のいずれかを使用します。

- rdsadmin.rdsadmin_util.resize_temp_tablespace

- rdsadmin.rdsadmin_util.resize_tempfile
- rdsadmin.rdsadmin_util.autoextend_tempfile

resize_temp_tablespace プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_temp_tablespace_name	varchar2	—	はい	サイズを変更するテンポラリテーブルスペースの名前。
p_size	varchar2	—	はい	テーブルスペースのサイズ。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。

resize_tempfile プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_temp_file_id	number	—	はい	サイズを変更する一時ファイルの識別子。
p_size	varchar2	—	はい	一時ファイルのサイズ。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。

autoextend_tempfile プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_temp_file_id	number	—	はい	サイズを変更する一時ファイルの識別子。
p_autoextend_state	varchar2	—	はい	自動拡張機能の状態。ON を指定すると一時ファイルは自動的に拡張し、OFF を指定すると自動拡張はオフになります。
p_next	varchar2	—	いいえ	次の一時ファイル増分のサイズ。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。
p_maxsize	varchar2	—	いいえ	自動拡張に許可される最大ディスク容量。サイズをバイト (デフォルト)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) で指定します。UNLIMITED を指定するとファイルサイズ制限を削除できます。

次の例は、TEMP という一時テーブルスペースのサイズを 4 ギガバイトに変更します。

```
EXEC rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4G');
```

```
EXEC rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4294967296');
```

次の例は、ファイル識別子が 1 の一時ファイルに基づいて、一時テーブルスペースのサイズを 2 メガバイトに変更します。

```
EXEC rdsadmin.rdsadmin_util.resize_tempfile(1,'2M');
```

次の例では、一時ファイル 1 の自動拡張をオフにします。また、一時ファイル 2 の最大自動拡張サイズも 10 ギガバイト、増分を 100 メガバイトに設定します。

```
EXEC rdsadmin.rdsadmin_util.autoextend_tempfile(1,'OFF');
EXEC rdsadmin.rdsadmin_util.autoextend_tempfile(2,'ON','100M','10G');
```

Oracle DB インスタンスのリードレプリカの詳細については、「Amazon RDS for Oracle でのリードレプリカの使用」を参照してください。

RDS for Oracle のストレージボリューム間のデータの移動

データファイルとデータベースオブジェクトは、プライマリストレージボリュームと追加のストレージボリューム間で移動できます。データを移動する前に、次の点を考慮してください。

- ソースボリュームとターゲットボリュームには十分な空き領域が必要です。
- データ移動オペレーションは、両方のボリュームで I/O を消費します。
- 大きなデータ移動は、データベースのパフォーマンスに影響を与える可能性があります。
- スナップショットを復元する場合、EBS 遅延ロードの影響を受けると、ストレージボリューム間のデータの移動が遅くなる可能性があります。

トピック

- RDS for Oracle のボリューム間でのデータファイルの移動
- RDS for Oracle のボリューム間でテーブルデータとインデックスを移動する
- 追加ボリュームを使用した LOB ストレージの管理

RDS for Oracle のボリューム間でのデータファイルの移動

ストレージボリューム間でデータファイルを移動するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.move_datafile` を使用します。次の要件に注意してください。

- `move_datafile` プロシージャを実行するには、Oracle Enterprise Edition を使用する必要があります。

- テーブルスペース SYSTEM と RDSADMIN を移動することはできません。

move_datafile プロシージャには以下のパラメータがあります。

パラメータ名	データ型	必要	説明
p_data_file_id	number	はい	移動するデータファイルの ID。
p_location	varchar2	はい	データファイルを移動するストレージボリューム。

次の例では、テーブルスペースをデフォルトボリューム rdsdbdata から追加ボリューム rdsdbdata2 に移動します。

```
SQL> SELECT tablespace_name,file_id,file_name FROM dba_data_files
 WHERE tablespace_name = 'MYNEWTABLESPACE';

TABLESPACE_NAME              FILE_ID FILE_NAME
------------------------- ----------
 --------------------------------------------------------------------------------
MYNEWTABLESPACE                    6 /rdsdbdata/db/ORCL_A/datafile/
o1_mf_mynewtab_n123abcd_.dbf

EXECUTE rdsadmin.rdsadmin_util.move_datafile( 6, 'rdsdbdata2');

PL/SQL procedure successfully completed.

SQL> SELECT tablespace_name,file_id,file_name FROM dba_data_files
  WHERE tablespace_name = 'MYNEWTABLESPACE';

TABLESPACE_NAME              FILE_ID FILE_NAME
------------------------- ----------
 --------------------------------------------------------------------------------
MYNEWTABLESPACE                    6 /rdsdbdata2/db/ORCL_A/datafile/
o1_mf_mynewtab_n356efgh_.dbf
```

RDS for Oracle のボリューム間でテーブルデータとインデックスを移動する

追加のストレージボリュームにテーブルスペースを作成することで、データベースストレージを最適化できます。その後、標準の Oracle SQL を使用して、テーブル、インデックス、パーティションなどのオブジェクトをこれらのテーブルスペースに移動できます。このアプローチは、データベースにさまざまなアクセスパターンのデータが含まれている場合のパフォーマンス調整に役立ちます。例えば、アクセス頻度の高い運用データを高性能ストレージボリュームに保存しながら、アクセス頻度の低い履歴データを低コストのストレージボリュームに移動できます。

次の例では、高性能ボリューム rdsdbdata2 に新しいテーブルスペースを作成します。次に、テーブルがオンラインのときに、テーブルを追加のストレージボリュームに移動します。また、インデックスを同じボリュームに移動します。オンライン中にテーブルを移動してインデックスを再構築するには、Oracle Enterprise Edition が必要です。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata2/db';
CREATE TABLESPACE perf_tbs DATAFILE SIZE 10G;

ALTER TABLE employees
  MOVE TABLESPACE perf_tbs ONLINE;

ALTER INDEX employees_idx
  REBUILD ONLINE TABLESPACE perf_tbs;
```

次の例では、低コストのボリュームにテーブルスペースを作成します。次に、オンラインオペレーションを使用して、テーブルパーティションを低コストのストレージボリュームに移動します。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata3/db';
CREATE TABLESPACE hist_tbs DATAFILE SIZE 10G;

ALTER TABLE orders
  MOVE PARTITION orders_2022
  TABLESPACE hist_tbs ONLINE;
```

次の例では、アクティブなセッションのロングオペレーションをクエリします。

```
SELECT sid,opname,sofar,totalwork,time_remaining,elapsed_seconds
  FROM v$session_longops
  WHERE time_remaining > 0;
```

次のクエリを使用して、テーブルスペースの使用状況を確認できます。

```
SELECT tablespace_name, used_percent
  FROM dba_tablespace_usage_metrics
  ORDER BY used_percent DESC;
```

追加ボリュームを使用した LOB ストレージの管理

データベースには、大量のストレージを消費するが、アクセス頻度が低い BLOB オブジェクトまたは CLOB オブジェクトを含むテーブルが含まれている場合があります。ストレージを最適化するために、これらの LOB セグメントを追加のストレージボリュームのテーブルスペースに再配置できます。

次の例では、低アクセスデータを対象とした低コストのボリュームで LOB データのテーブルスペースを作成します。次に、このボリュームにデータを保存するテーブルを作成します。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata3/db';
CREATE TABLESPACE lob_data DATAFILE SIZE 5G AUTOEXTEND ON NEXT 1G;

CREATE TABLE documents (
    doc_id NUMBER PRIMARY KEY,
    doc_date DATE,
    doc_content CLOB
) TABLESPACE user_data
LOB(doc_content) STORE AS (TABLESPACE lob_data);
```

RDS for Oracle での外部テーブルの使用

Oracle 外部テーブルは、データベースにないデータのテーブルです。代わりに、データはデータベースがアクセスできる外部ファイルにあります。外部テーブルを使用して、データベースにロードしないでデータにアクセスできます。外部テーブルの詳細については、Oracle ドキュメントの「外部表の管理」を参照してください。

Amazon RDS では、外部テーブルファイルをディレクトリオブジェクトに保存できます。ディレクトリオブジェクトは、新規作成するか、Oracle データベースに定義済みのもの (DATA_PUMP_DIR ディレクトリなど) を使用できます。ディレクトリオブジェクトの作成方法については、「主要データストレージ領域でのディレクトリの作成と削除」を参照してください。Amazon RDS Oracle DB インスタンスのディレクトリオブジェクトを一覧表示するには、ALL_DIRECTORIES ビューをクエリできます。

> **Note**
>
> ディレクトリオブジェクトは、インスタンスで使用されているメインのデータストレージ領域 (Amazon EBS ボリューム) を参照します。データファイル、REDO ログ、監査、追跡、およびその他のファイルで使用される領域は、割り当てられたストレージの消費としてカウントされます。

外部データファイルを Oracle データベース間で移行するには、DBMS_FILE_TRANSFER パッケージまたは UTL_FILE パッケージを使用できます。外部データファイルは、移行元のデータベースのディレクトリから、移行先のデータベースの指定されたディレクトリに移行されます。DBMS_FILE_TRANSFER の使用の詳細については、「Oracle Data Pump を使用したインポート」を参照してください。

移行した外部データファイルを使用して、外部テーブルを作成できます。次の例では、USER_DIR1 ディレクトリの emp_xt_file1.txt ファイルを使用する外部テーブルを作成します。

```
CREATE TABLE emp_xt (
  emp_id      NUMBER,
  first_name  VARCHAR2(50),
  last_name   VARCHAR2(50),
  user_name   VARCHAR2(20)
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY USER_DIR1
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE
    FIELDS TERMINATED BY ','
    MISSING FIELD VALUES ARE NULL
    (emp_id,first_name,last_name,user_name)
  )
  LOCATION ('emp_xt_file1.txt')
)
PARALLEL
REJECT LIMIT UNLIMITED;
```

Amazon RDS Oracle DB インスタンスにあるデータを外部データファイル内に移行するとします。この場合、外部テーブルを作成し、データベースのテーブルから選択したデータを外部データファイ

ルに入力できます。例えば、次の SQL ステートメントでは、データベースの orders_xt テーブルをクエリすることで orders 外部テーブルを作成します。

```
CREATE TABLE orders_xt
  ORGANIZATION EXTERNAL
   (
     TYPE ORACLE_DATAPUMP
     DEFAULT DIRECTORY DATA_PUMP_DIR
     LOCATION ('orders_xt.dmp')
   )
   AS SELECT * FROM orders;
```

この例では、データは DATA_PUMP_DIR ディレクトリの orders_xt.dmp ファイルに入力されます。

データベースのチェックポイント機能

データベースのチェックポイントを作成するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.checkpoint を使用します。checkpoint プロシージャにはパラメータはありません。

次の例では、データベースのチェックポイントを作成します。

```
EXEC rdsadmin.rdsadmin_util.checkpoint;
```

分散復旧の設定

分散復旧を設定するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.enable_distr_recovery と disable_distr_recovery を使用します。このプロシージャにはパラメータはありません。

次の例では、分散復旧を有効にします。

```
EXEC rdsadmin.rdsadmin_util.enable_distr_recovery;
```

次の例では、分散復旧を無効にします。

```
EXEC rdsadmin.rdsadmin_util.disable_distr_recovery;
```

データベースタイムゾーンの設定

以下の方法で、Amazon RDS Oracle データベースのタイムゾーンを設定できます。

- `Timezone` オプション

  `Timezone` オプションは、ホストレベルでタイムゾーンを変更し、SYSDATE など、すべての日付列と値に影響します。詳細については、「Oracle のタイムゾーン」を参照してください。
- Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.alter_db_time_zone`

  `alter_db_time_zone` プロシージャは、特定のデータ型のみのタイムゾーンを変更し、SYSDATE は変更しません。タイムゾーンの設定に関する他の制限については、Oracle ドキュメントに示されています。

> **Note**
>
> Oracle Scheduler のデフォルトのタイムゾーンを設定することもできます。詳細については、「Oracle Scheduler ジョブのタイムゾーンの設定」を参照してください。

`alter_db_time_zone` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_new_tz	varchar2	—	はい	指定したリージョンまたは協定世界時 (UTC) からの絶対オフセットとしての新しいタイムゾーン。有効なオフセットの範囲は -12:00 ～ +14:00 です。

以下の例では、タイムゾーンを UTC + 3 時間に変更します。

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => '+3:00');
```

以下の例では、タイムゾーンをアフリカ/アルジェーのタイムゾーンに変更します。

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => 'Africa/Algiers');
```

alter_db_time_zone プロシージャを使用してタイムゾーンを変更した後、変更を有効にするために DB インスタンスを再起動します。詳細については、「DB インスタンスの再起動」を参照してください。タイムゾーンのアップグレードについては、「タイムゾーンに関する考慮事項」を参照してください。

自動ワークロードリポジトリ (AWR) を使用したパフォーマンスレポートの生成

パフォーマンスデータを収集してレポートを生成するにあたって、Oracle では自動ワークロードリポジトリ (AWR) を推奨しています。AWR には、Oracle Database Enterprise Edition と診断パックおよびチューニングパックのライセンスが必要です。AWR を有効にするには、初期化パラメータ CONTROL_MANAGEMENT_PACK_ACCESS に、DIAGNOSTIC または DIAGNOSTIC+TUNING を設定します。

RDS での AWR レポートの使用

AWR レポートを生成するには、awrrpt.sql などのスクリプトを実行します。これらのスクリプトは、データベースホストサーバーにインストールされます。Amazon RDS では、ホストに直接アクセスすることはできません。ただし、Oracle Database の別のインストールから SQL スクリプトのコピーを取得することは可能です。

また、SYS.DBMS_WORKLOAD_REPOSITORY PL/SQL パッケージでプロシージャを実行して AWR を使用することもできます。このパッケージを使用して、ベースラインとスナップショットを管理したり、ASH および AWR レポートを表示したりできます。例えば、テキスト形式で AWR レポートを生成するには、DBMS_WORKLOAD_REPOSITORY.AWR_REPORT_TEXT プロシージャを実行します。ただし、これらの AWR レポートには、AWS マネジメントコンソール からはアクセスできません。

AWR を使用する場合は、rdsadmin.rdsadmin_diagnostic_util プロシージャを使用することをお勧めします。これらのプロシージャを使用すると、次の項目を生成できます。

- AWR レポート
- アクティブセッション履歴 (ASH) レポート
- 自動データベース診断モニター (ADDM) レポート
- AWR データの Oracle Data Pump Export ダンプファイル

rdsadmin_diagnostic_util プロシージャは、レポートを DB インスタンスファイルシステムに保存します。これらのレポートには、コンソールからアクセスできます。rdsadmin.rds_file_util プロシージャを使用してレポートにアクセスしたり、S3 統合オプションを使用して Amazon S3 にコピーされたレポートにアクセスすることもできます。詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」および「「Amazon S3 統合」」を参照してください。

以下の Amazon RDS for Oracle DB エンジンバージョンの rdsadmin_diagnostic_util 手順を使用できます。

- すべての Oracle Database 26ai バージョン
- すべての Oracle Database 21c バージョン
- 19.0.0.0.ru-2020-04.rur-2020-04.r1 以上の Oracle Database 19c バージョン

レプリケーションシナリオで診断レポートを使用する方法を説明しているブログについては、「Amazon RDS for Oracle リードレプリカの AWR レポートを生成する」を参照してください。

Diagnostic Utility Package の一般的なパラメータ

rdsadmin_diagnostic_util パッケージを使用して AWR および ADDM を管理する場合は、通常、次のパラメータを使用します。

パラメータ	データ型	デフォルト	必須	説明
begin_snap_id	NUMBEI	—	はい	スタートスナップショットの ID。
end_snap_id	NUMBEI	—	はい	終了スナップショットの ID。
dump_directory	VARCHA	BDUMI	いいえ	レポートまたはエクスポートファイルの書き込み先のディレクトリ。デフォルト以外のディレクトリを指定する場合、 rdsadmin_diagnostic_util プロシージャを実行するユーザーは、そのディレクトリに対する書き込み権限を持っている必要があります。

パラメータ	データ型	デフォルト	必須	説明
p_tag	VARCHA	—	いいえ	レポート出力ファイルを識別するタグ。最大 30 文字を指定できます。有効な文字は、a-z、A-Z、0-9、アンダースコア (_)、ダッシュ (-)、およびピリオド (.) です。
report_type	VARCHA	HTML	いいえ	レポートの形式。有効な値は、TEXT および HTML です。
dbid	NUMBEF	—	いいえ	Oracle の DBA_HIST_DATABASE_INSTANCE ビューに表示される有効なデータベース識別子 (DBID)。このパラメータを指定しない場合、RDS は V$DATABASE.DBID ビューに表示される現在の DBID を使用します。

rdsadmin_diagnostic_util パッケージで ASH を管理する場合は、通常、次のパラメータを使用します。

Parameter	データ型	デフォルト	必須	説明
begin_time	DATE	—	はい	ASH 分析のスタート時刻。
end_time	DATE	—	はい	ASH 分析の終了時刻。
slot_width	NUMBER	0	いいえ	ASH レポートの「Top Activity」セクションで使用されるスロットの期間 (秒単位)。このパラメータを指定しない場合、begin_time と end_time と間の時間間隔は 10 以下のスロットを使用します。
sid	NUMBER	Null	いいえ	セッション ID。

Parameter	データ型	デフォルト	必須	説明
sql_id	VARCHAR2	Null	いいえ	SQL ID。
wait_class	VARCHAR2	Null	いいえ	待機クラス名。
service_hash	NUMBER	Null	いいえ	サービス名のハッシュ。
module_name	VARCHAR2	Null	いいえ	モジュール名。
action_name	VARCHAR2	Null	いいえ	アクション名。
client_id	VARCHAR2	Null	いいえ	データベースセッションのアプリケーション固有の ID。
plsql_entry	VARCHAR2	Null	いいえ	PL/SQL エントリポイント。

AWR レポートの生成

AWR レポートを生成するには、rdsadmin.rdsadmin_diagnostic_util.awr_report の手順を使用します。

AWR レポートの有効なスナップショット ID を検索するには、次のクエリを実行します。

```
SELECT SNAP_ID, BEGIN_INTERVAL_TIME, END_INTERVAL_TIME FROM DBA_HIST_SNAPSHOT ORDER BY
 SNAP_ID DESC FETCH FIRST 10 ROWS ONLY;
```

次の例は、スナップショット範囲 101 ~ 106 の AWR レポートを生成します。出力テキストファイルには awrrpt_101_106.txt という名前が付けられます。このレポートには、AWS マネジメントコンソール からアクセスできます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_report(101,106,'TEXT');
```

次の例は、スナップショット範囲 63～65 の HTML レポートを生成します。出力 HTML ファイルには awrrpt_63_65.html という名前が付けられます。プロシージャによって、レポートが AWR_RPT_DUMP という名前のデフォルト以外のデータベースディレクトリに書き込まれます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_report(63,65,'HTML','AWR_RPT_DUMP');
```

ダンプファイルへの AWR データの抽出

AWR データをダンプファイルに抽出するには、rdsadmin.rdsadmin_diagnostic_util.awr_extract プロシージャを使用します。この関数は PDB レベルでのみ使用できます。

次の例は、スナップショット範囲 101～106 を抽出します。出力ダンプファイルには awrextract_101_106.dmp という名前が付けられます。このファイルには、コンソールからアクセスできます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_extract(101,106);
```

次の例は、スナップショット範囲 63～65 を抽出します。出力ダンプファイルには awrextract_63_65.dmp という名前が付けられます。ファイルは、AWR_RPT_DUMP という名前のデフォルト以外のデータベースディレクトリに保存されます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.awr_extract(63,65,'AWR_RPT_DUMP');
```

ADDM レポートの生成

ADDM レポートを生成するには、rdsadmin.rdsadmin_diagnostic_util.addm_report プロシージャを使用します。

次の例は、スナップショット範囲 101～106 の ADDM レポートを生成します。出力テキストファイルには addmrpt_101_106.txt という名前が付けられます。レポートには、コンソールからアクセスできます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.addm_report(101,106);
```

次の例は、スナップショット範囲 63～65 の ADDM レポートを生成します。出力テキストファイルには addmrpt_63_65.txt という名前が付けられます。ファイルは、ADDM_RPT_DUMP という名前のデフォルト以外のデータベースディレクトリに保存されます。

```
EXEC rdsadmin.rdsadmin_diagnostic_util.addm_report(63,65,'ADDM_RPT_DUMP');
```

ASH レポートの生成

ASH レポートを生成するには、rdsadmin.rdsadmin_diagnostic_util.ash_report プロシージャを使用します。

次の例では、14 分前から現在の時刻までのデータを含む ASH レポートを生成します。出力ファイルの名前は ashrpt*begin_time*_*end_time*.txt の形式を使用し、*begin_time* および *end_time* は YYYYMMDDHH24MISS の形式を使用します。ファイルには、コンソールからアクセスできます。

```
BEGIN
    rdsadmin.rdsadmin_diagnostic_util.ash_report(
        begin_time     =>     SYSDATE-14/1440,
        end_time       =>     SYSDATE,
        report_type    =>     'TEXT');
END;
/
```

次の例では、2019 年 9 月 18 日の午後 6:07 から 2019 年 9 月 18 日の午後 6:15 までのデータを含む ASH レポートを生成します。出力 HTML レポートの名前は ashrpt_20190918180700_20190918181500.html です。レポートは、AWR_RPT_DUMP という名前のデフォルト以外のデータベースディレクトリに保存されます。

```
BEGIN
    rdsadmin.rdsadmin_diagnostic_util.ash_report(
        begin_time     =>    TO_DATE('2019-09-18 18:07:00', 'YYYY-MM-DD HH24:MI:SS'),
        end_time       =>    TO_DATE('2019-09-18 18:15:00', 'YYYY-MM-DD HH24:MI:SS'),
        report_type    =>    'html',
        dump_directory =>    'AWR_RPT_DUMP');
END;
/
```

コンソールまたは CLI からの AWR レポートへのアクセス

AWR レポートにアクセスしたり、ダンプファイルをエクスポートしたりするには、AWS マネジメントコンソール または AWS CLI を使用します。詳細については、「データベースログファイルのダウンロード」を参照してください。

VPC の DB インスタンスで使用するデータベースリンクの調整

同じ Virtual Private Cloud (VPC) 内またはピア接続された VPC 内の Amazon RDS DB インスタンスで Oracle データベースリンクを使用するには、2 つの DB インスタンス間に有効なルートが存在する必要があります。VPC ルーティングテーブルおよびネットワークアクセス制御リスト (ACL) を使用して、DB インスタンス間の有効なルートを確認します。

各 DB インスタンスのセキュリティグループは他の DB インスタンスの受信と送信を許可する必要があります。インバウンドルールとアウトバウンドルールは、同じ VPC またはピアリング接続先 VPC からセキュリティグループを参照できます。詳細については、「セキュリティグループの更新によるピア VPC セキュリティグループの参照」を参照してください。

VPC で DHCP オプションセットを使用してカスタム DNS サーバーを設定した場合、カスタム DNS サーバーはデータベースリンクターゲットの名前を解決できる必要があります。詳細については、「カスタム DNS サーバーのセットアップ」を参照してください。

Oracle Data Pump でのデータベースリンクの使用の詳細については、「Oracle Data Pump を使用したインポート」を参照してください。

DB インスタンスのデフォルトエディションの設定

データベースオブジェクトは、エディションと呼ばれるプライベート環境で再定義できます。エディションベースの再定義を使用して、最小限のダウンタイムでアプリケーションのデータベースオブジェクトをアップグレードできます。

Amazon RDS Oracle DB インスタンスのデフォルトエディションは、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.alter_default_edition` を使用して設定できます。

次の例では、Amazon RDS Oracle DB インスタンスのデフォルトエディションを `RELEASE_V1` に設定します。

```
EXEC rdsadmin.rdsadmin_util.alter_default_edition('RELEASE_V1');
```

次の例では、Amazon RDS Oracle DB インスタンスのデフォルトエディションを Oracle のデフォルトに設定し直します。

```
EXEC rdsadmin.rdsadmin_util.alter_default_edition('ORA$BASE');
```

Oracle エディションベースの再定義の詳細については、Oracle ドキュメントの「Managing editions」を参照してください。

SYS.AUD$ テーブルの監査を有効にする

データベースの監査証跡テーブル SYS.AUD$ の監査を有効にするには、Amazon RDS プロシージャ rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table を使用します。サポートされている監査プロパティは、ALL のみです。個々のステートメントまたはオペレーションは監査することもしないこともできません。

監査の有効化は、次のバージョンを実行している Oracle DB インスタンスでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

audit_all_sys_aud_table プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_by_access	boolean	true	いいえ	true を監査するには、BY ACCESS に設定します。false を監査するには、BY SESSION に設定します。

以下のクエリでは、データベースの SYS.AUD$ に対する現在の監査設定が返ります。

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE OWNER='SYS' AND OBJECT_NAME='AUD$';
```

以下のコマンドでは、ALL SYS.AUD$ の BY ACCESS の監査を有効にします。

```
EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table;

EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table(p_by_access => true);
```

以下のコマンドでは、ALL SYS.AUD$ の BY SESSION の監査を有効にします。

```
EXEC rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table(p_by_access => false);
```

詳細については、Oracle ドキュメントの「AUDIT (従来の監査)」を参照してください。

SYS.AUD$ テーブルの監査を無効にする

データベースの監査証跡テーブル SYS.AUD$ の監査を無効にするには、Amazon RDS プロシージャ rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table を使用します。この手順では、パラメータは使用しません。

以下のクエリでは、データベースの SYS.AUD$ に対する現在の監査設定が返ります。

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE OWNER='SYS' AND OBJECT_NAME='AUD$';
```

以下のコマンドでは、ALL の SYS.AUD$ の監査が無効にされます。

```
EXEC rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table;
```

詳細については、Oracle ドキュメントの「NOAUDIT (従来の監査)」を参照してください。

中断したオンラインインデックス構築のクリーンアップ

失敗したオンラインインデックス構築をクリーンアップするには、Amazon RDS プロシージャ rdsadmin.rdsadmin_dbms_repair.online_index_clean を使用します。

online_index_clean プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
object_id	binary_integer	ALL_INDEX_ID	いいえ	インデックスのオブジェクト ID。通常、ORA

パラメータ名	データ型	デフォルト	必須	説明
				-08104 エラーテキストからのオブジェクト ID を使用できます。
wait_for_lock	binary_integer	rdsadmin.rdsadmin_dbms_repair.lock_wait	いいえ	デフォルトの rdsadmin.rdsadmin_dbms_repair.lock_wait を指定して、基盤となるオブジェクトのロックを取得することを試みます。ロックが失敗した場合は、内部制限に達するまで再試行できます。 rdsadmin.rdsadmin_dbms_repair.lock_nowait を指定して、基盤となるオブジェクトのロックを取得することを試みることができますが、ロックが失敗すると再試行はできません。

以下の例は、失敗したオンラインインデックス構築のクリーンアップを示しています。

```
declare
  is_clean boolean;
begin
  is_clean := rdsadmin.rdsadmin_dbms_repair.online_index_clean(
    object_id     => 1234567890,
    wait_for_lock => rdsadmin.rdsadmin_dbms_repair.lock_nowait
  );
end;
/
```

詳細については、Oracle ドキュメントの「ONLINE_INDEX_CLEAN Function」を参照してください。

破損ブロックのスキップ

インデックスとテーブルのスキャンの中に破損ブロックをスキップするには、rdsadmin.rdsadmin_dbms_repair パッケージを使用します。

以下の手順では、sys.dbms_repair.admin_table 手順の機能をラップし、パラメータを取得しません。

- rdsadmin.rdsadmin_dbms_repair.create_repair_table
- rdsadmin.rdsadmin_dbms_repair.create_orphan_keys_table
- rdsadmin.rdsadmin_dbms_repair.drop_repair_table
- rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table
- rdsadmin.rdsadmin_dbms_repair.purge_repair_table
- rdsadmin.rdsadmin_dbms_repair.purge_orphan_keys_table

以下の手順では、Oracle データベースの DBMS_REPAIR パッケージの対応物として、同じパラメータを取得します。

- rdsadmin.rdsadmin_dbms_repair.check_object
- rdsadmin.rdsadmin_dbms_repair.dump_orphan_keys
- rdsadmin.rdsadmin_dbms_repair.fix_corrupt_blocks
- rdsadmin.rdsadmin_dbms_repair.rebuild_freelists
- rdsadmin.rdsadmin_dbms_repair.segment_fix_status
- rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks

データベースの破損の処理の詳細については、Oracle のドキュメントの「DBMS_REPAIR」を参照してください。

Example破損ブロックへの対応

この例では、破損ブロックに応答するための基本的なワークフローを示しています。ステップは、ブロックの破損の場所と性質によって異なります。

> **Important**
>
> 破損ブロックを修復する前に、「DBMS_REPAIR」ドキュメントをよく確認してください。

インデックスとテーブルのスキャン中に破損ブロックをスキップするには

1. 修復テーブルが存在しない場合は、以下のプロシージャを実行して作成します。

```
EXEC rdsadmin.rdsadmin_dbms_repair.create_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.create_orphan_keys_table;
```

2. 適切な場合は、以下の手順を実行して、既存のレコードを確認し、消去します。

```
SELECT COUNT(*) FROM SYS.REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.ORPHAN_KEY_TABLE;
SELECT COUNT(*) FROM SYS.DBA_REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.DBA_ORPHAN_KEY_TABLE;

EXEC rdsadmin.rdsadmin_dbms_repair.purge_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.purge_orphan_keys_table;
```

3. 以下の手順を実行して、破損ブロックを確認します。

```
SET SERVEROUTPUT ON
DECLARE v_num_corrupt INT;
BEGIN
  v_num_corrupt := 0;
  rdsadmin.rdsadmin_dbms_repair.check_object (
    schema_name => '&corruptionOwner',
    object_name => '&corruptionTable',
    corrupt_count =>  v_num_corrupt
  );
  dbms_output.put_line('number corrupt: '||to_char(v_num_corrupt));
END;
/

COL CORRUPT_DESCRIPTION FORMAT a30
COL REPAIR_DESCRIPTION FORMAT a30

SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_TYPE, MARKED_CORRUPT,
       CORRUPT_DESCRIPTION, REPAIR_DESCRIPTION
```

```
FROM   SYS.REPAIR_TABLE;

SELECT SKIP_CORRUPT
FROM   DBA_TABLES
WHERE  OWNER = '&corruptionOwner'
AND    TABLE_NAME = '&corruptionTable';
```

4. skip_corrupt_blocks プロシージャを使用して、該当するテーブルの破損スキップを有効または無効にします。状況によっては、新しいテーブルにデータを抽出してから、破損ブロックを含むテーブルを削除することが必要になる場合もあります。

   以下の手順を実行して、該当するテーブルの破損スキップを有効にします。

```
begin
  rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks (
    schema_name => '&corruptionOwner',
    object_name => '&corruptionTable',
    object_type => rdsadmin.rdsadmin_dbms_repair.table_object,
    flags => rdsadmin.rdsadmin_dbms_repair.skip_flag);
end;
/
select skip_corrupt from dba_tables where owner = '&corruptionOwner' and table_name
 = '&corruptionTable';
```

   以下の手順を実行して、破損スキップを無効にします。

```
begin
  rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks (
    schema_name => '&corruptionOwner',
    object_name => '&corruptionTable',
    object_type => rdsadmin.rdsadmin_dbms_repair.table_object,
    flags => rdsadmin.rdsadmin_dbms_repair.noskip_flag);
end;
/

select skip_corrupt from dba_tables where owner = '&corruptionOwner' and table_name
 = '&corruptionTable';
```

5. すべての修復作業が完了したら、以下の手順を実行して修復テーブルを削除します。

```
EXEC rdsadmin.rdsadmin_dbms_repair.drop_repair_table;
EXEC rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table;
```

ごみ箱を空にする

テーブルを削除しても、Oracle データベースはただちにストレージ領域を削除しません。データベースによってテーブルの名前が変更され、テーブルと関連オブジェクトがごみ箱に入れられます。ごみ箱を空にすると、これらのアイテムが削除され、ストレージ領域が解放されます。

ごみ箱全体を空にするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.purge_dba_recyclebin` を使用します。ただし、このプロシージャでは、 SYS および RDSADMIN オブジェクトのごみ箱は空になりません。これらのオブジェクトを消去する必要がある場合は、AWS サポートにお問い合わせください。

次の例は、ごみ箱全体を空にします。

```
EXEC rdsadmin.rdsadmin_util.purge_dba_recyclebin;
```

フルリダクションのデフォルト表示値の設定

Amazon RDS Oracle インスタンスでの完全リダクションのデフォルトの表示値を変更するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val` を使用します。リダクションポリシーは、Oracle データベースのドキュメントで説明されているように、DBMS_REDACT PL/SQL パッケージを使用して作成することに注意してください。dbms_redact_upd_full_rdct_val プロシージャは、既存のポリシーの影響を受けるさまざまなデータ型について表示する文字を指定します。

`dbms_redact_upd_full_rdct_val` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_number_val	数値	Null	いいえ	NUMBER データ型の列のデフォルト値を変更します。
p_binfloat_val	binary_float	Null	いいえ	BINARY_FLOAT データ型の列のデフォルト値を変更します。

パラメータ名	データ型	デフォルト	必須	説明
p_bindouble_val	binary_double	Null	いいえ	BINARY_DOUBLE データ型の列のデフォルト値を変更します。
p_char_val	char	Null	いいえ	CHAR データ型の列のデフォルト値を変更します。
p_varchar_val	varchar2	Null	いいえ	VARCHAR2 データ型の列のデフォルト値を変更します。
p_nchar_val	nchar	Null	いいえ	NCHAR データ型の列のデフォルト値を変更します。
p_nvarchar_val	nvarchar2	Null	いいえ	NVARCHAR2 データ型の列のデフォルト値を変更します。
p_date_val	date	Null	いいえ	DATE データ型の列のデフォルト値を変更します。
p_ts_val	timestamp	Null	いいえ	TIMESTAMP データ型の列のデフォルト値を変更します。
p_tswtz_val	timestamp with time zone	Null	いいえ	TIMESTAMP WITH TIME ZONE データ型の列のデフォルト値を変更します。
p_blob_val	blob	Null	いいえ	BLOB データ型の列のデフォルト値を変更します。

パラメータ名	データ型	デフォルト	必須	説明
p_clob_val	clob	Null	いいえ	CLOB データ型の列のデフォルト値を変更します。
p_nclob_val	nclob	Null	いいえ	NCLOB データ型の列のデフォルト値を変更します。

次の例では、CHAR データ型のデフォルトのリダクション値を * に変更します。

```
EXEC rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val(p_char_val => '*');
```

次の例では、NUMBER、DATE、および CHAR データ型のデフォルトのリダクション値を変更します。

```
BEGIN
rdsadmin.rdsadmin_util.dbms_redact_upd_full_rdct_val(
    p_number_val=>1,
    p_date_val=>to_date('1900-01-01','YYYY-MM-DD'),
    p_varchar_val=>'X');
END;
/
```

dbms_redact_upd_full_rdct_val プロシージャを使用してフルリダクションのデフォルト値を変更した後、変更を有効にするために DB インスタンスを再起動します。詳細については、「DB インスタンスの再起動」を参照してください。

Oracle DB インスタンスの一般的なログ関連タスクの実行

次に、Oracle を実行している Amazon RDS DB インスタンスへのログ記録に関連する一般的な DBA タスクの実行方法を示します。マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスは提供していません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。

詳細については、「Amazon RDS for Oracle データベースのログファイル」を参照してください。

トピック

- 強制ログ作成の設定
- サプリメンタルロギングの設定
- オンラインログファイルを切り替える
- オンライン REDO ログの追加
- オンライン REDO ログの削除
- オンライン REDO ログのサイズ変更
- アーカイブ REDO ログの保持
- オンライン およびアーカイブ REDO ログへのアクセス
- Amazon S3 からのアーカイブ REDO ログのダウンロード

強制ログ作成の設定

強制ログ作成モードでは、Oracle はテンポラリテーブルスペースとテンポラリセグメントの変更を除き、すべての変更をデータベースに記録します (NOLOGGING 句は無視されます)。詳細については、Oracle ドキュメントの「FORCE LOGGING モードの指定」を参照してください。

強制ログ作成を設定するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.force_logging` を使用します。`force_logging` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_enable`	boolean	true	いいえ	データベースを強制ログ作成モードに設定するには `true`、データベースを強制ログ作成モードから解除するには `false` に設定します。

次の例では、データベースを強制ログ作成モードに設定します。

```
EXEC rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

強制ログ記録が有効になっていることを確認するには、次のクエリを実行します。

```
SELECT FORCE_LOGGING FROM V$DATABASE;
```

サプリメンタルロギングの設定

補足的なログ記録を有効にすると、LogMiner には、チェーンされた行とクラスター化されたテーブルをサポートするために必要な情報が表示されます。サプリメンタルロギングの詳細については、Oracle ドキュメントの「サプリメンタルロギング」を参照してください。

Oracle データベースでは、デフォルトではサプリメンタルロギングが有効になっていません。サプリメンタルロギングを有効/無効にするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.alter_supplemental_logging` を使用します。Amazon RDS が Oracle DB インスタンスのアーカイブ REDO ログの保持を管理する方法の詳細については、「アーカイブ REDO ログの保持」を参照してください。

`alter_supplemental_logging` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_action`	varchar2	—	はい	サプリメンタルロギングを追加するには `'ADD'`、サプリメンタルロギングを削除するには `'DROP'`。
`p_type`	varchar2	null	いいえ	サプリメンタルロギングのタイプ。有効な値は、`'ALL'`、`'FOREIGN KEY'`、`'PRIMARY KEY'`、`'UNIQUE'`、`'PROCEDURAL'` のいずれかです。

次の例では、補足ログを有効にします。

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD');
end;
```

```
/
```

次の例では、すべての固定長の最大サイズの列に対して補足ログを有効にします。

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'ALL');
end;
/
```

次の例では、プライマリキー列に対して補足ログを有効にします。

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'PRIMARY KEY');
end;
/
```

サプリメンタルロギングが有効になっていることを確認するには、次のクエリを実行します。

```
SELECT SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI,
 SUPPLEMENTAL_LOG_DATA_FK, SUPPLEMENTAL_LOG_DATA_ALL FROM V$DATABASE;
```

オンラインログファイルを切り替える

ログファイルを切り替えるには、Amazon RDS のプロシージャ `rdsadmin.rdsadmin_util.switch_logfile` を使用します。`switch_logfile` プロシージャにはパラメータはありません。

次の例では、ログファイルを切り替えています。

```
EXEC rdsadmin.rdsadmin_util.switch_logfile;
```

オンライン REDO ログの追加

Oracle を実行している Amazon RDS DB インスタンスには、初期から 4 つのオンライン REDO ログ (それぞれ 128 MB) があります。別の REDO ログを追加するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.add_logfile` を使用します。

add_logfile プロシージャには以下のパラメータがあります。

> **Note**
>
> パラメータは相互に排他的です。

パラメータ名	データ型	デフォルト	必須	説明
bytes	positive	null	いいえ	ログファイルのサイズ (バイト単位)。 このパラメータは、ログのサイズが 2,147,483,648 バイト (2 GiB) 未満の場合にのみ使用します。それ以外の場合、RDS はエラーを発行します。このバイト値を超えるログサイズの場合は、代わりに p_size パラメータを使用します。
p_size	varchar2	—	いいえ	ログファイルのサイズは、キロバイト (K)、メガバイト (M)、またはギガバイト (G) です。

> **Note**
>
> bytes または p_size のどちらかを指定する必要があります。両方を指定した場合、プロシージャは p_size を使用します。

次のコマンドは 100 MB のログファイルを追加します。

```
EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '100M');
```

オンライン REDO ログの削除

REDO ログを削除するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.drop_logfile` を使用します。`drop_logfile` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
grp	positive	—	はい	ログのグループ番号。

次の例では、グループ番号 3 のログを削除します。

```
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
```

ステータスが未使用または非アクティブのログのみを削除できます。次の例では、ログのステータスを取得します。

```
SELECT GROUP#, STATUS FROM V$LOG;

GROUP#     STATUS
---------- ----------------
1          CURRENT
2          INACTIVE
3          INACTIVE
4          UNUSED
```

オンライン REDO ログのサイズ変更

Oracle を実行している Amazon RDS DB インスタンスには、初期から 4 つのオンライン REDO ログ (それぞれ 128 MB) があります。次の例では、Amazon RDS プロシージャを使用してログをそれぞれ 128 MB から 512 MB にサイズ変更する方法を示します。

```
/* Query V$LOG to see the logs.          */
/* You start with 4 logs of 128 MB each. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
```

```
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE


/* Add four new logs that are each 512 MB */

EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '512M');
EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '512M');
EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '512M');
EXEC rdsadmin.rdsadmin_util.add_logfile(p_size => '512M');


/* Query V$LOG to see the logs. */
/* Now there are 8 logs.        */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Drop each inactive log using the group number. */

EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 1);
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 4);


/* Query V$LOG to see the logs. */
/* Now there are 5 logs.        */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
```

```
2          134217728  CURRENT
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Switch logs so that group 2 is no longer current. */

EXEC rdsadmin.rdsadmin_util.switch_logfile;


/* Query V$LOG to see the logs.        */
/* Now one of the new logs is current. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  ACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* If the status of log 2 is still "ACTIVE", issue a checkpoint to clear it to
 "INACTIVE".  */

EXEC rdsadmin.rdsadmin_util.checkpoint;


/* Query V$LOG to see the logs.            */
/* Now the final original log is inactive. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  INACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED
```

```
/* Drop the final inactive log. */

EXEC rdsadmin.rdsadmin_util.drop_logfile(grp => 2);


/* Query V$LOG to see the logs.    */
/* Now there are four 512 MB logs. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED
```

> **Tip**
>
> チェックポイントの後もログステータスが ACTIVE のままの場合、長時間実行されるトランザクションがそのログを保持している可能性があります。クエリ V$TRANSACTION を実行してアクティブなトランザクションを識別し、完了後にチェックポイントを再試行します。

アーカイブ REDO ログの保持

アーカイブ REDO ログを、Oracle LogMiner (DBMS_LOGMNR) などの製品で使用できるように DB インスタンスにローカルで保持できます。REDO ログを保持した後、LogMiner を使用してログを分析できます。詳細については、Oracle ドキュメントの「LogMiner を使用した REDO ログファイルの分析」を参照してください。

アーカイブされた REDO ログを保持するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.set_configuration` を使用します。この手順を Oracle Data Guard のプライマリインスタンスで使用すると、RDS はプライマリインスタンスとオープンリードレプリカでのアーカイブログ保持設定を変更しますが、マウントされたレプリカでは変更しません。RDS は、マウントされたレプリカに最新のアーカイブ REDO ログを短期間保持します。RDS は、マウントされたレプリカにダウンロードされた古いログを自動的に削除します。

`set_configuration` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
name	varchar	—	はい	更新する設定の名前。アーカイブ REDO ログの保持時間を変更するには、名前を archivelog retention hours に設定します。
value	varchar	—	はい	設定の値。値をログを保持する時間数に設定します。

次の例では、24 時間分の REDO ログを保持します。

```
begin
    rdsadmin.rdsadmin_util.set_configuration(
        name  => 'archivelog retention hours',
        value => '24');
end;
/
commit;
```

> **Note**
>
> 変更を反映するにはコミットが必要です。

アーカイブ REDOログが DB インスタンスに保持される期間を表示するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_util.show_configuration を使用します。

次の例は、ログの保持時間を示しています。

```
set serveroutput on
EXEC rdsadmin.rdsadmin_util.show_configuration;
```

出力は、archivelog retention hours の現在の設定を示します。次の出力は、アーカイブ REDO ログを 48 時間保持することを示しています。

```
NAME:archivelog retention hours
VALUE:48
DESCRIPTION:ArchiveLog expiration specifies the duration in hours before archive/redo
 log files are automatically deleted.
```

アーカイブ REDO ログは、DB インスタンス上に保持されるので、DB インスタンスにログ用の十分な割り当て済みストレージがあることを確認します。DB インスタンスが過去 X 時間に使用した容量を調べるには、次のクエリを実行できます。X は時間数に置き換えます。

```
SELECT SUM(BLOCKS * BLOCK_SIZE) bytes
  FROM V$ARCHIVED_LOG
 WHERE FIRST_TIME >= SYSDATE-(X/24) AND DEST_ID=1;
```

DB インスタンスのバックアップ保持期間にゼロより大きな値が設定されている場合のみ、RDS for Oracle がアーカイブ REDO ログを生成します。デフォルトでは、バックアップ保持期間は 0 より大きな値になっています。

アーカイブされたログの保持期間が終了すると、RDS for Oracle はアーカイブされた REDO ログを、DB インスタンスから削除します。DB インスタンスを特定の時点の状態に復元できるようにするために、Amazon RDS は、アーカイブされた REDO ログを、バックアップ保持期間に基づいて DB インスタンスの外部に保持します。バックアップ保持期間の変更方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> アーカイブ REDO ログをダウンロードするために JDBC を Linux で使用している場合、レイテンシー時間と接続のリセットが長くなることがあります。このような場合、原因は Java クライアントの乱数ジェネレーターのデフォルト設定にある可能性があります。ブロックしない乱数ジェネレーターを使用するように JDBC ドライバを設定することをお勧めします。

オンライン およびアーカイブ REDO ログへのアクセス

GoldenGate、Attunity、Informatica などの外部ツールを使用して、オンライン REDO ログファイルやアーカイブ REDO ログファイルにアクセスすることがあります。これらのファイルにアクセスするには、次の手順を実行します。

1. 物理ファイルパスへの読み取り専用アクセスを提供する、ディレクトリオブジェクトを作成します。

`rdsadmin.rdsadmin_master_util.create_archivelog_dir` および `rdsadmin.rdsadmin_master_util.create_onlinelog_dir` を使用します。

2. PL/SQL を使用してファイルを読み取ります。

   ファイルは、PL/SQL により読み取ることができます。ディレクトリオブジェクトからファイルを読み込む方法については、「DB インスタンスディレクトリ内のファイルのリスト化」および「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

トランザクションログへのアクセスがサポートされるのは、以下のリリースです。

- Oracle Database 26ai
- Oracle Database 21c
- Oracle Database 19c

以下のコードでは、オンライン REDO ログファイルやアーカイブ REDO ログファイルに読み取り専用アクセスを提供するディレクトリが作成されます。

> **Important**
>
> また、このコードによって、`DROP ANY DIRECTORY` 権限は取り消されます。

```
EXEC rdsadmin.rdsadmin_master_util.create_archivelog_dir;
EXEC rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

次のコードでは、オンラインの REDO ログファイルやアーカイブ REDO ログファイルのディレクトリを削除します。

```
EXEC rdsadmin.rdsadmin_master_util.drop_archivelog_dir;
EXEC rdsadmin.rdsadmin_master_util.drop_onlinelog_dir;
```

次のコードでは、`DROP ANY DIRECTORY` 権限の付与と取り消しを行います。

```
EXEC rdsadmin.rdsadmin_master_util.revoke_drop_any_directory;
EXEC rdsadmin.rdsadmin_master_util.grant_drop_any_directory;
```

Amazon S3 からのアーカイブ REDO ログのダウンロード

`rdsadmin.rdsadmin_archive_log_download` パッケージを使用して、アーカイブ REDO ログを DB インスタンスにダウンロードできます。アーカイブ REDO ログが DB インスタンス上から失われた場合は、Amazon S3 から再度ダウンロードすることが可能です。その後、ログを取り出したり、データベースの復旧またはレプリケート用に使用したりできます。

> **Note**
>
> アーカイブされた REDO ログは、リードレプリカインスタンスにダウンロードすることはできません。

アーカイブ RED Oログのダウンロード:基本的なステップ

アーカイブ REDO ログを使用できるかどうかは、以下に挙げる保存ポリシーによって異なります。

- バックアップ保持ポリシー - このポリシー内のログは Amazon S3 で使用できます。このポリシー外のログは削除されています。
- アーカイブログ保持ポリシー - このポリシー内のログは、DB インスタンスで使用できます。このポリシー外のログは削除されています。

インスタンスにログがなく、バックアップ保持期間によって保護されている場合は、`rdsadmin.rdsadmin_archive_log_download` を使用して、再度ダウンロードしてください。RDS for Oracle は、ログをDB インスタンスの `/rdsdbdata/log/arch` ディレクトリに保存します。

Amazon S3 からアーカイブ REDO ログをダウンロードするには

1. 保持期間を設定して、ダウンロードしたアーカイブ REDO ログを必要な期間保持します。忘れずに変更を `COMMIT` します。

   RDS は、ダウンロードしたログを、ログがダウンロードされた時点から、アーカイブされたログ保持ポリシーに従って保持します。保持ポリシーを設定する方法については、「アーカイブ REDO ログの保持」を参照してください。
2. アーカイブログの保持ポリシーに対する変更が有効になるまで、最大 5 分待機します。
3. `rdsadmin.rdsadmin_archive_log_download` により、アーカイブ REDO ログを Amazon S3 からダウンロードします。

詳細については、「単一のアーカイブ REDO ログのダウンロード」および「一連のアーカイブ REDO ログのダウンロード」を参照してください。

> **Note**
>
> RDS は、ダウンロードの実行前に、利用可能なストレージを自動的に確認します。要求されたログで大量の領域が消費される場合は、アラートが表示されます。

4. Amazon S3 からのログのダウンロードが、正常に実行されたことを確認します。

   ダウンロードタスクのステータスは、bdump ファイルで確認できます。bdump ファイルには、パス名 /rdsdbdata/log/trace/dbtask-*task-id*.log が含まれています。前出のダウンロードステップで実行した SELECT ステートメントでは、タスク ID が VARCHAR2 データ型で返されます。詳細については、「ファイル転送のステータスをモニタリングする」で類似の例をご確認ください。

単一のアーカイブ REDO ログのダウンロード

単一のアーカイブ REDO ログを /rdsdbdata/log/arch ディレクトリにダウンロードするには、rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum を使用します。このプロシージャには次のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
seqnum	number	—	はい	アーカイブ REDO ログのシーケンス番号。

アーカイブされた有効な REDO ログシーケンス番号を検索するには、次のクエリを実行します。

```
SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME FROM V$ARCHIVED_LOG WHERE DEST_ID = 1 ORDER BY
 SEQUENCE# DESC FETCH FIRST 20 ROWS ONLY;
```

次の例では、シーケンス番号 20 のログをダウンロードしています。

```
SELECT rdsadmin.rdsadmin_archive_log_download.download_log_with_seqnum(seqnum => 20)
       AS TASK_ID
FROM   DUAL;
```

一連のアーカイブ REDO ログのダウンロード

一連のアーカイブ REDO ログを /rdsdbdata/log/arch ディレクトリにダウンロードするには、download_logs_in_seqnum_range を使用します。ダウンロードできるログの数は、リクエストごとに 300 に制限されています。download_logs_in_seqnum_range プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
start_seq	number	—	はい	全体のスタートシーケンス番号。
end_seq	number	—	はい	全体の終了シーケンス番号。

次の例では、シーケンス番号が 50 から 100 までのログをダウンロードしています。

```
SELECT rdsadmin.rdsadmin_archive_log_download.download_logs_in_seqnum_range(start_seq
 => 50, end_seq => 100)
       AS TASK_ID
FROM   DUAL;
```

Oracle DB インスタンスの一般的な RMAN タスクの実行

次のセクションでは、Oracle を実行している Amazon RDS DB インスタンスで Oracle Recovery Manager (RMAN) DBA タスクを実行する方法について説明します。マネージドサービスエクスペリエンスを提供するために、Amazon RDS は DB インスタンスへのシェルアクセスを提供していません。また、高度な特権を必要とする特定のシステムプロシージャやテーブルへのアクセスを制限しています。

Amazon RDS パッケージ rdsadmin.rdsadmin_rman_util を使用して、Amazon RDS for Oracle データベースのディスクへの RMAN バックアップを実行します。rdsadmin.rdsadmin_rman_util パッケージは、データベースファイル、テーブルスペース、およびアーカイブされた REDO ログの完全バックアップと増分バックアップをサポートしています。

RMAN バックアップが完了したら、バックアップファイルを Amazon RDS for Oracle DB インスタンスホスト以外にコピーできます。これを行うのは、RDS ホスト以外に復元してバックアップを長

期保存するためです。例えば、バックアップファイルを Amazon S3 バケットにコピーできます。詳細については、「「Amazon S3 統合」」を参照してください。

RMAN バックアップのバックアップファイルは、手動で削除するまでは、Amazon RDS DB インスタンスホストに残ります。ディレクトリからファイルを削除するには、Oracle プロシージャ `UTL_FILE.FREMOVE` を使用できます。詳細については、Oracle データベースドキュメントの「FREMOVE プロシージャ」を参照してください。

RMAN を使用して RDS for Oracle DB インスタンスを復元することはできません。ただし、RMAN を使用して、バックアップをオンプレミスまたは Amazon EC2 インスタンスに復元することはできます。詳細については、ブログ記事「Amazon RDS for Oracle インスタンスをセルフマネージドインスタンスに復元する」を参照してください。

> **Note**
>
> Amazon RDS for Oracle の別の DB インスタンスにバックアップして復元するには、引き続き Amazon RDS のバックアップおよび復元機能を使用できます。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

トピック

- RMAN バックアップの前提条件
- RMAN プロシージャの共通パラメータ
- RDS for Oracle でのデータベースファイルの検証
- ブロック変更追跡の有効化/無効化
- アーカイブ REDO ログのクロスチェック
- アーカイブ REDO ログファイルのバックアップ
- データベースの完全バックアップの実行
- テナントデータベースの完全バックアップの実行
- データベースの増分バックアップの実行
- テナントデータベースの増分バックアップの実行
- テーブルスペースのバックアップ
- コントロールファイルのバックアップ
- ブロックメディアリカバリの実行

RMAN バックアップの前提条件

`rdsadmin.rdsadmin_rman_util` パッケージを使用してデータベースをバックアップする前に、以下の前提条件を満たしていることを確認してください。

- RDS for Oracle データベースが `ARCHIVELOG` モードであることを確認してください。このモードを有効にするには、バックアップ保持期間をゼロ以外の値に設定します。
- アーカイブされた REDO ログをバックアップしたり、アーカイブされた REDO ログを含む完全バックアップまたは増分バックアップを実行する場合は、REDO ログの保持期間をゼロ以外の値に設定する必要があります。リカバリ時にデータベースファイルの整合性を保つには、アーカイブされたREDO ログが必要です。詳細については、「アーカイブ REDO ログの保持」を参照してください。
- DB インスタンスにバックアップを保持するのに十分な空き容量があることを確認してください。データベースをバックアップするときには、プロシージャ呼び出しのパラメータとして Oracle ディレクトリオブジェクトを指定します。RMAN は、指定されたディレクトリにファイルを配置します。デフォルトのディレクトリ (`DATA_PUMP_DIR` など) を使用するか、新しいディレクトリを作成できます。詳細については、「主要データストレージ領域でのディレクトリの作成と削除」を参照してください。

  CloudWatch メトリクス `FreeStorageSpace` を使用して、RDS for Oracle インスタンスの現在の空き容量をモニタリングできます。RMAN はフォーマットされたブロックのみをバックアップし、圧縮をサポートしますが、データベースの現在のサイズを超える空き容量を確保しておくことをお勧めします。

RMAN プロシージャの共通パラメータ

RMAN のタスクを実行するには、Amazon RDS パッケージ `rdsadmin.rdsadmin_rman_util` を使用できます。いくつかのパラメータは、パッケージ内のすべてのプロシージャに共通です。パッケージ内の共通パラメータは以下のとおりです。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`p_directory_name`	varchar	データベースの有効な	—	はい	バックアップファイルを含めるディレクトリの名前。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
		ディレクトリ名。			
p_label	varchar	a-z, A-Z, 0-9, '_', '-', '.'	—	いいえ	バックアップファイル名に含まれている一意の文字列。 **Note** 上限は 30 文字です。
p_owner	varchar	p_directory_name に指定されたディレクトリの有効な所有者。	—	はい	バックアップファイルを含めるディレクトリの所有者。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_tag	varchar	a-z, A-Z, 0-9, '_', '-', '.'	NULL	いいえ	日次、週次、増分レベルのバックアップなど、バックアップの目的または使用方法を示すためにバックアップを区別できるようにする目的で使用する文字列。 上限は 30 文字です。タグでは、大文字と小文字は区別されません。タグは、入力する際に使用されるのが大文字か小文字かに関係なく、常に大文字で格納されます。 タグは一意である必要はないので、複数のバックアップに同じタグを付けることができます。 タグを指定しない場合、RMAN は、TAG*YYYYMMDDTHHMMSS* の形式を使用してデフォルトのタグを自動的に割り当てます。ここで *YYYY* は年、*MM* は月、*DD* は日、*HH* は時間 (24 時間形式)、*MM* は分、*SS* は秒です。日付と時刻は、RMAN がバックアップを開始した日時を示します。 例えば、2019-09-27 の 21:45:17 に開始されたバックアップに対して、バックアップに TAG20190927T214517 というタグが付けられることがあります。 p_tag パラメータは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
					• Oracle Database 26ai (26.0.0) • Oracle Database 21c (21.0.0) • 19.0.0.0.ru-2021-10.rur-2021-10.r1 以降を使用する、Oracle Database 19c (19.0.0)
p_compress	boolear	TRUE, FALSE	FALSE	いいえ	BASIC バックアップ圧縮を行うには TRUE を指定します。 BASIC バックアップ圧縮を無効にするには FALSE を指定します。
p_include_archive_logs	boolear	TRUE, FALSE	FALSE	いいえ	アーカイブ REDO ログをバックアップに含めるには、TRUE を指定します。 アーカイブ REDO ログをバックアップから除外するには、FALSE を指定します。 アーカイブ REDO ログをバックアップに含める場合は、rdsadmin.rdsadmin_util.set_configuration プロシージャを使用して保持期間を 1 時間以上に設定します。また、バックアップを実行する直前に rdsadmin.rdsadmin_rman_util.crosscheck_archivelog プロシージャを呼び出します。そうしないと、アーカイブ REDO ログファイルが Amazon RDS 管理プロシージャで削除されて見つからないため、バックアップは失敗する場合があります。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_include_controlfile	boolean	TRUE, FALSE	FALSE	いいえ	制御ファイルをバックアップに含めるには、TRUE を指定します。 制御ファイルをバックアップから除外するには、FALSE を指定します。
p_optimize	boolean	TRUE, FALSE	TRUE	いいえ	バックアップの最適化を有効にするには、TRUE を指定します。アーカイブ REDO ログを含める場合は、バックアップサイズが縮小されます。 バックアップの最適化を無効にするには、FALSE を指定します。
p_parallel	number	Oracle Database Enterprise Edition (EE) の場合は 1〜254 の有効な整数 1その他の Oracle データベースエディションの場合は	1	いいえ	チャネルの数。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	いいえ	TRUE の場合、RMAN 出力は DBMS_OUTPUT ディレクトリのファイルに加えて、BDUMP パッケージに送信されます。SQL*Plus で、SET SERVEROUTPUT ON を使用して出力を確認します。 FALSE の場合、RMAN 出力は BDUMP ディレクトリのファイルにのみ送信されます。
p_section_size_mb	number	有効な整数	NULL	いいえ	セクションサイズのメガバイト (MB)。 各ファイルを指定されたセクションサイズに分割し、パラレルして検証します。 NULL の場合、パラメータは無視されます。
p_validation_type	varchar	'PHYSICAL', 'PHYSICAL+LOGICAL'	'PHYS'	いいえ	破損の検出のレベル。 物理的な破損を確認するには、'PHYSICAL' を指定します。物理的な破損の一例は、ヘッダーおよびフッターの不一致によるブロックです。 物理的な破損に加えて論理的な不整合を確認するには、'PHYSICAL+LOGICAL' を指定します。論理的な破損の一例は破損ブロックです。

RDS for Oracle でのデータベースファイルの検証

Amazon RDS パッケージ `rdsadmin.rdsadmin_rman_util` を使用して、Amazon RDS for Oracle のデータベースファイル (データファイル、テーブルスペース、制御ファイル、およびサーバーパラメータファイル (SPFILE) など) を検証できます。

RMAN の検証の詳細については、Oracle ドキュメントの「データベースファイルおよびバックアップの検証」と「VALIDATE」を参照してください。

トピック

- データベースの検証
- テナントデータベースの検証
- テーブルスペースの検証
- 制御ファイルの検証
- SPFILE の検証
- Oracle データファイルの検証

データベースの検証

RDS の Oracle データベースで使用されているすべての関連ファイルを検証するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.validate_database` を使用します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_validation_type`
- `p_parallel`
- `p_section_size_mb`
- `p_rman_to_dbms_output`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

次の例では、パラメータのデフォルト値を使用してデータベースを検証します。

```
EXEC rdsadmin.rdsadmin_rman_util.validate_database;
```

次の例では、パラメータの指定値を使用してデータベースを検証します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_database(
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
/
```

p_rman_to_dbms_output パラメータを FALSE に設定すると、RMAN 出力は BDUMP ディレクトリのファイルに書き込まれます。

BDUMP ディレクトリのファイルを表示するには、次の SELECT ステートメントを実行します。

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

BDUMP ディレクトリのファイルの内容を表示するには、次の SELECT ステートメントを実行します。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','rds-rman-
validate-nnn.txt'));
```

ファイル名を、表示するファイルの名前に置き換えます。

テナントデータベースの検証

コンテナデータベース (CDB) 内のテナントデータベースのデータファイルを検証するには、Amazon RDS のプロシージャ rdsadmin.rdsadmin_rman_util.validate_tenant を使用します。

このプロシージャは現在のテナントデータベースにのみ適用され、以下の RMAN タスクの共通パラメータを使用します。

- p_validation_type
- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。このプロシージャは、以下の DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0) CDB
- Oracle Database 21c (21.0.0) CDB
- Oracle Database 19c (19.0.0) CDB

次の例では、パラメータのデフォルト値を使用して現行のテナントデータベースを検証します。

```
EXEC rdsadmin.rdsadmin_rman_util.validate_tenant;
```

次の例では、パラメータの指定値を使用して現行のテナントデータベースを検証します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_tenant(
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
/
```

p_rman_to_dbms_output パラメータを FALSE に設定すると、RMAN 出力は BDUMP ディレクトリのファイルに書き込まれます。

BDUMP ディレクトリのファイルを表示するには、次の SELECT ステートメントを実行します。

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

BDUMP ディレクトリのファイルの内容を表示するには、次の SELECT ステートメントを実行します。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','rds-rman-
validate-nnn.txt'));
```

ファイル名を、表示するファイルの名前に置き換えます。

テーブルスペースの検証

テーブルスペースに関連付けられたファイルを検証するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_rman_util.validate_tablespace を使用します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- p_validation_type
- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、次の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_tablespace_name	varchar2	有効なテーブルスペース名	—	はい	テーブルスペースの名前。

制御ファイルの検証

Amazon RDS Oracle DB インスタンスで使用されている制御ファイルのみを検証するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_rman_util.validate_current_controlfile を使用します。

このプロシージャでは、次の RMAN タスクの共通パラメータを使用します。

- p_validation_type
- p_rman_to_dbms_output

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

SPFILE の検証

Amazon RDS Oracle DB インスタンスで使用されているサーバーパラメータファイル (SPFILE) を検証するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_rman_util.validate_spfile を使用します。

このプロシージャでは、次の RMAN タスクの共通パラメータを使用します。

- p_validation_type
- p_rman_to_dbms_output

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

Oracle データファイルの検証

データファイルを検証するには、Amazon RDS プロシージャ rdsadmin.rdsadmin_rman_util.validate_datafile を使用します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- p_validation_type
- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、以下の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_datafile	varchar2	有効なデータファイル ID 番号または完全なパスを含む有効なデータファイル名	—	はい	データファイル ID 番号 (v$datafile.file# より) またはパスを含む詳細なデータファイル名 (v$datafile.name より)。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_from_block	number	有効な整数	NULL	いいえ	データファイル内で検証が始まるブロックの番号。これが NULL の場合、1 が使用されます。
p_to_block	number	有効な整数	NULL	いいえ	データファイル内で検証が終わるブロックの番号。これが NULL の場合、データファイルの最大ブロックが使用されます。

ブロック変更追跡の有効化/無効化

ブロック変更の追跡は、トラッキングファイル内の変更されたブロックを記録します。この手法により、RMAN 増分バックアップのパフォーマンスを向上させることができます。詳細は、Oracle Database のドキュメントの「ブロック変更の追跡を使用した増分バックアップのパフォーマンスの向上」を参照してください。

RMAN 機能はリードレプリカではサポートされていません。ただし、高可用性戦略の一環として、手順 rdsadmin.rdsadmin_rman_util.enable_block_change_tracking に従って読み取り専用レプリカでブロック追跡を有効にすることもできます。この読み取り専用レプリカをソース DB インスタンスに昇格すると、新しいソースインスタンスでブロック変更追跡が有効になります。そのため、インスタンスは高速増分バックアップの恩恵を受けることができます。

ブロック変更の追跡手順は、次の DB エンジンバージョンの Enterprise Edition でのみサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

> **Note**
>
> シングルテナント CDB では、以下のオペレーションは機能しますが、お客様が表示可能なメカニズムはオペレーションの現在のステータスを検出できません。「RDS for Oracle CDB の制限事項」も参照してください。

DB インスタンスのブロック変更の追跡を有効にするには、Amazon RDS 手順 `rdsadmin.rdsadmin_rman_util.enable_block_change_tracking` を使用します。ブロック変更の追跡を無効にするには、`disable_block_change_tracking`を使用します。これらのプロシージャではパラメータを使用しません。

使用している DB インスタンスでブロック変更追跡が有効になっているかどうかを確認するには、次のクエリを実行します。

```
SELECT STATUS, FILENAME FROM V$BLOCK_CHANGE_TRACKING;
```

次の例では、DB インスタンスのブロック変更追跡を有効にします。

```
EXEC rdsadmin.rdsadmin_rman_util.enable_block_change_tracking;
```

次の例では、DB インスタンスのブロック変更追跡を無効にします。

```
EXEC rdsadmin.rdsadmin_rman_util.disable_block_change_tracking;
```

アーカイブ REDO ログのクロスチェック

Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.crosscheck_archivelog` を使用して、アーカイブ REDO ログをクロスチェックできます。

このプロシージャを使用して、制御ファイルに登録済みのアーカイブ REDO ログをクロスチェックし、必要に応じて期限切れのログレコードを削除できます。RMAN によってバックアップが作成されると、制御ファイルにレコードが作成されます。時間の経過に伴って、これらのレコードのために制御ファイルのサイズが肥大します。期限切れのレコードは定期的に削除することをお勧めします。

> **Note**
>
> スタンダードの Amazon RDS バックアップでは RMAN を使用しないため、レコードは制御ファイルに作成されません。

このプロシージャでは、RMAN タスクの共通パラメータ `p_rman_to_dbms_output` を使用します。

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、次の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`p_delete_expired`	boolean	`TRUE`, `FALSE`	`TRUE`	いいえ	`TRUE` の場合、期限切れのアーカイブ REDO ログレコードを制御ファイルから削除します。 `FALSE` の場合、期限切れのアーカイブ REDO ログレコードを制御ファイルに保持します。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

次の例では、制御ファイル内のアーカイブ REDO ログレコードを期限切れとしてマークしますが、レコードは削除しません。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.crosscheck_archivelog(
        p_delete_expired      => FALSE,
        p_rman_to_dbms_output => FALSE);
END;
/
```

次の例では、期限切れのアーカイブ REDO ログレコードを制御ファイルから削除します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.crosscheck_archivelog(
        p_delete_expired      => TRUE,
        p_rman_to_dbms_output => FALSE);
END;
/
```

アーカイブ REDO ログファイルのバックアップ

Amazon RDS パッケージ `rdsadmin.rdsadmin_rman_util` を使用して Amazon RDS Oracle DB インスタンスのアーカイブ REDO ログをバックアップできます。

アーカイブ REDO ログをバックアップするプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

トピック

- すべてのアーカイブ REDO ログのバックアップ
- 日付範囲からのアーカイブ REDO ログのバックアップ
- SCN 範囲からのアーカイブ REDO ログのバックアップ
- シーケンス番号範囲からのアーカイブ REDO ログのバックアップ

すべてのアーカイブ REDO ログのバックアップ

Amazon RDS Oracle DB インスタンスのすべてのアーカイブ REDO ログをバックアップするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_archivelog_all` を使用します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

次の例では、DB インスタンスのすべてのアーカイブ REDO ログをバックアップします。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_all(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_tag                 => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

日付範囲からのアーカイブ REDO ログのバックアップ

日付範囲を指定して Amazon RDS Oracle DB インスタンスの特定のアーカイブ REDO ログをバックアップするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_archivelog_date` を使用します。日付範囲では、どのアーカイブ REDO ログをバックアップするかを指定します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_compress
- p_rman_to_dbms_output
- p_tag

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、以下の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_from_date	date	ディスクにあるアーカイブ REDO ログの start_date ~next_ の日付。この値は、p_to_date で指定した値以下にする必要があります。	—	はい	アーカイブログのバックアップのスタート日。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_to_date	date	ディスクにあるアーカイブ REDO ログの start_date ~next_ の日付。この値は、p_from_date で指定した値以上にする必要があります。	—	はい	アーカイブログのバックアップの終了日。

次の例では、DB インスタンスの日付範囲のアーカイブ REDO ログをバックアップします。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_date(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_from_date            => '03/01/2019 00:00:00',
        p_to_date              => '03/02/2019 00:00:00',
        p_parallel             => 4,
        p_tag                  => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

SCN 範囲からのアーカイブ REDO ログのバックアップ

システム変更番号 (SCN) 範囲を指定して Amazon RDS Oracle DB インスタンスの特定のアーカイブ REDO ログをバックアップするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_archivelog_scn` を使用します。SCN 範囲では、どのアーカイブ REDO ログをバックアップするかを指定します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、以下の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`p_from_scn`	number	ディスクにあるアーカイブ REDO ログの SCN。この値は、`p_to_scn` で指定した値以下にする必	—	はい	アーカイブログのバックアップのスタート SCN。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
		要があります。			
p_to_scn	number	ディスクにあるアーカイブ REDO ログの SCN。この値は、p_from_scn で指定した値以上にする必要があります。	—	はい	アーカイブログのバックアップの終了 SCN。

次の例では、DB インスタンスの SCN 範囲のアーカイブ REDO ログをバックアップします。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_scn(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_from_scn             => 1533835,
        p_to_scn               => 1892447,
        p_parallel             => 4,
        p_tag                  => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

シーケンス番号範囲からのアーカイブ REDO ログのバックアップ

シーケンス番号範囲を指定して Amazon RDS Oracle DB インスタンスの特定のアーカイブ REDO ログをバックアップするには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_archivelog_sequence` を使用します。シーケンス番号範囲では、どのアーカイブ REDO ログをバックアップするかを指定します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、以下の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`p_from_sequence`	number	ディスクにあるアーカイブ REDO ログのシーケンス番号。この値は、`p_to_sequence` で指定した値以下に	—	はい	アーカイブログのバックアップのスタートシーケンス番号。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
		する必要があります。			
p_to_sequence	number	ディスクにあるアーカイブ REDO ログのシーケンス番号。この値は、p_fromsequence で指定した値以上にする必要があります。	—	はい	アーカイブログのバックアップの終了シーケンス番号。

次の例では、DB インスタンスのシーケンス番号範囲のアーカイブ REDO ログをバックアップします。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_archivelog_sequence(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_from_sequence       => 11160,
        p_to_sequence         => 11160,
        p_parallel            => 4,
        p_tag                 => 'MY_LOG_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

データベースの完全バックアップの実行

Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_database_full` を使用してバックアップに含まれているデータファイルのすべてのブロックのバックアップを実行します。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

> **Note**
>
> バックアップを実行する前に、ターゲットディレクトリが存在することを確認してください。ディレクトリを作成するには、次のコマンドを実行します。`EXEC rdsadmin.rdsadmin_util.create_directory('MYDIRECTORY');`

次の例では、パラメータの値を指定して DB インスタンスの完全バックアップを実行します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_database_full(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'FULL_DB_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

バックアップが完了したことを確認し、出力ファイルを一覧表示するには、次のコマンドを実行します。

```
SELECT FILENAME, FILESIZE FROM TABLE(rdsadmin.rds_file_util.listdir('MYDIRECTORY'));
```

テナントデータベースの完全バックアップの実行

コンテナデータベース (CDB) 内のテナントデータベースに含まれるすべてのデータブロックのバックアップを実行します。Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_tenant_full` を使用します。このプロシージャは現在のデータベースバックアップにのみ適用され、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

`rdsadmin_rman_util.backup_tenant_full` プロシージャは、以下の RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0) CDB
- Oracle Database 21c (21.0.0) CDB
- Oracle Database 19c (19.0.0) CDB

次の例では、パラメータの指定値を指定して、現行のテナントデータベースの完全バックアップを実行します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tenant_full(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_tag                 => 'FULL_TENANT_DB_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

データベースの増分バックアップの実行

Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_database_incremental` を使用して DB インスタンスの増分バックアップを実行できます。

増分バックアップの詳細については、Oracle ドキュメントの「増分バックアップ」を参照してください。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`

- p_include_controlfile
- p_optimize
- p_compress
- p_rman_to_dbms_output
- p_tag

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

このプロシージャでは、次の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_level	number	0, 1	0	いいえ	フル増分バックアップを有効にするには、0 を指定します。 非累積増分バックアップを有効にするには、1 を指定します。

次の例では、パラメータの値を指定して DB インスタンスの増分バックアップを実行します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_database_incremental(
        p_owner                 => 'SYS',
        p_directory_name        => 'MYDIRECTORY',
        p_level                 => 1,
        p_parallel              => 4,
```

```
        p_section_size_mb     => 10,
        p_tag                 => 'MY_INCREMENTAL_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

テナントデータベースの増分バックアップの実行

CDB 内の現在のテナントデータベースの増分バックアップを実行できます。Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_tenant_incremental` を使用します。

増分バックアップの詳細については、Oracle Database ドキュメントの「増分バックアップ」を参照してください。

このプロシージャは現在のテナントデータベースにのみ適用され、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`
- `p_section_size_mb`
- `p_include_archive_logs`
- `p_include_controlfile`
- `p_optimize`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0) CDB
- Oracle Database 21c (21.0.0) CDB
- Oracle Database 19c (19.0.0) CDB

このプロシージャでは、次の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_level	number	0, 1	0	いいえ	フル増分バックアップを有効にするには、0 を指定します。 非累積増分バックアップを有効にするには、1 を指定します。

次の例では、パラメータの指定値を指定して、現行のテナントデータベースの増分バックアップを実行します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tenant_incremental(
        p_owner                => 'SYS',
        p_directory_name       => 'MYDIRECTORY',
        p_level                => 1,
        p_parallel             => 4,
        p_section_size_mb      => 10,
        p_tag                  => 'MY_INCREMENTAL_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

テーブルスペースのバックアップ

Amazon RDS プロシージャ rdsadmin.rdsadmin_rman_util.backup_tablespace を使用してテーブルスペースをバックアップできます。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- p_owner
- p_directory_name
- p_label
- p_parallel

- p_section_size_mb
- p_include_archive_logs
- p_include_controlfile
- p_optimize
- p_compress
- p_rman_to_dbms_output
- p_tag

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、次の追加のパラメータも使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
p_tablespace_name	varchar2	有効なテーブルスペース名。	—	はい	バックアップするテーブルスペースの名前。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

次の例では、パラメータの値を指定してテーブルスペースバックアップを実行します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_tablespace(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_tablespace_name     => 'MYTABLESPACE',
        p_parallel            => 4,
```

```
        p_section_size_mb     => 10,
        p_tag                 => 'MYTABLESPACE_BACKUP',
        p_rman_to_dbms_output => FALSE);
END;
/
```

コントロールファイルのバックアップ

Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.backup_current_controlfile` を使用して、コントロールファイルをバックアップできます。

このプロシージャでは、以下の RMAN タスクの共通パラメータを使用します。

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_compress`
- `p_rman_to_dbms_output`
- `p_tag`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

次の例では、パラメータに指定した値を使用して、コントロールファイルをバックアップします。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.backup_current_controlfile(
        p_owner               => 'SYS',
        p_directory_name      => 'MYDIRECTORY',
        p_tag                 => 'CONTROL_FILE_BACKUP',
        p_rman_to_dbms_output => FALSE);
```

```
END;
/
```

ブロックメディアリカバリの実行

Amazon RDS プロシージャ `rdsadmin.rdsadmin_rman_util.recover_datafile_block` を使用して、ブロックメディアリカバリと呼ばれる個々のデータブロックを復元できます。この重複された手順を使用して、個々のデータブロックまたはデータブロックの範囲を復元できます。

このプロシージャでは、次の RMAN タスクの共通パラメータを使用します。

- `p_rman_to_dbms_output`

詳細については、「RMAN プロシージャの共通パラメータ」を参照してください。

このプロシージャでは、以下の追加のパラメータを使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`p_datafile`	`NUMBER`	有効なデータファイル ID 番号。	—	はい	破損ブロックを含むデータファイル。次のいずれかの方法でデータファイルを指定します。 • `V$DATAFILE.FILE#` にあるするデータファイル ID 番号 • `V$DATAFILE.NAME` にあるパスを含む完全なデータファイル名
`p_block`	`NUMBER`	有効な整数。	—	はい	復元する個々のブロックの数。 次のパラメータは相互に排他的です。 • `p_block`

パラメータ名	データ型	有効な値	デフォルト	必須	説明
					• p_from_block および p_to_block
p_from_block	NUMBER	有効な整数。	—	はい	復元するブロックの範囲の最初のブロック番号。 次のパラメータは相互に排他的です。 • p_block • p_from_block および p_to_block
p_to_block	NUMBER	有効な整数。	—	はい	復元するブロックの範囲内の最後のブロック番号。 次のパラメータは相互に排他的です。 • p_block • p_from_block および p_to_block

このプロシージャは、以下の Amazon RDS for Oracle DB エンジンバージョンでサポートされています。

- Oracle Database 26ai (26.0.0)
- Oracle Database 21c (21.0.0)
- Oracle Database 19c (19.0.0)

次の例では、データファイル 5 のブロック 100 を復元します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.recover_datafile_block(
```

```
        p_datafile            => 5,
        p_block               => 100,
        p_rman_to_dbms_output => TRUE);
END;
/
```

次の例では、データファイル 5 のブロック 100 ~ 150 を復元します。

```
BEGIN
    rdsadmin.rdsadmin_rman_util.recover_datafile_block(
        p_datafile            => 5,
        p_from_block          => 100,
        p_to_block            => 150,
        p_rman_to_dbms_output => TRUE);
END;
/
```

Oracle DB インスタンスの一般的なスケジューリングタスクの実行

SYSが所有する一部のスケジューラジョブは、通常のデータベースオペレーションを妨げる可能性があります。このような場合、Oracle サポートはスケジュールを変更することを推奨しています。SYS ジョブを有効または無効にする必要がある場合は、本番環境に実装する前に、テスト環境のスケジュールされたジョブでオペレーションをテストします。Amazon RDS パッケージ`rdsadmin.rdsadmin_dbms_scheduler`を使用して、SYSが所有する Oracle スケジューラジョブのタスクを実行します。

`rdsadmin.rdsadmin_dbms_scheduler` 手順は、以下の表に示す Amazon RDS for Oracle DB エンジンバージョンでサポートされています。このパッケージを使用する場合は、表に示されている SYS ジョブを指定できます。

データベースリリース	デフォルトで有効のジョブ	デフォルトで無効のジョブ
Oracle Database 19c	BSLN_MAINTAIN_STATS_JOB CLEANUP_NON_EXIST_OBJ CLEANUP_ONLINE_IND_BUILD CLEANUP_ONLINE_PMO CLEANUP_TAB_IOT_PMO CLEANUP_TRANSIENT_PKG	FGR$AUTOPURGE_JOB FILE_WATCHER HM_CREATE_OFFLINE_DICTIONARY LOAD_OPATCH_INVENTORY ORA$PREPLUGIN_BACKUP_JOB XMLDB_NFS_CLEANUP_JOB

データベースリリース	デフォルトで有効のジョブ	デフォルトで無効のジョブ
	CLEANUP_TRANSIENT_TYPE DRA_REEVALUATE_OPEN_FAILU RES FILE_SIZE_UPD ORA$AUTOTASK_CLEAN PMO_DEFERRED_GIDX_MAINT_JO B PURGE_LOG RSE$CLEAN_RECOVERABLE_SC RIPT SM$CLEAN_AUTO_SPLIT_MERGE	
Oracle Database 21c	BSLN_MAINTAIN_STATS_JOB CLEANUP_NON_EXIST_OBJ CLEANUP_ONLINE_IND_BUILD CLEANUP_ONLINE_PMO CLEANUP_TAB_IOT_PMO CLEANUP_TRANSIENT_PKG CLEANUP_TRANSIENT_TYPE DRA_REEVALUATE_OPEN_FAILU RES FILE_SIZE_UPD ORA$AUTOTASK_CLEAN PMO_DEFERRED_GIDX_MAINT_JO B PURGE_LOG	FGR$AUTOPURGE_JOB FILE_WATCHER HM_CREATE_OFFLINE_DICTIONARY LOAD_OPATCH_INVENTORY ORA$PREPLUGIN_BACKUP_JOB ORA$_ATSK_AUTOSTS XMLDB_NFS_CLEANUP_JOB

Oracle Scheduler プロシージャの共通パラメータ

Oracle Scheduler のタスクを実行するには、Amazon RDS パッケージ `rdsadmin.rdsadmin_dbms_scheduler` を使用します。いくつかのパラメータは、パッケージ内のすべてのプロシージャに共通です。パッケージ内の共通パラメータは以下のとおりです。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
name	varchar2	「Oracle DB インスタンスの一般的なスケジューリングタスクの実行」の表に記載されている手順	—	はい	変更するジョブの名前。
attribute	varchar2	'REPEAT_INTERVAL' _NAME'	–	はい	変更する属性。 ジョブの繰り返し間隔を変更するには、'REPEAT_INTERVAL' を指定します。 ジョブのスケジュール名を変更するには、'SCHEDULE_NAME' を指定します。
value	varchar2	使用される属性に応じて、有効なスケジュール間隔またはスケ	–	はい	属性の新しい値。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
		ジュール名。			

DBMS_SCHEDULER ジョブの変更

Oracle 手順dbms_scheduler.set_attributeを使用して、Oracle スケジューラの特定のコンポーネントを変更します。詳細については、Oracle ドキュメントの「DBMS_SCHEDULER」と「SET_ATTRIBUTE プロシージャ」を参照してください。

Amazon RDS DB インスタンスを使用するときは、オブジェクト名にスキーマ名 SYS を付加します。次の例では、Monday ウィンドウオブジェクトのリソースプラン属性を設定します。

```
BEGIN
    DBMS_SCHEDULER.SET_ATTRIBUTE(
        name      => 'SYS.MONDAY_WINDOW',
        attribute => 'RESOURCE_PLAN',
        value     => 'resource_plan_1');
END;
/
```

自動タスクメンテナンスウィンドウの変更

Amazon RDS for Oracle インスタンスは、メンテナンスウィンドウのデフォルト設定で作成されます。オプティマイザ統計収集などの自動メンテナンスタスクは、これらのウィンドウで実行されます。デフォルトでは、メンテナンスウィンドウは Oracle Database リソースマネージャーをオンにします。

DBMS_SCHEDULERパッケージを使用して。ウィンドウを変更します。次のような理由から、メンテナンスウィンドウの設定を変更する必要がある場合があります。

- メンテナンスジョブを別の時間に実行するか、異なる設定で実行するか、まったく実行しないようにする。例えば、ウィンドウの継続時間を変更したり、繰り返し時間と間隔を変更したりできます。
- メンテナンス中にリソースマネージャーを有効にしたときの、パフォーマンスへの影響を回避する必要があります。例えば、デフォルトのメンテナンスプランが指定されていて、データベースがロードされている間にメンテナンスウィンドウが開いた場合、resmgr:cpu quantumのような待

機イベントを表示する必要があります。この待機イベントは、データベースリソースマネージャーに関連しています。次のオプションがあります。

- DB インスタンスのオフピーク時にメンテナンスウィンドウがアクティブであることを確認します。
- デフォルトのメンテナンスプランを無効にするには、空の文字列にresource_plan属性を設定します。
- パラメータグループのresource_manager_planパラメータをFORCE:に設定します。インスタンスでエンタープライズエディションを使用している場合、この設定により、データベースリソースマネージャープランがアクティブ化されなくなります。

メンテナンスウィンドウの設定を変更するには

1. Oracle SQL クライアントを使用して、データベースに接続します。
2. スケジューラウィンドウの現在の設定をクエリします。

   次の例では、MONDAY_WINDOWの設定をクエリします。

```
SELECT ENABLED, RESOURCE_PLAN, DURATION, REPEAT_INTERVAL
FROM   DBA_SCHEDULER_WINDOWS
WHERE  WINDOW_NAME='MONDAY_WINDOW';
```

   次の出力は、ウィンドウがデフォルト値を使用していることを示しています。

```
ENABLED         RESOURCE_PLAN                  DURATION         REPEAT_INTERVAL
--------------- ------------------------------ ----------------
 ------------------------------
TRUE            DEFAULT_MAINTENANCE_PLAN       +000 04:00:00
 freq=daily;byday=MON;byhour=22
                                                                 ;byminute=0;
 bysecond=0
```

3. DBMS_SCHEDULERパッケージを使用して、ウィンドウを変更します。

   次の例では、リソースプランを null に設定して、リソースマネージャーがメンテナンスウィンドウ中に実行されないようにします。

```
BEGIN
  -- disable the window to make changes
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);
```

```
  -- specify the empty string to use no plan
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'RESOURCE_PLAN', value=>'');

  -- re-enable the window
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
/
```

次の例では、ウィンドウの最大時間を 2 時間に設定します。

```
BEGIN
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'DURATION', value=>'0 2:00:00');
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
/
```

次の例では、繰り返し間隔を毎週月曜日の午前 10 時に設定します。

```
BEGIN
  DBMS_SCHEDULER.DISABLE(name=>'"SYS"."MONDAY_WINDOW"',force=>TRUE);
  DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'"SYS"."MONDAY_WINDOW"',
 attribute=>'REPEAT_INTERVAL',
 value=>'freq=daily;byday=MON;byhour=10;byminute=0;bysecond=0');
  DBMS_SCHEDULER.ENABLE(name=>'"SYS"."MONDAY_WINDOW"');
END;
/
```

Oracle Scheduler ジョブのタイムゾーンの設定

Oracle Scheduler のタイムゾーンを変更するには、Oracle プロシージャ `dbms_scheduler.set_scheduler_attribute` を使用できます。`dbms_scheduler` パッケージの詳細については、Oracle ドキュメントの DBMS_SCHEDULER および SET_SCHEDULER_ATTRIBUTE を参照してください。

現在のタイムゾーン設定を変更するには

1. SQL Developer などのクライアントを使用してデータベースに接続します。詳細については、「Oracle SQL Developer を使用した DB インスタンスへの接続」を参照してください。
2. デフォルトのタイムゾーンを以下のように設定し、*time_zone_name* をお客様のタイムゾーンに置き換えます。

```
BEGIN
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    attribute => 'default_timezone',
    value => 'time_zone_name'
  );
END;
/
```

以下の例では、タイムゾーンをアジア/上海に変更します。

以下に示すように、まず現在のタイムゾーンをクエリします。

```
SELECT VALUE FROM DBA_SCHEDULER_GLOBAL_ATTRIBUTE WHERE
 ATTRIBUTE_NAME='DEFAULT_TIMEZONE';
```

出力は、現在のタイムゾーンが ETC/UTC であることを示しています。

```
VALUE
-------
Etc/UTC
```

次に、タイムゾーンをアジア/上海に設定します。

```
BEGIN
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    attribute => 'default_timezone',
    value => 'Asia/Shanghai'
  );
END;
/
```

システムのタイムゾーンの変更の詳細については、「Oracle のタイムゾーン」を参照してください。

SYS が所有する Oracle Scheduler ジョブの無効化

SYS ユーザーが所有する Oracle Scheduler ジョブを無効にするには、`rdsadmin.rdsadmin_dbms_scheduler.disable` プロシージャを使用します。

このプロシージャでは、Oracle Scheduler タスクの共通パラメータ `name` を使用します。詳細については、「Oracle Scheduler プロシージャの共通パラメータ」を参照してください。

次の例では、`SYS.CLEANUP_ONLINE_IND_BUILD` Oracle Scheduler ジョブを無効にします。

```
BEGIN
   rdsadmin.rdsadmin_dbms_scheduler.disable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

SYS が所有する Oracle Scheduler ジョブを有効にする

SYS が所有する Oracle Scheduler ジョブを有効にするには、`rdsadmin.rdsadmin_dbms_scheduler.enable` プロシージャを使用します。

このプロシージャでは、Oracle Scheduler タスクの共通パラメータ `name` を使用します。詳細については、「Oracle Scheduler プロシージャの共通パラメータ」を参照してください。

次の例では、`SYS.CLEANUP_ONLINE_IND_BUILD` Oracle Scheduler ジョブを有効にします。

```
BEGIN
   rdsadmin.rdsadmin_dbms_scheduler.enable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

CALENDAR タイプのジョブに対する Oracle Scheduler の繰り返し間隔の変更

`CALENDAR` タイプ の SYS 所有の Oracle Scheduler ジョブの繰り返し間隔を変更するには、`rdsadmin.rdsadmin_dbms_scheduler.set_attribute` プロシージャを使用します。

このプロシージャでは、以下の Oracle Scheduler タスクの共通パラメータを使用します。

- `name`
- `attribute`
- `value`

詳細については、「Oracle Scheduler プロシージャの共通パラメータ」を参照してください。

次の例では、SYS.CLEANUP_ONLINE_IND_BUILD Oracle Scheduler ジョブの繰り返し間隔を変更します。

```
BEGIN
     rdsadmin.rdsadmin_dbms_scheduler.set_attribute(
          name      => 'SYS.CLEANUP_ONLINE_IND_BUILD',
          attribute => 'repeat_interval',
          value     => 'freq=daily;byday=FRI,SAT;byhour=20;byminute=0;bysecond=0');
END;
/
```

NAMED タイプのジョブに対する Oracle Scheduler の繰り返し間隔の変更

一部の Oracle Scheduler ジョブでは、間隔ではなくスケジュール名が使用されます。このようなタイプのジョブの場合、マスターユーザースキーマに新しい名前付きスケジュールを作成する必要があります。これを行うには、スタンダードの sys.dbms_scheduler.create_schedule プロシージャを使用します。また、rdsadmin.rdsadmin_dbms_scheduler.set_attribute procedure を使用して、新しい名前付きスケジュールをジョブに割り当てます。

このプロシージャでは、以下の Oracle Scheduler タスクの共通パラメータを使用します。

- name
- attribute
- value

詳細については、「Oracle Scheduler プロシージャの共通パラメータ」を参照してください。

次の例では、SYS.BSLN_MAINTAIN_STATS_JOB Oracle Scheduler ジョブの繰り返し間隔を変更します。

```
BEGIN
     DBMS_SCHEDULER.CREATE_SCHEDULE (
          schedule_name   => 'rds_master_user.new_schedule',
          start_date      => SYSTIMESTAMP,
          repeat_interval =>
 'freq=daily;byday=MON,TUE,WED,THU,FRI;byhour=0;byminute=0;bysecond=0',
          end_date        => NULL,
          comments        => 'Repeats daily forever');
```

```
END;
/

BEGIN
     rdsadmin.rdsadmin_dbms_scheduler.set_attribute (
          name      => 'SYS.BSLN_MAINTAIN_STATS_JOB',
          attribute => 'schedule_name',
          value     => 'rds_master_user.new_schedule');
END;
/
```

Oracle Scheduler ジョブ作成のオートコミットをオフにする

DBMS_SCHEDULER.CREATE_JOB が Oracle Scheduler ジョブを作成すると、すぐにジョブが作成され、変更がコミットされます。次の操作を行うには、Oracle Scheduler ジョブの作成をユーザートランザクションに組み込むことが必要な場合があります。

- ユーザートランザクションがロールバックされたら、Oracle Schedule ジョブをロールバックします。
- メインユーザートランザクションがコミットされたら、Oracle Scheduler ジョブを作成します。

rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag プロシージャを使用してこの動作をオンにできます。この手順では、パラメータは使用しません。このプロシージャは、次の RDS for Oracle リリースで使用できます。

- 21.0.0.0.ru-2022-07.rur-2022-07.r1 以上
- 19.0.0.0.ru-2022-07.rur-2022-07.r1 以上

次の例では、Oracle Scheduler のオートコミットをオフにし、Oracle Scheduler ジョブを作成してから、トランザクションをロールバックします。オートコミットがオフになっているため、データベースは Oracle Scheduler ジョブの作成もロールバックします。

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.set_no_commit_flag;
  DBMS_SCHEDULER.CREATE_JOB(job_name   => 'EMPTY_JOB',
                            job_type   => 'PLSQL_BLOCK',
                            job_action => 'begin null; end;',
                            auto_drop  => false);
  ROLLBACK;
```

```
END;
/

PL/SQL procedure successfully completed.

SELECT * FROM DBA_SCHEDULER_JOBS WHERE JOB_NAME='EMPTY_JOB';

no rows selected
```

RDS for Oracle でのリアルタイム SQL 計画管理

Oracle Database 26ai (26.0.0.0) 以降、Amazon RDS for Oracle はリアルタイム SQL 計画管理をサポートしています。この機能は、実行計画の変更を検出し、以前の計画と照らし合わせて評価し、必要に応じて SQL 計画ベースラインから最もパフォーマンスの高い計画を選択することで、SQL パフォーマンスのリグレッションを防ぎます。このプロセスは透過的であり、手動による介入は必要ありません。

RDS for Oracle でリアルタイム SQL 計画管理を有効にするには、`DBMS_SPM.CONFIGURE` を使用して自動進化タスクを有効にします。次に、`rdsadmin.rdsadmin_spm_util` パッケージを使用して SYS 所有のタスクパラメータを設定します。

トピック

- 概要
- 要件
- リアルタイム SQL 計画管理
- 設定の確認
- リアルタイム SQL 計画管理の無効化
- rdsadmin_spm_util プロシージャ
- SQL 計画ベースラインのモニタリング
- 考慮事項
- 関連リソース

概要

SQL パフォーマンスのリグレッションは、Oracle オプティマイザが以前の計画よりもパフォーマンスが低い新しい実行計画を選択した場合に発生します。リアルタイム SQL 計画管理は、これらのリグレッションの検出と防止を自動化します。

リアルタイム SQL 計画管理が有効になっている場合、データベースは以下を実行します。

1. 実行計画の変更をリアルタイムで検出します。
2. 新しい計画のパフォーマンスを以前の計画と照らし合わせて評価します。
3. SQL 計画ベースラインを自動的に作成または更新します。
4. リグレッションを防ぐために、最もパフォーマンスの高い計画を選択します。

要件

リアルタイム SQL 計画管理を使用するには、DB インスタンスが次の要件を満たしている必要があります。

- Oracle Database 26ai 以降
- Oracle Enterprise Edition

リアルタイム SQL 計画管理

リアルタイム SQL 計画管理を有効にするには、次の 2 つのステップを実行します。

ステップ 1: 自動 SPM 進化タスクを有効にする

マスターユーザーとして、または DBA ロールを持つユーザーとして、次のステートメントを実行します。

```
BEGIN
  DBMS_SPM.CONFIGURE('AUTO_SPM_EVOLVE_TASK', 'AUTO');
END;
/
```

ステップ 2: SYS 所有のタスクパラメータを設定する

SYS_AUTO_SPM_EVOLVE_TASK タスクは SYS によって所有されます。RDS for Oracle は、この目的で `rdsadmin.rdsadmin_spm_util` パッケージを提供します。

データベースが進化したプランを自動的に受け入れるように ACCEPT_PLANS を TRUE に設定します。

```
EXEC rdsadmin.rdsadmin_spm_util.set_evolve_task_accept_plans('TRUE');
```

(オプション) 代替計画ソースを AUTO に設定します。

```
EXEC rdsadmin.rdsadmin_spm_util.set_evolve_task_alternate_plan_source('AUTO');
```

設定の確認

リアルタイム SQL 計画管理が有効になっていることを確認するには、次のクエリを実行します。

```
SELECT parameter_value
FROM DBA_SQL_MANAGEMENT_CONFIG
WHERE parameter_name = 'AUTO_SPM_EVOLVE_TASK';
```

このクエリは AUTO を返します。

タスクパラメータの設定を確認するには、次のクエリを実行します。

```
SELECT parameter_name, parameter_value
FROM DBA_ADVISOR_PARAMETERS
WHERE task_name = 'SYS_AUTO_SPM_EVOLVE_TASK'
  AND parameter_name IN ('ACCEPT_PLANS', 'ALTERNATE_PLAN_SOURCE');
```

リアルタイム SQL 計画管理の無効化

リアルタイム SQL 計画管理を無効にするには、次のステートメントを実行します。

```
BEGIN
  DBMS_SPM.CONFIGURE('AUTO_SPM_EVOLVE_TASK', 'OFF');
END;
/
```

rdsadmin_spm_util プロシージャ

`rdsadmin.rdsadmin_spm_util` パッケージには、次のプロシージャが用意されています。

rdsadmin_spm_util プロシージャ

手順	パラメータ	デフォルト	説明
set_evolve_task_accept_plans	p_value	'TRUE'	SYS_AUTO_SPM_EVOLVE_TASK の ACCEPT_PLANS パラメータを設定します。有効な値は、'TRUE' および 'FALSE' です。
set_evolve_task_alternate_plan_source	p_value	'AUTO'	SYS_AUTO_SPM_EVOLVE_TASK の ALTERNATE_PLAN_SOURCE パラメータを設定します。有効な値は、'AUTO'、'CURSOR_CACHE' 、'AUTOMATIC_WORKLOAD_REPOSITORY' 、および 'SQL_TUNING_SET' です。

SQL 計画ベースラインのモニタリング

リアルタイム SQL 計画管理を有効にすると、データベースが自動的に管理する計画ベースラインをモニタリングできます。

```
SELECT SQL_HANDLE, PLAN_NAME, ENABLED, ACCEPTED, AUTOPURGE
FROM DBA_SQL_PLAN_BASELINES
ORDER BY LAST_MODIFIED DESC;
```

考慮事項

リアルタイム SQL 計画管理を使用する場合は、次の点を考慮してください。

- DBA ロールを持つユーザーは、DBMS_SPM.CONFIGURE を直接呼び出すことができます。このステップでは、ラッパーは必要ありません。
- マスターユーザーは SYS 所有のアドバイザータスクに直接アクセスできないため、rdsadmin.rdsadmin_spm_util の手順は SYS 所有の SYS_AUTO_SPM_EVOLVE_TASK タスクでタスクパラメータを設定するために必要です。
- リアルタイム SQL 計画管理は透過的に動作します。アプリケーションコードを変更する必要はありません。
- SQL 計画ベースラインは、SYSAUX テーブルスペースのスペースを消費します。この機能を有効にした後、SYSAUX の使用状況をモニタリングしてください。

- DBMS_SPM.CONFIGURE に渡されるパラメータ値 ('AUTO' や 'OFF' など) では、大文字と小文字が区別されます。大文字の値を使用します。

関連リソース

- DBMS_SPM (Oracle Database ドキュメント)
- SQL 計画ベースラインの管理 (Oracle Database ドキュメント)

RDS for Oracle でのテーブルスペースの縮小

Oracle Database 26ai (26.0.0.0) 以降、Amazon RDS for Oracle は rdsadmin.rdsadmin_util.shrink_tablespace プロシージャによるテーブルスペースの縮小をサポートしています。データファイルを縮小して、テーブルスペース内の未使用のスペースを再利用するには、この手順を使用します。縮小すると、データ移行やテーブルスペースの再作成を行わずにストレージの消費量を削減できます。

トピック

- 概要
- 要件
- 構文
- パラメータ
- 例
- 考慮事項
- 関連リソース

概要

時間の経過とともに、Oracle Database のテーブルスペースは、削除または再編成されたデータから未使用のスペースを蓄積する可能性があります。rdsadmin.rdsadmin_util.shrink_tablespace プロシージャは、Oracle DBMS_SPACE.SHRINK_TABLESPACE の機能をラップし、必要な内部権限を処理します。

テーブルスペースの縮小では、次のモードがサポートされています。

- 分析モード (DBMS_SPACE.TS_MODE_ANALYZE) – テーブルスペースを評価し、変更を加えずに再利用できるスペースを報告します。

- 縮小モード (DBMS_SPACE.TS_MODE_SHRINK) – オブジェクトを移動し、テーブルスペース内のデータファイルのサイズを変更して、未使用のスペースを再利用します。これはデフォルトモードです。

要件

テーブルスペースの縮小を使用するには、DB インスタンスが次の要件を満たしている必要があります。

- Oracle Database 26ai 以降
- Oracle Enterprise Edition

構文

```
EXEC rdsadmin.rdsadmin_util.shrink_tablespace(
  p_tablespace_name => 'tablespace_name',
  p_shrink_mode     => mode,
  p_target_size     => target_bytes);
```

パラメータ

shrink_tablespace パラメータ

パラメータ名	データ型	デフォルト	必須	説明
p_tablespace_name	VARCHAR2	—	はい	縮小するテーブルスペースの名前。
p_shrink_mode	NUMBER	2 (TS_MODE_SHRINK)	いいえ	縮小モード。分析のみを行う場合は DBMS_SPACE.TS_MODE_ANALYZE (1)、縮小を実行する場合は DBMS_SPACE.TS_MODE_SHRINK (2) を使用します。
p_target_size	NUMBER	0	いいえ	ターゲットサイズ (バイト単位)。0 (デフォルト) を使用して、可能な最小サイズ

パラメータ名	データ型	デフォルト	必須	説明
				に縮小します。特定のターゲットに縮小する正の値を指定します。

例

再利用可能なスペースのテーブルスペースの分析

変更を加えずにテーブルスペースを分析するには、次のコマンドを実行します。

```
EXEC rdsadmin.rdsadmin_util.shrink_tablespace('USERS', DBMS_SPACE.TS_MODE_ANALYZE);
```

出力例を次に示します。

```
-------------------ANALYZE RESULT-------------------
Total Movable Objects: 0
Total Movable Size(GB): 0
Original Datafile Size(GB): .1
Suggested Target Size(GB): .06
Process Time: +00 00:00:00.021513
```

テーブルスペースを最小サイズに縮小する

テーブルスペースを縮小し、未使用のスペースをすべて再利用するには、次のコマンドを実行します。

```
EXEC rdsadmin.rdsadmin_util.shrink_tablespace('USERS');
```

出力例を次に示します。

```
-------------------SHRINK RESULT-------------------
Total Moved Objects: 0
Total Moved Size(GB): 0
Original Datafile Size(GB): .1
New Datafile Size(GB): .05
Process Time: +00 00:00:00.281151
```

特定のターゲットサイズへのテーブルスペースの縮小

テーブルスペースをバイト単位でターゲットサイズに縮小するには、p_target_size パラメータを指定します。ターゲットが達成可能な最小サイズより小さい場合、Oracle はターゲットを自動的に上方に調整します。

```
EXEC rdsadmin.rdsadmin_util.shrink_tablespace('USERS', DBMS_SPACE.TS_MODE_SHRINK,
 52428800);
```

考慮事項

テーブルスペースを縮小する場合は、次の点を考慮してください。

- テーブルスペースの縮小はオンラインオペレーションです。テーブルスペース内のオブジェクトには、縮小している間も引き続きアクセスできます。
- 縮小オペレーションでは、テーブルやインデックスなどのオブジェクトをテーブルスペース内に移動して、データファイルのサイズを変更する前に空きスペースを統合する場合があります。
- インデックスは縮小オペレーション後も有効であり、データの整合性は保持されます。
- p_target_size の値がオブジェクトが現在占有しているスペースよりも小さい場合、Oracle は達成可能な最小サイズにターゲットを調整します。
- p_target_size の値は、現在のテーブルスペースサイズを超えることはできません。現在のサイズ以下のターゲットを指定します。
- テーブルスペース内のオブジェクトに対するアクティブなトランザクションにより、オペレーションが使用可能なすべてのスペースを再利用できない可能性があります。最良の結果を得るには、アクティビティが少ない期間に縮小を実行します。
- rdsadmin_util ラッパーなしで DBMS_SPACE.SHRINK_TABLESPACE を直接呼び出すことはサポートされていません。必ず rdsadmin.rdsadmin_util.shrink_tablespace ラッパーを使用してください。

関連リソース

- DBMS_SPACE.SHRINK_TABLESPACE (Oracle Database ドキュメント)
- 詳細については、「Oracle DB インスタンスの一般的なデータベースタスクの実行」を参照してください。

RDS for Oracle DB インスタンスの問題の診断

Oracle Databaseは、データベースの問題を調査するために使用できる障害診断能力インフラストラクチャを備えています。Oracle の用語では、問題は、コードのバグやデータの破損などの重大なエラーです。インシデントとは、問題の発生です。同じエラーが 3 回発生した場合、このインフラストラクチャには、この問題の 3 つのインシデントが示されます。詳細は、Oracle Databaseドキュメントの「問題の診断と解決」を参照してください。

自動診断リポジトリコマンドインタープリタ (ADRCI) ユーティリティは、診断データの管理に使用する Oracle コマンドラインツールです。例えば、このツールを使用して問題を調査し、診断データをパッケージ化することができます。インシデントパッケージには、特定の問題を参照する 1 つのインシデントまたはすべてのインシデントの診断データが含まれます。.zipファイルとして実装されているインシデントパッケージを Oracle サポートにアップロードできます。

マネージドサービスエクスペリエンスを提供するために、Amazon RDS は ADRCI へのシェルアクセスを提供していません。RDS for Oracle DB インスタンスの診断タスクを実行するには、Amazon RDS パッケージ `rdsadmin.rdsadmin_adrci_util` を使用します。

`rdsadmin_adrci_util` で関数を使用すると、問題やインシデントをリストしてパッケージ化でき、トレースファイルも表示できます。すべての関数はタスク ID を返します。この ID は、`dbtask-`*`task_id`*`.log` のように ADRCI 出力を含むログファイルの名前の一部を形成します。ログファイルは BDUMP ディレクトリにあります。データベースログファイルのダウンロード で説明した手順に従って、ログファイルをダウンロードできます。

診断手順の共通パラメータ

診断タスクを実行するには、 Amazon RDS パッケージ `rdsadmin.rdsadmin_adrci_util` 内の関数を使用します。パッケージ内の共通パラメータは以下のとおりです。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
`incident_id`	number	有効なインシデント ID または null	Null	いいえ	値が null の場合、関数はすべてのインシデントを表示します。値が null ではなく、有効なインシデント ID を指定してい

パラメータ名	データ型	有効な値	デフォルト	必須	説明
					る場合は、関数は指定されたインシデントを表示します。
problem_id	number	有効な問題 ID または null	Null	いいえ	値が null の場合、関数はすべての問題を表示します。値が null ではなく、有効な問題 ID を表す場合、関数は指定された問題を表示します。
last	number	0 より大きい有効な整数、または null	Null	いいえ	値が null の場合、関数は最大 50 個の項目を表示します。値が null でない場合、関数は指定された数を表示します。

インシデントのリスト化

Oracle の診断インシデントを一覧表示するには、Amazon RDS 関数 `rdsadmin.rdsadmin_adrci_util.list_adrci_incidents` を使用します。インシデントは、ベーシックモードまたは詳細モードのいずれかでリストできます。デフォルトでは、この関数は最新の 50 件のインシデントをリストします。

この関数は、以下の共通パラメータを使用します。

- `incident_id`
- `problem_id`
- `last`

`incident_id` と `problem_id` を指定すると、`incident_id` は `problem_id` より優先されます。詳細については、「診断手順の共通パラメータ」を参照してください。

この関数は、次の追加パラメータを使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
detail	boolean	TRUE、、または FALSE	FALSE	いいえ	TRUE の場合、関数はインシデントを詳細モードでリストします。FALSE の場合、この関数はインシデントをベーシックモードでリストします。

すべてのインシデントを一覧表示するには、引数を指定せずに rdsadmin.rdsadmin_adrci_util.list_adrci_incidents 関数をクエリします。クエリはタスク ID を返します。

```
SQL> SELECT rdsadmin.rdsadmin_adrci_util.list_adrci_incidents AS task_id FROM DUAL;

TASK_ID
------------------
1590786706158-3126
```

または、引数を指定せずに rdsadmin.rdsadmin_adrci_util.list_adrci_incidents 関数を呼び出し、出力を SQL クライアント変数に保存します。変数は、他のステートメントで使用できます。

```
SQL> VAR task_id VARCHAR2(80);
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_incidents;

PL/SQL procedure successfully completed.
```

ログファイルを読み取るには、Amazon RDS プロシージャ rdsadmin.rds_file_util.read_text_file を呼び出します。ファイル名の一部としてタスク ID を指定します。次の出力は、53523、53522、53521 の 3 つのインシデントを示しています。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:11:46.193 UTC [INFO ] Listing ADRCI incidents.
2020-05-29 21:11:46.256 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
INCIDENT_ID PROBLEM_KEY                                                 CREATE_TIME
----------- ------------------------------------------------------------
 ----------------------------------------
53523       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003 2020-05-29
 20:15:20.928000 +00:00
53522       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 2020-05-29
 20:15:15.247000 +00:00
53521       ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_001 2020-05-29
 20:15:06.047000 +00:00
3 rows fetched


2020-05-29 21:11:46.256 UTC [INFO ] The ADRCI incidents were successfully listed.
2020-05-29 21:11:46.256 UTC [INFO ] The task finished successfully.

14 rows selected.
```

特定のインシデントをリストするには、incident_id パラメータを使用してその ID を指定します。次の例では、インシデント 53523 のみのログファイルをクエリします。

```
SQL> EXEC :task_id :=
 rdsadmin.rdsadmin_adrci_util.list_adrci_incidents(incident_id=>53523);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:15:25.358 UTC [INFO ] Listing ADRCI incidents.
2020-05-29 21:15:25.426 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
```

```
INCIDENT_ID          PROBLEM_KEY
 CREATE_TIME
-------------------- ------------------------------------------------------------
 ---------------------------------
53523                ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003
 2020-05-29 20:15:20.928000 +00:00
1 rows fetched


2020-05-29 21:15:25.427 UTC [INFO ] The ADRCI incidents were successfully listed.
2020-05-29 21:15:25.427 UTC [INFO ] The task finished successfully.

12 rows selected.
```

問題のリスト化

Oracle の診断問題を一覧表示するには、Amazon RDS 関数 `rdsadmin.rdsadmin_adrci_util.list_adrci_problems` を使用します。

デフォルトでは、この関数は最新の 50 個の問題をリストします。

この関数は、共通パラメータ `problem_id` と `last` を使用します。詳細については、「診断手順の共通パラメータ」を参照してください。

すべての問題のタスク ID を取得するには、引数を指定せずに `rdsadmin.rdsadmin_adrci_util.list_adrci_problems` 関数を呼び出し、出力を SQL クライアント可変に格納します。

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_problems;

PL/SQL procedure successfully completed.
```

ログファイルを読み込むには、ファイル名の一部としてタスク ID を指定し、`rdsadmin.rds_file_util.read_text_file` 関数を呼び出します。次の出力では、ログファイルには 1、2、3 の 3 つの問題が表示されます。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:18:50.764 UTC [INFO ] Listing ADRCI problems.
```

```
2020-05-29 21:18:50.829 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
PROBLEM_ID   PROBLEM_KEY                                                 LAST_INCIDENT
       LASTINC_TIME
---------- ------------------------------------------------------------ -------------
 -------------------------------
2          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_003 53523
 2020-05-29 20:15:20.928000 +00:00
3          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 53522
 2020-05-29 20:15:15.247000 +00:00
1          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_001 53521
 2020-05-29 20:15:06.047000 +00:00
3 rows fetched


2020-05-29 21:18:50.829 UTC [INFO ] The ADRCI problems were successfully listed.
2020-05-29 21:18:50.829 UTC [INFO ] The task finished successfully.

14 rows selected.
```

次の例では、問題 3 のみをリストします。

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_problems(problem_id=>3);

PL/SQL procedure successfully completed.
```

問題 3 のログファイルを読み込むには、rdsadmin.rds_file_util.read_text_file を呼び出します。ファイル名の一部としてタスク ID を指定します。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:19:42.533 UTC [INFO ] Listing ADRCI problems.
2020-05-29 21:19:42.599 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/ORCL:
*********************************************************************
PROBLEM_ID PROBLEM_KEY                                                 LAST_INCIDENT
 LASTINC_TIME
---------- ------------------------------------------------------------ -------------
 -------------------------------
```

```
3          ORA 700 [EVENT_CREATED_INCIDENT] [942] [SIMULATED_ERROR_002 53522
 2020-05-29 20:15:15.247000 +00:00
1 rows fetched


2020-05-29 21:19:42.599 UTC [INFO ] The ADRCI problems were successfully listed.
2020-05-29 21:19:42.599 UTC [INFO ] The task finished successfully.

12 rows selected.
```

インシデントパッケージの作成

Amazon RDS 関数 rdsadmin.rdsadmin_adrci_util.create_adrci_package を使用してインシデントパッケージを作成できます。出力は、Oracle サポートに提供できる .zip ファイルです。

この関数は、以下の共通パラメータを使用します。

- problem_id
- incident_id

上記のパラメータのいずれかを必ず指定してください。両方のパラメータを指定した場合、incident_id は problem_id をオーバーライドします。詳細については、「診断手順の共通パラメータ」を参照してください。

特定のインシデントのパッケージを作成するには、rdsadmin.rdsadmin_adrci_util.create_adrci_package パラメータを指定して Amazon RDS 関数 incident_id を呼び出します。次の例では、インシデント 53523 のパッケージを作成します。

```
SQL> EXEC :task_id :=
 rdsadmin.rdsadmin_adrci_util.create_adrci_package(incident_id=>53523);

PL/SQL procedure successfully completed.
```

ログファイルを読み込むには、rdsadmin.rds_file_util.read_text_file を呼び出します。ファイル名の一部としてタスク ID を指定することができます。出力は、インシデントパッケージ ORA700EVE_20200529212043_COM_1.zip を生成したことを示しています。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));
```

```
TEXT
--------------------------------------------------------------------------------
2020-05-29 21:20:43.031 UTC [INFO ] The ADRCI package is being created.
2020-05-29 21:20:47.641 UTC [INFO ] Generated package 1 in file /rdsdbdata/log/trace/
ORA700EVE_20200529212043_COM_1.zip, mode complete
2020-05-29 21:20:47.642 UTC [INFO ] The ADRCI package was successfully created.
2020-05-29 21:20:47.642 UTC [INFO ] The task finished successfully.
```

特定の問題の診断データをパッケージ化するには、problem_id パラメータを使用してその ID を指定します。次の例では、問題 3 のデータのみをパッケージ化します。

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.create_adrci_package(problem_id=>3);

PL/SQL procedure successfully completed.
```

タスク出力を読み取るには、ファイル名の一部としてタスク ID を指定して rdsadmin.rds_file_util.read_text_file を呼び出します。出力は、インシデントパッケージ ORA700EVE_20200529212111_COM_1.zip を生成したことを示しています。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:21:11.050 UTC [INFO ] The ADRCI package is being created.
2020-05-29 21:21:15.646 UTC [INFO ] Generated package 2 in file /rdsdbdata/log/trace/
ORA700EVE_20200529212111_COM_1.zip, mode complete
2020-05-29 21:21:15.646 UTC [INFO ] The ADRCI package was successfully created.
2020-05-29 21:21:15.646 UTC [INFO ] The task finished successfully.
```

ログファイルをダウンロードすることもできます。詳細については、「データベースログファイルのダウンロード」を参照してください。

トレースファイルの表示

Amazon RDS 関数 rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile を使用すると、trace ディレクトリの下にあるトレースファイルと、現在の ADR ホームにあるすべてのインシデントディレクトリを一覧表示できます。トレースファイルとインシデントトレースファイルの内容を表示することもできます。

この関数は、次のパラメータを使用します。

パラメータ名	データ型	有効な値	デフォルト	必須	説明
filename	varchar2	有効なトレースファイル名	Null	いいえ	値が null の場合、関数はすべてのトレースファイルを表示します。null でない場合、関数は指定されたファイルを表示します。

トレースファイルを表示するには、Amazon RDS 関数 `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile` を呼び出します。

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile;

PL/SQL procedure successfully completed.
```

トレースファイル名をリストするには、ファイル名の一部としてタスク ID を指定して、Amazon RDS プロシージャ `rdsadmin.rds_file_util.read_text_file` を呼び出します。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log')) WHERE TEXT LIKE '%/alert_%';

TEXT
--------------------------------------------------------------------
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-28
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-27
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-26
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-25
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-24
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-23
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-22
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log.2020-05-21
     diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log

9 rows selected.
```

次の例では、alert_ORCL.log の出力を生成します。

```
SQL> EXEC :task_id := rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile('diag/rdbms/
orcl_a/ORCL/trace/alert_ORCL.log');

PL/SQL procedure successfully completed.
```

ログファイルを読み込むには、rdsadmin.rds_file_util.read_text_file を呼び出します。ファイル名の一部としてタスク ID を指定します。出力には、alert_ORCL.log の初期の 10 行が表示されます。

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||:task_id||'.log')) WHERE ROWNUM <= 10;

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:24:02.083 UTC [INFO ] The trace files are being displayed.
2020-05-29 21:24:02.128 UTC [INFO ] Thu May 28 23:59:10 2020
Thread 1 advanced to log sequence 2048 (LGWR switch)
  Current log# 3 seq# 2048 mem# 0: /rdsdbdata/db/ORCL_A/onlinelog/o1_mf_3_hbl2p8xs_.log
Thu May 28 23:59:10 2020
Archived Log entry 2037 added for thread 1 sequence 2047 ID 0x5d62ce43 dest 1:
Fri May 29 00:04:10 2020
Thread 1 advanced to log sequence 2049 (LGWR switch)
  Current log# 4 seq# 2049 mem# 0: /rdsdbdata/db/ORCL_A/onlinelog/o1_mf_4_hbl2qgmh_.log
Fri May 29 00:04:10 2020

10 rows selected.
```

ログファイルをダウンロードすることもできます。詳細については、「データベースログファイルのダウンロード」を参照してください。

Oracle DB インスタンスのその他のタスクの実行

次に、Oracle を実行している Amazon RDS DB インスタンスでその他の DBA タスクを実行する方法を示します。マネージド型サービスの操作性を実現するために、Amazon RDS では DB インスタンスへのシェルアクセスは提供していません。また、上位の権限を必要とする特定のシステムプロシージャやシステムテーブルへのアクセスが制限されます。

トピック

- 主要データストレージ領域でのディレクトリの作成と削除

- DB インスタンスディレクトリ内のファイルのリスト化
- DB インスタンスディレクトリ内のファイルの読み取り
- Opatch ファイルへのアクセス
- アドバイザータスクの管理
- テーブルスペースの転送

主要データストレージ領域でのディレクトリの作成と削除

ディレクトリを作成するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.create_directory` を使用します。最大 10,000 個のディレクトリを作成でき、すべてメインデータストレージ領域に配置されます。ディレクトリを作成するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.drop_directory` を使用します。

`create_directory` および `drop_directory` プロシージャには、以下の必須パラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_directory_name	VARCHAR2	—	はい	ディレクトリの名前。

次の例では、新しいディレクトリ PRODUCT_DESCRIPTIONS を作成します。

```
EXEC rdsadmin.rdsadmin_util.create_directory(p_directory_name =>
 'product_descriptions');
```

データディクショナリは、ディレクトリ名を大文字で保存します。DBA_DIRECTORIES を照会することで、ディレクトリを一覧表示できます。システムによって実際のホストパス名が自動的に選択されます。以下の例では、PRODUCT_DESCRIPTIONS という名前のディレクトリのディレクトリパスを取得しています。

```
SELECT DIRECTORY_PATH
  FROM DBA_DIRECTORIES
 WHERE DIRECTORY_NAME='PRODUCT_DESCRIPTIONS';

DIRECTORY_PATH
----------------------------------------
/rdsdbdata/userdirs/01
```

DB インスタンスのマスターユーザー名には、新しいディレクトリでの読み取りおよび書き込み権限があり、他のユーザーにアクセスを許可できます。EXECUTE 権限は、DB インスタンス上のディレクトリでは使用できません。ディレクトリはメインデータストレージ領域に作成され、領域および I/O 帯域幅を消費します。

以下の例では、PRODUCT_DESCRIPTIONS という名前のディレクトリを削除します。

```
EXEC rdsadmin.rdsadmin_util.drop_directory(p_directory_name => 'product_descriptions');
```

> **Note**
>
> Oracle SQL コマンド DROP DIRECTORY を使用してディレクトリを削除することもできます。

ディレクトリを削除してもその内容は削除されません。rdsadmin.rdsadmin_util.create_directory プロシージャはパス名を再利用できるため、削除されたディレクトリのファイルが新たに作成されたディレクトリに存在する可能性があります。ディレクトリを削除する前に、UTL_FILE.FREMOVE を使用してディレクトリからファイルを削除することをお勧めします。詳細については、Oracle ドキュメントの「FREMOVE プロシージャ」を参照してください。

DB インスタンスディレクトリ内のファイルのリスト化

ディレクトリ内のファイルを一覧表示するには、Amazon RDS プロシージャ rdsadmin.rds_file_util.listdir を使用します。このプロシージャは Oracle レプリカではサポートされていません。listdir プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_directory	varchar2	—	はい	一覧表示するディレクトリの名前。

次の例では、ディレクトリ PRODUCT_DESCRIPTIONS に対する読み取り/書き込み権限をユーザー rdsadmin に付与し、このディレクトリ内のファイルを一覧表示します。

```
GRANT READ,WRITE ON DIRECTORY PRODUCT_DESCRIPTIONS TO rdsadmin;
```

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory =>
 'PRODUCT_DESCRIPTIONS'));
```

DB インスタンスディレクトリ内のファイルの読み取り

テキストファイルを読み取るには、Amazon RDS プロシージャ `rdsadmin.rds_file_util.read_text_file` を使用します。`read_text_file` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
`p_directory`	varchar2	—	はい	ファイルを含むディレクトリの名前。
`p_filename`	varchar2	—	はい	読み取るファイルの名前。

次の例では `rice.txt` ディレクトリの `PRODUCT_DESCRIPTIONS` ファイルを読み取ります。

```
declare
  fh sys.utl_file.file_type;
begin
  fh := utl_file.fopen(location=>'PRODUCT_DESCRIPTIONS', filename=>'rice.txt',
 open_mode=>'w');
  utl_file.put(file=>fh, buffer=>'AnyCompany brown rice, 15 lbs');
  utl_file.fclose(file=>fh);
end;
/
```

次の例では、ディレクトリ `rice.txt` からファイル `PRODUCT_DESCRIPTIONS` を読み取ります。

```
SELECT * FROM TABLE
    (rdsadmin.rds_file_util.read_text_file(
        p_directory => 'PRODUCT_DESCRIPTIONS',
        p_filename  => 'rice.txt'));
```

Opatch ファイルへのアクセス

Opatch は、Oracle ソフトウェアへのパッチの適用とロールバックを可能にする Oracle ユーティリティです。データベースに適用されているパッチを判別するための Oracle のメカニズム

は、opatch lsinventory コマンドです。Bring Your Own License (BYOL) の顧客のサービスリクエストを開くために、Oracle サポートは lsinventory ファイルをリクエストします。Opatch によって生成される lsinventory_detail ファイルをリクエストする場合もあります。

マネージドサービスエクスペリエンスを提供するために、Amazon RDS には Opatch へのシェルアクセスは用意されていません。代わりに、BDUMP ディレクトリのlsinventory-*dbv*.txtには、現在のエンジンのバージョンに関連するパッチ情報が含まれています。マイナーアップグレードまたはメジャーアップグレードを実行すると、Amazon RDS はパッチを適用してから 1 時間以内にlsinventory-*dbv*.txtを更新します。適用されたパッチを確認するには、lsinventory-*dbv*.txtを読み取ります。このアクションは、opatch lsinventoryコマンドの実行に似ています。

> **Note**
>
> このセクションの例では、BDUMP ディレクトリ名を BDUMP としています。リードレプリカでは、BDUMP ディレクトリ名が異なります。BDUMP 名を取得するためにリードレプリカに対してクエリ V$DATABASE.DB_UNIQUE_NAME を実行する方法については、「ファイルのリスト化」を参照してください。

インベントリファイルは Amazon RDS 命名規則 lsinventory-*dbv*.txt および lsinventory_detail-*dbv*.txt を使用します。ここで、*dbv* は DB バージョンの完全な名前です。lsinventory-*dbv*.txt ファイルはすべての DB バージョンで使用できます。対応する lsinventory_detail-*dbv*.txt は、19.0.0.0、ru-2020-01.rur-2020-01.r1 以降で利用できます。

例えば、DB のバージョンが 19.0.0.0.ru-2021-07.rur-2021-07.r1 の場合、インベントリファイルは以下の名前になります。

```
lsinventory-19.0.0.0.ru-2021-07.rur-2021-07.r1.txt
lsinventory_detail-19.0.0.0.ru-2021-07.rur-2021-07.r1.txt
```

DB エンジンの現在のバージョンと一致するファイルをダウンロードしてください。

コンソール

コンソールを使用してインベントリファイルをダウンロードするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。

3. 表示するログファイルのある DB インスタンスの名前を選択します。
4. [ログとイベント] タブを選択します。
5. [ログ] セクションまで下にスクロールします。
6. [ログ] セクションで、lsinventory を検索します。
7. アクセスするファイルを選択し、[ダウンロード] を選択します。

SQL

SQL クライアントで lsinventory-*dbv*.txt を読み取るには、SELECT ステートメントを使用できます。この手法では、rdsadmin 関数として rdsadmin.rds_file_util.read_text_file または rdsadmin.tracefile_listing のいずれかを使用します。

以下のサンプルクエリでは、*dbv* を Oracle DB のバージョンに置き換えます。例えば、DB のバージョンは 19.0.0.0.ru-2020-04.rur-2020-04.r1 です。

```
SELECT text
FROM   TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP', 'lsinventory-dbv.txt'));
```

PL/SQL

SQL クライアントで lsinventory-*dbv*.txt を読み取るには、PL/SQL プログラムを作成します。このプログラムは、utl_file を使用してファイルを読み取り、dbms_output を使用して出力します。これらは、Oracle が提供するパッケージです。

以下のサンプルプログラムでは、*dbv* を Oracle DB のバージョンに置き換えます。例えば、DB のバージョンは 19.0.0.0.ru-2020-04.rur-2020-04.r1 です。

```
SET SERVEROUTPUT ON
DECLARE
  v_file              SYS.UTL_FILE.FILE_TYPE;
  v_line              VARCHAR2(1000);
  v_oracle_home_type  VARCHAR2(1000);
  c_directory         VARCHAR2(30) := 'BDUMP';
  c_output_file       VARCHAR2(30) := 'lsinventory-dbv.txt';
BEGIN
  v_file := SYS.UTL_FILE.FOPEN(c_directory, c_output_file, 'r');
  LOOP
    BEGIN
```

```
      SYS.UTL_FILE.GET_LINE(v_file, v_line,1000);
      DBMS_OUTPUT.PUT_LINE(v_line);
    EXCEPTION
      WHEN no_data_found THEN
        EXIT;
    END;
  END LOOP;
END;
/
```

または、rdsadmin.tracefile_listing をクエリし、出力をファイルにスプールします。以下の例では、出力を /tmp/tracefile.txt にスプールします。

```
SPOOL /tmp/tracefile.txt
SELECT *
FROM   rdsadmin.tracefile_listing
WHERE  FILENAME LIKE 'lsinventory%';
SPOOL OFF;
```

アドバイザータスクの管理

Oracle Database には、多数のアドバイザーが含まれています。各アドバイザーは、自動タスクと手動タスクをサポートします。rdsadmin.rdsadmin_util パッケージ内のプロシージャを使用して、いくつかのアドバイザータスクを管理できます。

アドバイザータスク手順は、次のエンジンバージョンで使用できます。

- Oracle Database 26ai (26.0.0.0)
- Oracle Database 21c (21.0.0.0)
- Version 19.0.0.0.ru-2021-01.rur-2021-01.r1 以上の Oracle Database 19c バージョン

  詳細については、「Amazon RDS for Oracle リリースノート」の「バージョン 19.0.0.0.ru-2021-01.rur-2021-01.r1」を参照してください。

トピック

- アドバイザータスクのパラメータの設定
- AUTO_STATS_ADVISOR_TASK を無効にする
- AUTO_STATS_ADVISOR_TASK を再度有効にする

アドバイザータスクのパラメータの設定

一部のアドバイザータスクのパラメータを設定するには、Amazon RDS 手順 `rdsadmin.rdsadmin_util.advisor_task_set_parameter` を使用します。`advisor_task_set_parameter` プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_task_name	varchar2	—	はい	パラメータを変更するアドバイザータスクの名前。以下の値が有効です。 • AUTO_STATS_ADVISOR_TASK • INDIVIDUAL_STATS_ADVISOR_TASK • SYS_AUTO_SPM_EVOLVE_TASK • SYS_AUTO_SQL_TUNING_TASK

p_parameter	varchar2	—	はい	タスクパラメータの名前。アドバイザータスクの有効なパラメータを検索するには、次のクエリを実行します。*p_task_name* を p_task_name の有効な値に置き換えます:
				`COL PARAMETER_NAME FORMAT a30`
				`COL PARAMETER_VALUE FORMAT a30`
				`SELECT PARAMETER_NAME, PARAMETER _VALUE`
				`FROM DBA_ADVISOR_PARAMETERS`
				`WHERE TASK_NAME='  p_task_name '`
				`AND PARAMETER_VALUE != 'UNUSED'`
				`ORDER BY PARAMETER_NAME;`

p_value	varchar2	—	はい	タスクパラメータの値。タスクパラメータの有効な値を検索するには、次のクエリを実行します。*p_task_name* を p_task_name の有効な値に置き換えます:
				`COL PARAMETER_NAME FORMAT a30`
				`COL PARAMETER_VALUE FORMAT a30`

パラメータ名	データ型	デフォルト	必須	説明
				SELECT PARAMETER_NAME, PARAMETER_VALUE FROM DBA_ADVISOR_PARAMETERS WHERE TASK_NAME=' *p_task_name* ' AND PARAMETER_VALUE != 'UNUSED' ORDER BY PARAMETER_NAME;

次の PL/SQL プログラムは ACCEPT_PLANS を FALSE の SYS_AUTO_SPM_EVOLVE_TASK に設定します。SQL Plan Management の自動化タスクでは、プランが検証され、その結果のレポートが生成されますが、プランを自動的には進化させません。レポートを使用して、新しい SQL 計画ベースラインを識別し、手動で受け入れることができます。

```
BEGIN
  rdsadmin.rdsadmin_util.advisor_task_set_parameter(
    p_task_name => 'SYS_AUTO_SPM_EVOLVE_TASK',
    p_parameter => 'ACCEPT_PLANS',
    p_value     => 'FALSE');
END;
```

次の PL/SQL プログラムは EXECUTION_DAYS_TO_EXPIRE を 10 の AUTO_STATS_ADVISOR_TASK に設定します。事前定義されたタスク AUTO_STATS_ADVISOR_TASK は、メンテナンスウィンドウで 1 日 1 回自動的に実行されます。この例では、タスク実行の保持期間を 10 日に設定します。

```
BEGIN
  rdsadmin.rdsadmin_util.advisor_task_set_parameter(
    p_task_name => 'AUTO_STATS_ADVISOR_TASK',
    p_parameter => 'EXECUTION_DAYS_TO_EXPIRE',
    p_value     => '10');
END;
```

AUTO_STATS_ADVISOR_TASK を無効にする

AUTO_STATS_ADVISOR_TASK を無効にするには、Amazon RDS 手順 rdsadmin.rdsadmin_util.advisor_task_drop を使用します。advisor_task_drop 手順は、次のパラメータを受け付けます。

パラメータ名	データ型	デフォルト	必須	説明
p_task_name	varchar2	—	はい	無効にするアドバイザータスクの名前。唯一の有効な値は AUTO_STATS_ADVISOR_TASK です。

次のコマンドは AUTO_STATS_ADVISOR_TASK をドロップします。

```
EXEC rdsadmin.rdsadmin_util.advisor_task_drop('AUTO_STATS_ADVISOR_TASK')
```

rdsadmin.rdsadmin_util.dbms_stats_init を使用して AUTO_STATS_ADVISOR_TASK を再度有効にすることができます。

AUTO_STATS_ADVISOR_TASK を再度有効にする

AUTO_STATS_ADVISOR_TASK を再度有効にするには、Amazon RDS 手順 rdsadmin.rdsadmin_util.dbms_stats_init を使用します。dbms_stats_init 手順では、パラメータは使用しません。

次のコマンドは AUTO_STATS_ADVISOR_TASK を再度有効にします。

```
EXEC rdsadmin.rdsadmin_util.dbms_stats_init()
```

テーブルスペースの転送

Amazon RDS パッケージ rdsadmin.rdsadmin_transport_util を使用して、オンプレミスの Oracle データベースから RDS for Oracle DB インスタンスにテーブルスペースのセットをコピーします。物理レベルでは、この転送可能テーブルスペース機能は、ソースデータファイルとメタデータファイルをターゲットインスタンスに増分的にコピーします。ファイルは、Amazon EFS または Amazon S3 のいずれかを使用して転送できます。詳細については、「Oracle トランスポータブル表領域を使用した移行」を参照してください。

トピック

- 転送されたテーブルスペースを DB インスタンスにインポートする
- 転送可能テーブルスペースメタデータを DB インスタンスにインポートする
- テーブルスペースのインポート後の孤立ファイルを一覧表示する

- テーブルスペースのインポート後に孤立したデータファイルを削除する

転送されたテーブルスペースを DB インスタンスにインポートする

プロシージャ rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces を使用して、ソース DB インスタンスから以前にエクスポートしたテーブルスペースを復元します。トランスポートフェーズでは、読み取り専用のテーブルスペースをバックアップし、Data Pump メタデータをエクスポートし、これらのファイルをターゲット DB インスタンスに転送して、テーブルスペースをインポートします。詳細については、「フェーズ 4: 表領域をトランスポートする」を参照してください。

構文

```
FUNCTION import_xtts_tablespaces(
    p_tablespace_list IN CLOB,
    p_directory_name  IN VARCHAR2,
    p_platform_id     IN NUMBER DEFAULT 13,
    p_parallel        IN INTEGER DEFAULT 0) RETURN VARCHAR2;
```

パラメータ

パラメータ名	データ型	デフォルト	必須	説明
p_tablespace_list	CLOB	—	はい	インポートするテーブルスペースのリスト。
p_directory_name	VARCHAR2	—	はい	テーブルスペースのバックアップを含むディレクトリ。
p_platform_id	NUMBER	13	いいえ	バックアップフェーズで指定したものと一致するプラットフォーム ID を指定します。プラットフォームのリストを検索するには、V$TRANSPORTABLE_PLATFORM をクエリします。デフォルトのプラットフォームは

パラメータ名	データ型	デフォルト	必須	説明
				Linux x86 64 ビットで、これはリトルエンディアンです。
p_parallel	INTEGER	0	いいえ	並列処理の度合い。デフォルトでは、並列処理は無効になっています。

例

以下の例では、表領域*TBS1*、*TBS2*、および *TBS3* をディレクトリ *DATA_PUMP_DIR* からインポートします。ソースプラットフォームは AIX ベースのシステム (64 ビット) で、プラットフォーム ID は 6 です。プラットフォーム ID は、V$TRANSPORTABLE_PLATFORM をクエリすることで確認できます。

```
VAR task_id CLOB

BEGIN
  :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces(
        'TBS1,TBS2,TBS3',
        'DATA_PUMP_DIR',
        p_platform_id => 6);
END;
/

PRINT task_id
```

転送可能テーブルスペースメタデータを DB インスタンスにインポートする

プロシージャ rdsadmin.rdsadmin_transport_util.import_xtts_metadata を使用して、転送可能テーブルスペースメタデータを RDS for Oracle DB インスタンスにインポートします。操作中、メタデータのインポートのステータスがテーブル rdsadmin.rds_xtts_operation_info に表示されます。詳細については、「ステップ 5: ターゲット DB インスタンスに表領域メタデータをインポートする」を参照してください。

構文

```
PROCEDURE import_xtts_metadata(
```

```
    p_datapump_metadata_file IN SYS.DBA_DATA_FILES.FILE_NAME%TYPE,
    p_directory_name         IN VARCHAR2,
    p_exclude_stats          IN BOOLEAN DEFAULT FALSE,
    p_remap_tablespace_list  IN CLOB DEFAULT NULL,
    p_remap_user_list        IN CLOB DEFAULT NULL);
```

パラメータ

パラメータ名	データ型	デフォルト	必須	説明
p_datapump_metadata_file	SYS.DBA_DATA_FILES.FILE_NAME%TYPE	—	はい	転送可能テーブルスペースのメタデータを含む Oracle Data Pumpファイルの名前。
p_directory_name	VARCHAR2	—	はい	Data Pump ファイルを含むディレクトリ。
p_exclude_stats	BOOLEAN	FALSE	いいえ	統計を除外するかどうかを示すフラグ。
p_remap_tablespace_list	CLOB	NULL	いいえ	メタデータのインポート中に再マップされるテーブルスペースのリスト。形式 *from_tbs*:*to_tbs* を使用します。例えば、users:user_data と指定します。
p_remap_user_list	CLOB	NULL	いいえ	メタデータのインポート中に再マップされるユーザースキーマのリスト。形式 *from_sche*

パラメータ名	データ型	デフォルト	必須	説明
				ma_name :*to_schema_name* を使用します。例えば、hr:human_resources と指定します。

例

この例では、ディレクトリ *DATA_PUMP_DIR* にあるファイル *xttdump.dmp* からテーブルスペースメタデータをインポートします。

```
BEGIN
  rdsadmin.rdsadmin_transport_util.import_xtts_metadata('xttdump.dmp','DATA_PUMP_DIR');
END;
/
```

テーブルスペースのインポート後の孤立ファイルを一覧表示する

rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files プロシージャを使用して、テーブルスペースのインポート後に孤立したデータファイルを一覧表示します。データファイルを特定したら、rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import を呼び出して削除できます。

構文

```
FUNCTION list_xtts_orphan_files RETURN xtts_orphan_files_list_t PIPELINED;
```

例

次の例では、rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files プロシージャを呼び出します。出力には、孤立した 2 つのデータファイルが表示されます。

```
SQL> SELECT * FROM TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);
```

```
FILENAME       FILESIZE
-------------- ---------
datafile_7.dbf 104865792
datafile_8.dbf 104865792
```

テーブルスペースのインポート後に孤立したデータファイルを削除する

rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files プロシージャを使用して、テーブルスペースのインポート後に孤立したデータファイルを削除します。このコマンドを実行すると、BDUMP ディレクトリに、名前形式 rds-xtts-delete_xtts_orphaned_files-*YYYY-MM-DD.HH24-MI-SS.FF*.log を使用するログファイルが生成されます。プロシージャ rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import を使用して、孤立したファイルを見つけます。プロシージャ rdsadmin.rds_file_util.read_text_file を呼び出すことによって、ログファイルを読み取ることができます。詳細については、「フェーズ 6: 残ったファイルをクリーンアップする」を参照してください。

構文

```
PROCEDURE cleanup_incomplete_xtts_import(
    p_directory_name IN VARCHAR2);
```

パラメータ

パラメータ名	データ型	デフォルト	必須	説明
p_directory_name	VARCHAR2	—	はい	孤立したデータファイルを含むディレクトリ。

例

次の例では、*DATA_PUMP_DIR* にある孤立したデータファイルを削除します。

```
BEGIN
  rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import('DATA_PUMP_DIR');
END;
/
```

次の例では、前のコマンドで生成されたログファイルを読み取ります。

```
SELECT *
FROM TABLE(rdsadmin.rds_file_util.read_text_file(
       p_directory => 'BDUMP',
       p_filename  => 'rds-xtts-
delete_xtts_orphaned_files-2023-06-01.09-33-11.868894000.log'));

TEXT
--------------------------------------------------------------------------------
orphan transported datafile datafile_7.dbf deleted.
orphan transported datafile datafile_8.dbf deleted.
```

RDS for Oracle でストレージを使用する

すべての RDS for Oracle インスタンスにはプライマリストレージボリュームがあります。ストレージ容量を増やすには、DB インスタンスに最大 3 つの追加のストレージボリュームをアタッチできます。ワークロードの要件に応じて、ボリュームごとに gp3 ストレージと io2 ストレージのいずれかを選択します。例えば、頻繁にアクセスされるデータを io2 ボリュームに配置し、履歴データを gp3 ボリュームに配置することができます。

追加のストレージボリュームを使用すると、次の利点が得られます。

- 拡張容量 – 最大 3 つの追加ストレージボリュームをアタッチして、DB インスタンスごとに合計ストレージを最大 256 TiB までスケールします。
- 柔軟なストレージ設定とパフォーマンスの最適化 – さまざまなストレージタイプ (gp3 と io2) を混在させて、データアクセスパターンに基づいてコストとパフォーマンスの両方を最適化します。高性能 io2 ストレージのアクセス頻度の高いデータを、費用対効果の高い gp3 ストレージのアーカイブデータから分離します。
- 必要に応じてストレージ容量を拡張および削減する – データ移行中と同様に、追加のストレージが必要な場合はボリュームをアタッチし、後でボリュームを削除します。このようにして、DB インスタンスストレージの合計を拡張および削減できます。
- オンラインデータ移動 – Oracle データベースの組み込み機能を使用して、ダウンタイムなしでボリューム間でデータを移動します。

> **Note**
>
> 追加のストレージボリュームを削除することはできますが、プライマリボリュームを削除することはできません。

トピック

- RDS for Oracle で追加のストレージボリュームを使用する際の考慮事項
- RDS for Oracle で追加のストレージボリュームを使用する場合の制限
- RDS for Oracle の追加ストレージボリュームを使用したデータベース管理オペレーション
- RDS for Oracle を使用してストレージボリュームを追加、削除、または変更する
- RDS for Oracle で追加のストレージボリュームを使用してデータをバックアップおよび復元する
- RDS for Oracle の追加ストレージボリュームのユースケース

RDS for Oracle で追加のストレージボリュームを使用する際の考慮事項

RDS for Oracle で追加のストレージボリュームを使用する場合は、次の点を考慮してください。

- インスタンスごとに最大 3 つのストレージボリュームを追加できます。
- 追加のストレージボリュームでは、次のボリューム名を使用する必要があります。
  - rdsdbdata2
  - rdsdbdata3
  - rdsdbdata4
- 汎用 SSD (gp3) およびプロビジョンド IOPS SSD (io2) ストレージタイプのみを追加できます。
- Oracle のオンライン再配置機能を使用して、アプリケーションの実行中にボリューム間でデータを移動できます。
- DB インスタンスを変更して追加のストレージボリュームを作成すると、RDS はスケジュールの変更設定に関係なく、ストレージボリュームをすぐに作成します。ストレージボリュームの追加はオンラインオペレーションであり、データベースのパフォーマンスには影響しません。「スケジュール変更設定の使用」を参照してください。

最適なパフォーマンスを得るには、追加のストレージボリュームを使用する際に以下を確認してください。

- データ移動計画
  - オフピーク時に大きな動きをスケジュールする
  - 大きなオペレーションを小さなチャンクに分割する
  - 移動中にシステムリソースをモニタリングする
- リソース管理
  - 両方のボリュームに十分な空き領域を確保する
  - AWR または Statspack を使用して I/O パターンをモニタリングする
  - ストレージフルシナリオを監視する
- ベストプラクティス
  - 可能な場合はオンラインデータファイル再配置オペレーションを使用する
  - 適切なインデックスを維持する
  - スペースの使用状況を定期的にモニタリングする

レプリカで追加のストレージボリュームを使用する場合:

- 追加のストレージボリュームがある DB インスタンスの RDS for Oracle レプリカを作成する場合、RDS はレプリカに追加のストレージボリュームを自動的に設定します。ただし、プライマリ DB インスタンスのストレージボリュームに加えられた後続の変更は、レプリカに自動的に適用されません。
- ボリューム間でデータファイルの場所を管理する場合は、プライマリインスタンスとレプリカインスタンス間の一貫した動作を確保するために、セッションレベルの変更ではなくパラメータグループ設定を使用することをお勧めします。

RDS for Oracle で追加のストレージボリュームを使用する場合の制限

RDS for Oracle での追加のストレージボリュームの使用には、次の制限が適用されます。

- 64GiB 未満のメモリを持つインスタンスタイプには、大きなストレージボリュームをサポートするのに十分なメモリがないため、ストレージボリュームを追加することはできません。
- 追加のストレージボリュームの最小ストレージサイズは 200GiB です。DB インスタンスのプライマリストレージボリュームは、追加のストレージボリュームをアタッチするために 200GiB 以上である必要があります。DB インスタンスの最大ストレージサイズは、すべてのボリュームで合計 256 TiB です。
- 追加のストレージボリュームを持つ DB インスタンスでは、以下の機能はサポートされていません。
  - クロスリージョン自動バックアップ
  - ストレージの自動スケーリング (追加のストレージボリューム用)
  - パブリックスナップショット
- プライマリストレージボリューム (`rdsdbdata`) を削除することはできませんが、空である限り、他の追加のストレージボリュームを削除できます。
- オンライン REDO ログ、アーカイブされた REDO ログ、および制御ファイルを追加のストレージボリュームに保存することはできません。これらのファイルは、プライマリストレージボリューム (`rdsdbdata`) にのみ保存できます。

RDS for Oracle の追加ストレージボリュームを使用したデータベース管理オペレーション

RDS for Oracle で追加のストレージボリュームを使用しながら、テーブルスペースの作成やストレージボリューム間のデータの移動などのデータベース管理オペレーションを実行できます。追加のストレージボリュームを使用したデータベース管理オペレーションの詳細については、以下のセクションを参照してください。

- RDS for Oracle でのデータベースファイルの場所の指定
- RDS for Oracle でのテーブルスペースの作成とサイズ設定
- RDS for Oracle のボリューム間でのデータファイルの移動

RDS for Oracle を使用してストレージボリュームを追加、削除、または変更する

AWS マネジメントコンソールまたは AWS CLI を使用して、追加のストレージボリュームを追加、変更、削除できます。すべてのオペレーションでは、`additional-storage-volumes` パラメータを指定して `modify-db-instance` コマンドを使用します。

> **Important**
>
> ストレージボリュームを追加または削除すると、バックアップ保留中のアクションとブラックアウトウィンドウが作成されます。バックアップワークフローが完了すると、ブラックアウトウィンドウが閉じます。

ストレージボリュームの追加

プライマリストレージボリュームに加えて最大 3 つのストレージボリュームを追加できます。RDS for Oracle DB インスタンスに新しいストレージボリュームを追加するには、`additional-storage-volumes` パラメータを指定して `modify-db-instance` コマンドを使用します。

次のコードスニペットは、4000 プロビジョニング IOPS 名 `rdsdbdata3` を持つ新しい 5,000 GiB の汎用 SSD (gp3) ボリュームを追加します。

```
aws rds modify-db-instance \
```

```
  --db-instance-identifier my-oracle-instance \
  --region us-east-1 \
  --additional-storage-volumes '[
        {
            "VolumeName":"rdsdbdata3",
            "StorageType":"gp3",
            "AllocatedStorage":5000,
            "IOPS":4000}
    ]' \
  --apply-immediately
```

ストレージボリュームの変更

追加のストレージボリュームのストレージタイプ、割り当てられたストレージサイズ、IOPS、ストレージスループット設定を変更できます。次のコードスニペットは、rdsdbdata2 ボリュームの IOPS 設定を変更します。

追加のストレージボリュームを変更すると、そのボリュームはストレージ最適化フェーズに入ります。変更されたボリュームの最適化の進行状況をモニタリングするには、DescribeDBInstances レスポンスの AdditionalStorageVolumes 配列内の対応するエントリを調べます。最適化中、そのエントリの StorageOperationStatus は Optimizing に設定され、StorageOperationPercentProgress は 0 ~ 100 を報告します。詳細については、「ボリュームあたりのストレージオペレーションのモニタリング」を参照してください。

```
aws rds modify-db-instance \
  --db-instance-identifier my-oracle-instance \
  --region us-east-1 \
  --additional-storage-volumes '[
        {
            "VolumeName":"rdsdbdata2",
            "IOPS":8000}
    ]' \
  --apply-immediately
```

> **Note**
>
> インスタンスに追加した追加のストレージボリュームのストレージ割り当てを減らすことはできません。

ストレージボリュームの削除

不要になった追加のストレージボリュームは、RDS for Oracle DB インスタンスから削除できます。ボリュームを削除する前に、ボリュームからすべてのデータベースファイルを移動し、それを参照しているデータベースオブジェクトがないことを確認してください。ボリュームステータスが Not-in-use になっていることを確認します。追加のストレージボリュームを削除することはできますが、プライマリストレージボリュームを削除することはできません。

> **Warning**
>
> 追加のストレージボリュームを削除する前に、データベースファイルがボリュームに保存されていないことを確認してください。アクティブなデータベースファイルを持つボリュームを削除すると、データベースが破損する原因となります。

次の例では、rdsdbdata2 ボリュームを削除しています。

```
aws rds modify-db-instance \
  --db-instance-identifier my-oracle-instance \
  --region us-east-1 \
  --additional-storage-volumes '[
        {
            "VolumeName":"rdsdbdata2",
            "SetForDelete":true}
    ]' \
  --apply-immediately
```

RDS for Oracle で追加のストレージボリュームを使用してデータをバックアップおよび復元する

自動バックアップを使用して、追加のストレージボリュームを含む DB インスタンスで DB スナップショットを作成できます。すべてのバックアップオペレーションには、プライマリボリュームと追加のストレージボリュームの両方が含まれます。追加のストレージボリュームで DB インスタンスのポイントインタイムリカバリを使用することもできます。データベースを復元するときに、ストレージボリュームを追加できます。既存のボリュームのストレージ設定を変更することもできます。スナップショットからデータベースを復元するときに、追加のストレージボリュームを削除することはできません。

トピック

- 手動スナップショットを作成する
- 手動スナップショットの復元
- ポイントインタイムリカバリ

手動スナップショットを作成する

次の例では、追加のストレージボリュームを使用してデータベースの手動スナップショットを作成します。

```
aws rds create-db-snapshot \
--db-instance-identifier my-oracle-asv-instance \
--db-snapshot-identifier my-snapshot
```

手動スナップショットの復元

スナップショットから復元する場合、新しいストレージボリュームを追加したり、既存のボリュームの IOPS またはスループット設定を変更したりできます。次の例では、スナップショットから DB インスタンスを復元し、rdsdbdata2 ボリュームの IOPS 設定を変更します。

```
aws rds restore-db-instance-from-db-snapshot \
  --db-instance-identifier my-restored-instance \
  --db-snapshot-identifier my-snapshot \
  --region us-east-1 \
  --additional-storage-volumes '[
        {
            "VolumeName":"rdsdbdata2",
            "IOPS":5000
        }
    ]'
```

ポイントインタイムリカバリ

ポイントインタイムリカバリ (PITR) 中に、カスタム設定で新しいストレージボリュームを追加できます。次の例では、PITR を実行し、rdsdbdata2 ボリュームに対して 5000 IOPS と 200 MB/秒のストレージスループットを備えた新しい 5,000 GiB 汎用 SSD (gp3) を追加します。

```
aws rds restore-db-instance-to-point-in-time \
  --source-db-instance-identifier my-source-instance \
  --target-db-instance-identifier my-pitr-instance \
  --use-latest-restorable-time \
```

```
--region us-east-1 \
--additional-storage-volumes '[
      {
          "VolumeName":"rdsdbdata2",
          "StorageType":"gp3",
          "AllocatedStorage":5000,
          "IOPS":5000,
          "StorageThroughput":200
      }
  ]'
```

RDS for Oracle の追加ストレージボリュームのユースケース

追加のストレージボリュームは、さまざまなデータベース管理シナリオをサポートします。以下のセクションでは、一般的なユースケースと実装アプローチについて説明します。

トピック

- 64 TiB を超えるストレージ容量の拡張
- アクセス頻度の高いデータとアクセス頻度の低いデータを別々のボリュームにストレージの階層化
- データのロードとアンロード用の一時ストレージ
- 追加のストレージボリュームで Oracle トランスポータブル表領域を使用する

64 TiB を超えるストレージ容量の拡張

プライマリストレージボリュームが 64 TiB の制限に近づいたが、データベースにより多くのストレージ容量が必要な場合、追加のストレージボリュームを使用できます。`modify-db-instance` コマンドを使用して、それぞれ最大 64TiB の追加のストレージボリュームを DB インスタンスにアタッチできます。追加のストレージボリュームをアタッチした後、追加のストレージボリュームにテーブルスペースを作成し、標準の Oracle SQL を使用してテーブル、インデックス、パーティションなどのオブジェクトをこれらのテーブルスペースに移動できます。詳細については、「RDS for Oracle の追加ストレージボリュームを使用したデータベース管理オペレーション」を参照してください。

アクセス頻度の高いデータとアクセス頻度の低いデータを別々のボリュームにストレージの階層化

ボリューム間で異なるストレージタイプを設定することで、追加のストレージボリュームを使用してコストとパフォーマンスを最適化できます。例えば、費用対効果の高い汎用 (gp3) ストレージボ

リュームに履歴データを保存しながら、頻繁にアクセスされるデータに高性能のプロビジョンド IOPS SSD ストレージ (io2) ボリュームを使用できます。標準の Oracle コマンドを使用して、特定のデータベースオブジェクト (テーブル、インデックス、パーティション) をこれらのテーブルスペースに移動できます。詳細については、「RDS for Oracle の追加ストレージボリュームを使用したデータベース管理オペレーション」を参照してください。

データのロードとアンロード用の一時ストレージ

次のステップでは、追加のストレージボリュームを、大規模なデータロードまたはエクスポートの一時ストレージとして使用できます。

- 次のコマンドを使用して、追加のストレージボリュームにディレクトリを作成します。

```
BEGIN
rdsadmin.rdsadmin_util.create_directory(
            p_directory_name => 'DATA_PUMP_DIR2',
            p_database_volume_name => 'rdsdbdata2');
END;
/
```

- ディレクトリを作成したら、「Oracle Data Pump を使用したインポート」で説明されている手順に従って、データを新しいディレクトリにエクスポートおよびインポートします。
- オペレーションが完了したら、ファイルを削除し、オプションでボリュームを削除してストレージコストを節約します。追加のストレージボリュームは、ボリュームが空である場合にのみ削除できます。

追加のストレージボリュームで Oracle トランスポータブル表領域を使用する

追加のストレージボリュームを使用して、Oracle のトランスポータブル表領域を使用して、次のステップでデータファイルを追加のストレージボリュームに移動できます。

- 追加のストレージボリュームを使用してトランスポータブル表領域をターゲットデータベースにインポートする前に、セッションレベルで db_create_file_dest パラメータを設定します。

```
ALTER SESSION SET db_create_file_dest = '/rdsdbdata2/db';

VAR x CLOB;

BEGIN
:x := rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces(
```

```
p_tablespace_list => 'TBTEST1',
p_directory_name => 'XTTS_DIR_DATA2',
p_platform_id => 13);
END;
/

PRINT :x;
```

- トランスポータブル表領域のインポートステータスを確認します。

```
ALTER SESSION SET nls_date_format = 'DD.MM.YYYY HH24:MI:SS';

COL xtts_operation_start_utc FORMAT A30
COL xtts_operation_end_utc FORMAT A30
COL xtts_operation_state FORMAT A30
COL xtts_operation_type FORMAT A30

SELECT xtts_operation_start_utc, xtts_operation_type, xtts_operation_state
FROM rdsadmin.rds_xtts_operation_info;
```

- トランスポータブル表領域のインポートが完了したら、トランスポータブル表領域メタデータをインポートします。

```
BEGIN
rdsadmin.rdsadmin_transport_util.import_xtts_metadata(
p_datapump_metadata_file => 'xttdump.dmp',
p_directory_name => 'XTTS_DIR_DATA2');
END;
/
```

RDS for Oracle の高度な機能の設定

RDS for Oracle は、HugePages、インスタンスストア、拡張データ型など、さまざまな高度な機能をサポートしています。

トピック

- RDS for Oracle インスタンスストアへの一時データの保存
- サポートされている RDS for Oracle インスタンスで HugePages をオンにする
- RDS for Oracle で拡張データ型を有効にする

RDS for Oracle インスタンスストアへの一時データの保存

サポートされている RDS for Oracle DB のインスタンスクラスで、一時テーブルスペースとデータベースのスマートフラッシュキャッシュ (フラッシュキャッシュ) 用のインスタンスストアを使用します。

トピック

- RDS for Oracle インスタンスストアの概要
- RDS for Oracle インスタンスストアの有効化
- RDS for Oracle インスタンスストアの設定
- Oracle リードレプリカ上のインスタンスストアでの作業
- インスタンスストアと Amazon EBS の一時テーブルスペースグループの設定
- RDS for Oracle インスタンスストアの削除

RDS for Oracle インスタンスストアの概要

インスタンスストアは、RDS for Oracle DB インスタンスに一時ブロックレベルのストレージを提供します。インスタンスストアは、頻繁に変更される情報を一時的に保存するために使用できます。

インスタンスストアは、ホストコンピュータに物理的にアタッチされた不揮発性メモリエクスプレス (NVMe) デバイスをベースにしています。このストレージは、低レイテンシー、ランダム I/O パフォーマンス、シーケンシャル読み取りスループットを実現するために最適化されています。

インスタンスストアのサイズは DB インスタンスタイプによって異なります。インスタンスストアの詳細については、Linux インスタンス向け Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 インスタンスストア」を参照してください。

トピック

- RDS for Oracle インスタンスストア内のデータのタイプ
- RDS for Oracle インスタンスストアの利点
- サポートされている RDS for Oracle インスタンスストアのインスタンスクラス
- RDS for Oracle インスタンスストアでサポートされているエンジンバージョン
- RDS for Oracle インスタンスストアでサポートされている AWS リージョン
- RDS for Oracle インスタンスストアのコスト

RDS for Oracle インスタンスストア内のデータのタイプ

インスタンスストアには、次のタイプの RDS for Oracle 一時データを配置できます。

一時テーブルスペース

Oracle Database は、一時テーブルスペースを使用して、メモリに収まらない中間クエリ結果を保存します。クエリのサイズが大きいと、一時的にキャッシュする必要はあるが、永続化する必要はない大量の中間データが生成されることがあります。特に、一時テーブルスペースは、ソート、ハッシュ集計、および結合に役立ちます。RDS for Oracle DB インスタンスがエンタープライズエディションまたはスタンダードエディション 2 を使用している場合は、インスタンスストアに一時テーブルスペースを配置できます。

フラッシュキャッシュ

フラッシュキャッシュは、従来のパスでの単一ブロックのランダム読み取りのパフォーマンスを向上させます。ベストプラクティスは、アクティブなデータセットのほとんどを収容できるようにキャッシュのサイズを設定することです。RDS for Oracle DB インスタンスがエンタープライズエディションを使用している場合は、フラッシュキャッシュをインスタンスストアに配置できます。

デフォルトでは、インスタンスストアは一時テーブルスペース用に設定されていますが、フラッシュキャッシュ用には設定されていません。Oracle データファイルとデータベースログファイルは、インスタンスストアに配置できません。

RDS for Oracle インスタンスストアの利点

失ってもかまわない一時ファイルとキャッシュを保存するためにインスタンスストアを使用することを検討してください。DB のパフォーマンスを改善したい場合、またはワークロードの増加が

Amazon EBS ストレージのパフォーマンス上の問題を引き起こしている場合は、インスタンスストアをサポートするインスタンスクラスにスケーリングすることを検討してください。

一時テーブルスペースとフラッシュキャッシュをインスタンスストアに配置すると、次のような利点があります。

- 読み込みのレイテンシーを短縮
- 高スループット
- Amazon EBS ボリュームの負荷を軽減
- Amazon EBS の負荷が軽減されるため、ストレージとスナップショットのコストが削減されます
- 高い IOPS をプロビジョニングする必要性が減り、全体的なコストが下がる可能性がある

一時的なテーブルスペースをインスタンスストアに配置すると、一時スペースを使用するクエリのパフォーマンスがすぐに向上します。フラッシュキャッシュをインスタンスストアに配置すると、通常、キャッシュされたブロック読み取りのレイテンシーは Amazon EBS の読み取りよりもはるかに短くなります。フラッシュキャッシュは、パフォーマンスを向上させる前に「ウォームアップ」する必要があります。データベースバッファキャッシュからブロックが期限切れになると、データベースがフラッシュキャッシュにブロックを書き込むため、キャッシュは自動的にウォームアップします。

> **Note**
>
> キャッシュ管理のため、フラッシュキャッシュが原因でパフォーマンスのオーバーヘッドが発生する場合があります。本番環境でフラッシュキャッシュを有効にする前に、ワークロードを分析し、テスト環境でキャッシュをテストすることをお勧めします。

サポートされている RDS for Oracle インスタンスストアのインスタンスクラス

Amazon RDS は、次の DB インスタンスクラスのインスタンスストアをサポートしています。

- db.m5d
- db.m6id
- db.r5d
- db.r6id
- db.x2idn
- db.x2iedn

RDS for Oracle は、BYOL ライセンスモデルでのみ前述の DB インスタンスクラスをサポートします。詳細については、「サポートされている RDS for Oracle DB インスタンスクラス」および「EE および SE2 の Bring Your Own License (BYOL)」を参照してください。

サポート対象の DB インスタンスタイプのインスタンスストレージの合計を確認するには、AWS CLI で次のコマンドを実行します。

Example

```
aws ec2 describe-instance-types \
  --filters "Name=instance-type,Values=*5d.*large*,*6id.*large*" \
  --query "InstanceTypes[?contains(InstanceType,'m5d')||contains(InstanceType,'r5d')||
contains(InstanceType,'m6id')||contains(InstanceType,'r6id')][InstanceType,
 InstanceStorageInfo.TotalSizeInGB]" \
  --output table
```

前述のコマンドは、インスタンスストアの未フォーマットデバイスサイズを返します。RDS for Oracle は、このスペースのごく一部を設定に使用します。一時テーブルスペースまたはフラッシュキャッシュに使用できるインスタンスストア内のスペースは、少し小さくなっています。

RDS for Oracle インスタンスストアでサポートされているエンジンバージョン

インスタンスストアは、以下の RDS for Oracle エンジンバージョンでサポートされています。

- すべての Oracle Database 26ai バージョン
- 21.0.0.0.ru-2022-01.rur-2022-01.r1 以上の Oracle Database 21c バージョン
- 19.0.0.0.ru-2021-10.rur-2021-10.r1 以上の Oracle Database 19c バージョン

RDS for Oracle インスタンスストアでサポートされている AWS リージョン

インスタンスストアは、これらのインスタンスタイプの 1 つ以上がサポートされているすべての AWS リージョン で使用できます。db.m5d と db.r5d インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。Amazon RDS for Oracle でサポートされるインスタンスクラスの詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。

RDS for Oracle インスタンスストアのコスト

インスタンスストアのコストは、有効になっているインスタンスストアのコストに組み込まれます。RDS for Oracle DB インスタンスでインスタンスストアを有効にしても、追加料金は発生しま

せん。instance-store turned on インスタンスの詳細については、「サポートされている RDS for Oracle インスタンスストアのインスタンスクラス」を参照してください。

RDS for Oracle インスタンスストアの有効化

RDS for Oracle テンポラリデータのインスタンスストアを有効にするには、次のいずれかの操作を行います。

- サポートされているインスタンスクラスを使用して RDS for Oracle DB インスタンスを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- サポートされているインスタンスクラスを使用するように既存の RDS for Oracle DB インスタンスを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS for Oracle インスタンスストアの設定

デフォルトでは、インスタンスストア領域の 100% が一時テーブルスペースに割り当てられます。フラッシュキャッシュと一時テーブルスペースに領域を割り当てるようにインスタンスストアを設定するには、インスタンスのパラメータグループに次のパラメータを設定します。

db_flash_cache_size={DBInstanceStore*{0,2,4,6,8,10}/10}

このパラメータは、フラッシュキャッシュに割り当てられるストレージ容量を指定します。このパラメータは、Oracle Database Enterprise Edition でのみ有効です。デフォルト値は `{DBInstanceStore*0/10}` です。`db_flash_cache_size` にゼロ以外の値を設定した場合、RDS for Oracle インスタンスはインスタンスの再起動後にフラッシュキャッシュを有効にします。

rds.instance_store_temp_size={DBInstanceStore*{0,2,4,6,8,10}/10}

このパラメータは、一時テーブルスペースに割り当てられるストレージスペースの量を指定します。デフォルト値は `{DBInstanceStore*10/10}` です。このパラメータは Oracle Database Enterprise Edition では変更可能で、Standard Edition 2 では読み取り専用です `rds.instance_store_temp_size` にゼロ以外の値を設定すると、Amazon RDS はインスタンスストアのスペースを一時テーブルスペースに割り当てます。

インスタンスストアを使用しない DB インスタンスには、`db_flash_cache_size` および `rds.instance_store_temp_size` パラメータを設定できます。この場合、両方の設定が `0` に評価され、機能がオフになります。この場合、異なるインスタンスサイズやインスタンスストア

を使用しないインスタンスに同じパラメータグループを使用できます。これらのパラメータを変更する場合は、変更を反映させるために関連するインスタンスを再起動してください。

> **Important**
>
> 一時テーブルスペースにスペースを割り当てても、Amazon RDS は一時テーブルスペースを自動的に作成しません。インスタンスストアに一時テーブルスペースを作成する方法については、「インスタンスストアに一時テーブルスペースを作成する」を参照してください。

前述のパラメータの合計値は、10/10 または 100% を超えてはなりません。次の表は、有効なパラメータ設定と無効なパラメータ設定を示しています。

db_flash_cache_size 設定	rds.instance_store_temp_size 設定	説明
db_flash_cache_size={DBInstanceStore*0/10}	rds.instance_store_temp_size={DBInstanceStore*10/10}	これは、Oracle Database のすべてのエディションで有効な設定です。Amazon RDS は、インスタンスストアスペースの 100% を一時テーブルスペースに割り当てます。これがデフォルトです。
db_flash_cache_size={DBInstanceStore*10/10}	rds.instance_store_temp_size={DBInstanceStore*0/10}	これは、Oracle Database Enterprise Edition でのみ有効な設定です。Amazon

db_flash_cache_size 設定	rds.instance_store_temp_size 設定	説明
		RDS はインスタンスストア容量の 100% をフラッシュキャッシュに割り当てます。
db_flash_cache_size={DBInstanceStore*2/10}	rds.instance_store_temp_size={DBInstanceStore*8/10}	これは、Oracle Database Enterprise Edition でのみ有効な設定です。Amazon RDS はインスタンスストア容量の 20% をフラッシュキャッシュに、インスタンスストア容量の 80% を一時テーブルスペースに割り当てます。

db_flash_cache_size 設定	rds.instance_store_temp_size 設定	説明
db_flash_cache_size={DBInstanceStore*6/10}	rds.instance_store_temp_size={DBInstanceStore*4/10}	これは、Oracle Database Enterprise Edition でのみ有効な設定です。Amazon RDS は、インスタンスストア容量の 60% をフラッシュキャッシュに、40% のインスタンスストア容量を一時テーブルスペースに割り当てます。
db_flash_cache_size={DBInstanceStore*2/10}	rds.instance_store_temp_size={DBInstanceStore*4/10}	これは、Oracle Database Enterprise Edition でのみ有効な設定です。Amazon RDS は、インスタンスストア容量の 20% をフラッシュキャッシュに、40% のインスタンスストア容量を一時テーブルスペースに割り当てます。

db_flash_cache_size 設定	rds.instance_store_temp_size 設定	説明
db_flash_cache_size={DBInstanceStore*8/10}	rds.instance_store_temp_size={DBInstanceStore*8/10}	インスタンスストア容量の合計パーセンテージが 100% を超えているため、これは無効な設定です。このような場合、Amazon RDS は試行に失敗します。

DB インスタンスタイプを変更する際の考慮事項

DB インスタンスタイプを変更すると、インスタンスストアのフラッシュキャッシュまたは一時テーブルスペースの設定に影響する可能性があります。以下の変更とその影響を考慮してください。

インスタンスストアをサポートする DB インスタンスをスケールアップまたはスケールダウンします。

次の値は、インスタンスストアの新しいサイズに比例して増減します。

- フラッシュキャッシュの新しいサイズ。
- インスタンスストアにある一時テーブルスペースに割り当てられた容量。

例えば、db.m5d.4xlarge インスタンスで `db_flash_cache_size={DBInstanceStore*6/10}` を設定すると約 340 GB のフラッシュキャッシュ容量が提供されます。インスタンスタイプを db.m5d.8xlarge にスケールアップすると、フラッシュキャッシュ容量は約 680 GB に増加します。

インスタンスストアを使用しない DB インスタンスを、インスタンスストアを使用するインスタンスに変更します。

`db_flash_cache_size` を `0` より大きい値に設定すると、フラッシュキャッシュが設定されます。`rds.instance_store_temp_size` を `0` より大きい値に設定すると、インスタンスストア領域は一時テーブルスペースが使用できるように割り当てられます。RDS for Oracle は、一時ファイルをインスタンスストアに自動的に移動しません。割り当てられたスペースの使用方法に

ついては、「インスタンスストアに一時テーブルスペースを作成する」または「リードレプリカのインスタンスストアへの一時ファイルの追加」を参照してください。

インスタンスストアを使用する DB インスタンスを、インスタンスストアを使用しないインスタンスに変更します。

この場合、RDS for Oracle はフラッシュキャッシュを削除します。RDS は、Amazon EBS ボリュームのインスタンスストアに現在置かれている一時ファイルを再作成します。新しい一時ファイルの最大サイズは、`rds.instance_store_temp_size` パラメータの以前のサイズです。

Oracle リードレプリカ上のインスタンスストアでの作業

リードレプリカは、インスタンスストアのフラッシュキャッシュと一時テーブルスペースをサポートします。フラッシュキャッシュはプライマリ DB インスタンスと同じように機能しますが、一時テーブルスペースについては以下の違いに注意してください。

- リードレプリカでは一時テーブルスペースを作成できません。プライマリインスタンスに新しい一時テーブルスペースを作成すると、RDS for Oracle は一時ファイルなしでテーブルスペース情報をレプリケートします。新しい一時ファイルを追加するには、次のいずれかの操作を行います。
  - Amazon RDS プロシージャ `rdsadmin.rdsadmin_util.add_inst_store_tempfile` を使用します。RDS for Oracle は、リードレプリカのインスタンスストアに一時ファイルを作成し、指定された一時テーブルスペースに追加します。
  - `ALTER TABLESPACE … ADD TEMPFILE` コマンドを実行します。RDS for Oracle は一時ファイルを Amazon EBS ストレージに配置します。

> **Note**
>
> 一時ファイルのサイズとストレージタイプは、プライマリ DB インスタンスとリードレプリカで異なる場合があります。

- デフォルトの一時テーブルスペース設定は、プライマリ DB インスタンスでのみ管理できます。RDS for Oracle は、すべてのリードレプリカに設定を複製します。
- 一時テーブルスペースグループはプライマリ DB インスタンスでのみ設定できます。RDS for Oracle は、すべてのリードレプリカに設定を複製します。

インスタンスストアと Amazon EBS の一時テーブルスペースグループの設定

インスタンスストアと Amazon EBS の両方の一時テーブルスペースを含むように一時テーブルスペースグループを設定できます。この方法は、`rds.instance_store_temp_size` の最大設定で許容される容量を超える一時ストレージが必要な場合に便利です。

インスタンスストアと Amazon EBS の両方で一時テーブルスペースグループを設定すると、2 つのテーブルスペースのパフォーマンス特性が大きく異なります。Oracle Database は、内部アルゴリズムに基づいてクエリを処理するテーブルスペースを選択します。そのため、同様のクエリではパフォーマンスが異なる場合があります。

通常、次のようにインスタンスストアに一時テーブルスペースを作成します。

1. インスタンスストアに一時テーブルスペースを作成します。
2. 新しいテーブルスペースをデータベースのデフォルトの一時テーブルスペースとして設定します。

インスタンスストアのテーブルスペースサイズが不十分な場合は、次のように追加の一時ストレージを作成できます。

1. インスタンスストアの一時テーブルスペースを一時テーブルスペースグループに割り当てます。
2. Amazon EBS に新しい一時テーブルスペースがない場合は、作成してください。
3. Amazon EBS の一時テーブルスペースを、インスタンスストアテーブルスペースを含む同じテーブルスペースグループに割り当てます。
4. テーブルスペースグループをデフォルトの一時テーブルスペースとして設定します。

次の例では、インスタンスストア内の一時テーブルスペースのサイズがアプリケーションの要件を満たしていないことを前提としています。この例では、インスタンスストアに一時テーブルスペース `temp_in_inst_store` を作成し、それをテーブルスペースグループ `temp_group` に割り当て、このグループに `temp_in_ebs` という名前の既存の Amazon EBS テーブルスペースを追加し、このグループをデフォルトの一時テーブルスペースとして設定します。

```
SQL> EXEC rdsadmin.rdsadmin_util.create_inst_store_tmp_tblspace('temp_in_inst_store');

PL/SQL procedure successfully completed.

SQL> ALTER TABLESPACE temp_in_inst_store TABLESPACE GROUP temp_group;
```

```
Tablespace altered.

SQL> ALTER TABLESPACE temp_in_ebs TABLESPACE GROUP temp_group;

Tablespace altered.

SQL> EXEC rdsadmin.rdsadmin_util.alter_default_temp_tablespace('temp_group');

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DBA_TABLESPACE_GROUPS;

GROUP_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------
TEMP_GROUP                     TEMP_IN_EBS
TEMP_GROUP                     TEMP_IN_INST_STORE

SQL> SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE
 PROPERTY_NAME='DEFAULT_TEMP_TABLESPACE';

PROPERTY_VALUE
--------------
TEMP_GROUP
```

RDS for Oracle インスタンスストアの削除

インスタンスストアを削除するには、インスタンスストアをサポートしないインスタンスタイプ (db.m5 または db.r5) を使用するように RDS for Oracle DB インスタンスを変更します。

サポートされている RDS for Oracle インスタンスで HugePages をオンにする

Amazon RDS for Oracle は、データベースのスケーラビリティを増大する Linux Kernel の HugePages をサポートしています。HugePages により、ページのテーブルを小さくし、メモリ管理の CPU 経過時間を減少することで、大規模なデータベースインスタンスのパフォーマンスを向上できます。詳細については、Oracle ドキュメントの「HugePages の概要」を参照してください。

HugePages は、RDS for Oracle の次のバージョンとエディションで使用できます。

`use_large_pages` パラメータは、DB インスタンスで HugePages を有効にするかどうかを制御します。このパラメータに設定できる値は、`ONLY`、`FALSE`、および `{DBInstanceClassHugePagesDefault}` です。Oracle のデフォルト DB パラメータグループで

は、use_large_pages パラメータが {DBInstanceClassHugePagesDefault} に設定されます。

DB インスタンスで HugePages を自動的に有効にするかどうかを制御するには、DBInstanceClassHugePagesDefault 式の可変をパラメータグループで使用します。値は次のように決定されます。

- 以下の表に示す DB インスタンスクラスの場合、DBInstanceClassHugePagesDefault はデフォルトで常に FALSE と評価されます。use_large_pages は FALSE と評価されます。DB インスタンスクラスのメモリが 14 GiB 以上であれば、これらの DB インスタンスクラス用に HugePages を手動で有効化できます。
- 以下の表に示していない DB インスタンスクラスで、DB インスタンスクラスのメモリが 14 GiB 未満の場合は、DBInstanceClassHugePagesDefault は常に FALSE と評価されます。また、use_large_pages は FALSE と評価されます。
- 以下の表に示していない DB インスタンスクラスで、インスタンスクラスのメモリが 14 GiB 以上、100 GiB 未満の場合は、DBInstanceClassHugePagesDefault はデフォルトで TRUE と評価されます。また、use_large_pages は ONLY と評価されます。HugePages を手動で無効にするには、use_large_pages を FALSE に設定します。
- 次の表に示していない DB インスタンスクラスで、インスタンスクラスのメモリが 100 GiB 以上の場合は、DBInstanceClassHugePagesDefault は常に TRUE と評価されます。また、use_large_pages は ONLY と評価され、HugePages を無効にすることはできません。

HugePages は、以下の DB インスタンスクラスに対してはデフォルトで有効になりません。

DB インスタンスクラスファミリー	HugePages がデフォルトで有効になっていない DB インスタンスクラス
db.m5	db.m5.large
db.m4	db.m4.large、db.m4.xlarge、db.m4.2xlarge、db.m4.4xlarge、db.m4.10xlarge
db.t3	db.t3.micro、db.t3.small、db.t3.medium、db.t3.large

DB インスタンスクラスの詳細については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

新規または既存の DB インスタンスで HugePages を手動で有効にするには、use_large_pages パラメータを ONLY に設定します。Oracle 自動メモリ管理 (AMM) では HugePages を使用できません。use_large_pages パラメータを ONLY に設定するには、memory_target と memory_max_target の両方を 0 に設定する必要もあります。DB インスタンスの DB パラメータを設定する詳細については、「Amazon RDS のパラメータグループ」を参照してください。

sga_target、sga_max_size と pga_aggregate_target パラメータも設定できます。システムグローバルエリア (SGA) とプログラムグローバルエリア (PGA) のメモリパラメータを設定する場合には、値をまとめて追加します。この合計を使用可能なインスタンスメモリ (DBInstanceClassMemory) から減算して、HugePages の割当量を超える空きメモリを判断します。使用可能なインスタンスメモリ全体の少なくとも 10% または 2 GiB のどちらか少ない方を空きメモリとして残す必要があります。

パラメータを設定したら、DB インスタンスを再起動して変更を有効にする必要があります。詳細については、「 DB インスタンスの再起動」を参照してください。

> **Note**
>
> Oracle DB インスタンスは、フェイルオーバーなしでインスタンスが再起動されるまで、SGA 関連の初期化パラメータへの変更が延期されます。Amazon RDS コンソールで [Reboot (再起動)] を選択しますが、[Reboot with failover (フェイルオーバーありで再起動)] を選択しないでください。AWS CLI で、reboot-db-instance パラメータを指定して --no-force-failover コマンドを呼び出します。DB インスタンスは、フェイルオーバー中、またはインスタンスを再起動させる他のメンテナンスオペレーション中に、SGA 関連のパラメータを処理しません。

HugePages を手動で有効化する場合の HugePages のパラメータ設定の例を次に示します。必要に応じて値を設定してください。

```
memory_target            = 0
memory_max_target        = 0
pga_aggregate_target     = {DBInstanceClassMemory*1/8}
sga_target               = {DBInstanceClassMemory*3/4}
sga_max_size             = {DBInstanceClassMemory*3/4}
use_large_pages          = ONLY
```

パラメータグループに以下のパラメータ値を設定したとします。

```
memory_target            = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target        = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target     = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target               = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size             = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages          = {DBInstanceClassHugePagesDefault}
```

パラメータグループは、100 GiB 未満のメモリを持つ db.r4 DB インスタンスクラスによって使用されます。これらのパラメータ設定と `use_large_pages` を `{DBInstanceClassHugePagesDefault}` に設定した場合、HugePages は db.r4 インスタンスで有効になります。

別の例として、パラメータグループに以下のパラメータ値を設定した場合を考えます。

```
memory_target           = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target       = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target    = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target              = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size            = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages         = FALSE
```

パラメータグループは、メモリが 100 GiB 未満の DB インスタンスクラス db.r4 と db.r5 インスタンスの両方で使用されます。これらのパラメータ設定では、db.r4 および db.r5 インスタンスで HugePages が無効になります。

> **Note**
>
> このパラメータグループをメモリが 100 GiB 以上の db.r4 DB インスタンスクラス、または db.r5 DB インスタンスクラスで使用すると、`FALSE` の `use_large_pages` 設定はオーバー

ライドされ、ONLY に設定されます。この場合、オーバーライドに関する通知がユーザーに送信されます。

DB インスタンスで HugePages が有効になると、拡張モニタリングを有効にして HugePages の情報を表示できます。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

RDS for Oracle で拡張データ型を有効にする

Amazon RDS for Oracle は、拡張データ型をサポートします。拡張データ型では、VARCHAR2、NVARCHAR2、および RAW データ型の最大サイズは 32,767 バイトです。拡張データ型を使用する場合、MAX_STRING_SIZE パラメータを EXTENDED に設定します。詳細については、Oracle ドキュメントの「拡張データ型」を参照してください。

拡張データ型を使用しない場合は、MAX_STRING_SIZE パラメータを STANDARD (デフォルト) のままにします。この場合、サイズ制限は、VARCHAR2 および NVARCHAR2 データ型で 4,000 バイト、RAW データ型で 2,000 バイトです。

新しい DB インスタンス、または既存の DB インスタンスで拡張データ型を有効にできます。新しい DB インスタンスの場合、拡張データ型を有効にすると、通常、DB インスタンスの作成時間が長くなります。既存の DB インスタンスでは、変換プロセス中は DB インスタンスは無効です。

拡張データ型に関する考慮事項

DB インスタンスの拡張データ型を有効にするときには、次の点を考慮してください。

- 新規または既存の DB インスタンスの拡張データ型を有効にする場合は、変更を有効にするためにインスタンスを再起動する必要があります。
- 拡張データ型を有効にすると、DB インスタンスでデータ型の標準サイズを使用するように戻すことはできません。MAX_STRING_SIZE パラメータを STANDARD に戻すと、incompatible-parameters ステータスになります。

> **Important**
>
> 拡張データ型の有効化は元に戻せません。MAX_STRING_SIZE を EXTENDED に設定した後は、STANDARD に戻すことはできません。つまり、拡張データ型をサポートしていないエンジンバージョンにダウングレードすることはできません。拡張列を使用するテーブル

の論理エクスポート (Data Pump) は、STANDARD モードを使用するデータベースにインポートできない可能性があります。

- 拡張データ型を使用する DB インスタンスを復元する場合、MAX_STRING_SIZE パラメータを EXTENDED に設定して、パラメータグループを設定する必要があります。復元中は、デフォルトのパラメータグループ、または、他のパラメータグループの MAX_STRING_SIZE を STANDARD に設定すると、incompatible-parameters ステータスになります。
- DB インスタンスステータスが incompatible-parameters の設定のために MAX_STRING_SIZE の場合、MAX_STRING_SIZE パラメータを EXTENDED に設定して DB インスタンスを再起動するまで、DB インスタンスは使用できません。

新しい DB インスタンスで拡張データ型を有効にする

MAX_STRING_SIZE を EXTENDED に設定して DB インスタンスを作成すると、インスタンスの MAX_STRING_SIZE はデフォルトの STANDARD に設定されます。インスタンスを再起動して、変更を有効にします。

新しい DB インスタンスで拡張データ型を有効にするには

1. パラメータグループ内の MAX_STRING_SIZE パラメータを EXTENDED に設定する。

   新しいパラメータグループを作成するか、既存のパラメータグループを変更して、パラメータを設定できます。

   詳細については、「Amazon RDS のパラメータグループ」を参照してください。
2. 新しい RDS for Oracle DB インスタンスを作成します。

   詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
3. パラメータグループを、DB インスタンスで EXTENDED に設定された MAX_STRING_SIZE に関連付けます。

   詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
4. DB インスタンスを再起動してパラメータの変更を有効にします。

   詳細については、「 DB インスタンスの再起動」を参照してください。

既存の DB インスタンスで拡張データ型を有効にするには

DB インスタンスを変更して拡張データ型を有効にすると、RDS は、拡張サイズを使用するようにデータベース内のデータを変換します。変換とダウンタイムは、パラメータの変更後にデータベースを次に再起動したときに発生します。DB インスタンスは変換中は利用できません。

データの変換に要する時間は、DB インスタンスクラス、データベースのサイズ、および前回の DB スナップショットの時刻によって異なります。ダウンタイムを減らすには、再起動の直前にスナップショットを取ることを検討してください。これにより、変換ワークフロー中に行われるバックアップの時間が短縮されます。

> **Note**
>
> 拡張データ型を有効にすると、変換中はポイントインタイムの復元を実行できません。変換の直前または変換後の時間に復元することができます。

既存の DB インスタンスで拡張データ型を有効にするには

1. データベースのスナップショットを作成します。

   データベースに無効なオブジェクトがある場合、Amazon RDS はそれらの再コンパイルを試みます。Amazon RDS が無効なオブジェクトを再コンパイルできない場合、拡張データ型への変換は失敗する可能性があります。スナップショットを使用すると、変換に問題がある場合にデータベースを復元できます。変換前に無効なオブジェクトがないかを常に確認して、無効なオブジェクトを修正または削除してください。本番データベースの場合は、初期に DB インスタンスのコピーで変換プロセスをテストすることをお勧めします。

   詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
2. パラメータグループ内の `MAX_STRING_SIZE` パラメータを `EXTENDED` に設定する。

   新しいパラメータグループを作成するか、既存のパラメータグループを変更して、パラメータを設定できます。

   詳細については、「Amazon RDS のパラメータグループ」を参照してください。
3. DB インスタンスを修正して `MAX_STRING_SIZE` を `EXTENDED` に設定したパラメータグループと関連付けます。

詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

4. DB インスタンスを再起動してパラメータの変更を有効にします。

   詳細については、「 DB インスタンスの再起動」を参照してください。

Oracle の AI 機能

RDS for Oracle は、Oracle データベースから直接生成 AI 機能を使用できる AI 機能をサポートしています。

トピック

- Amazon Bedrock と RDS for Oracle の統合

Amazon Bedrock と RDS for Oracle の統合

RDS for Oracle は Amazon Bedrock との統合をサポートしているため、SQL を使用して Oracle Database から直接基盤モデルを呼び出すことができます。この統合により、データベースからデータを移動することなく、テキスト生成、要約、埋め込み生成などの生成 AI 機能を使用できます。

> **Note**
>
> Amazon Bedrock 統合は、Oracle Database 26ai でのみサポートされています。

トピック

- Bedrock 統合の要件
- Bedrock の統合を設定する
- Bedrock 統合の使用

Bedrock 統合の要件

Amazon Bedrock と RDS for Oracle の統合を使用する場合、次の要件が満たされていることを確認してください。

データベース要件

- DB インスタンスは Oracle Database 26ai (26.0.0.0) 以降を実行している必要があります。
- Bedrock モデルを呼び出すデータベースユーザーは、`DBMS_CLOUD` および `DBMS_CLOUD_AI` パッケージに対する `EXECUTE` 権限を有している必要があります。
- データベースには、Bedrock ランタイムエンドポイントへのアウトバウンド HTTPS アクセスを許可するアクセスコントロールエントリ (ACE) を設定する必要があります。詳細については、「Bedrock の統合を設定する」を参照してください。

AWS 認証情報の要件

- 必要な Amazon Bedrock 基盤モデル (`bedrock:InvokeModel` およびオプションで `bedrock:InvokeModelWithResponseStream` アクション) を呼び出すためのアクセス許可を持つ IAM ユーザーまたはロールが必要です。
- Amazon Bedrock コンソールを使用して、必要な基盤モデルへのアクセスをリクエストし、許可されている必要があります。詳細については、「Amazon Bedrock ユーザーガイド」の「Amazon Bedrock 基盤モデルにアクセスする」を参照してください。
- `DBMS_CLOUD.CREATE_CREDENTIAL` を使用してデータベースに保存するには、IAM ID の AWS アクセスキー (アクセスキー ID とシークレットアクセスキー) が必要です。

Amazon VPC ネットワークの要件

RDS for Oracle DB インスタンスは、ポート 443 (HTTPS) の Amazon Bedrock ランタイムエンドポイント (`bedrock-runtime.`*`region`*`.amazonaws.com`) に到達できる必要があります。これは、次のいずれかの方法で行うことができます。

オプション 1: Amazon VPC インターフェイスエンドポイント (AWS PrivateLink) – 推奨

Amazon Bedrock ランタイムサービスの Amazon VPC インターフェイスエンドポイントを作成します。これにより、AWS ネットワーク内のすべてのトラフィックがプライベートに保たれます。インターネットアクセスは必要ありません。

1. サービス名 `com.amazonaws.`*`region`*`.bedrock-runtime` を使用して、Amazon VPC インターフェイスエンドポイントを作成します。
2. エンドポイントを Amazon RDS DB インスタンスと同じ Amazon VPC とサブネット (または DB インスタンスサブネットに接続されているサブネット) に配置します。
3. DB インスタンスのセキュリティグループからのインバウンド HTTPS (ポート 443) を許可するセキュリティグループをエンドポイントにアタッチします。
4. 標準のホスト名 `bedrock-runtime.`*`region`*`.amazonaws.com` がプライベートエンドポイント IP アドレスに解決されるように、エンドポイントのプライベート DNS を有効にします。
5. DB インスタンスのセキュリティグループが、エンドポイントのセキュリティグループ (またはエンドポイントのプライベート IP アドレス) へのアウトバウンド HTTPS (ポート 443) を許可していることを確認します。

オプション 2: NAT ゲートウェイ (インターネットルート)

Amazon VPC エンドポイントを使用しない場合、DB インスタンスのサブネットには NAT ゲートウェイを介してインターネットへのルートが設定されている必要があります。

1. DB インスタンスが NAT ゲートウェイへのルートを持つプライベートサブネットに属していることを確認します。
2. NAT ゲートウェイがパブリックサブネット内に存在し、インターネットゲートウェイにルーティングされるように設定されている必要があります。
3. DB インスタンスのセキュリティグループは、`bedrock-runtime.region.amazonaws.com` へのアウトバウンド HTTPS (ポート 443) を許可する必要があります。
4. DB インスタンスのサブネット上の Amazon VPC ネットワーク ACL は、アウトバウンド TCP 443 および対応する一時的なインバウンドリターントラフィック (TCP ポート 1024 ~ 65535) を許可する必要があります。

次の表は、ネットワークパスを要約したものです。

Bedrock 統合のネットワークパス

アプローチ	インターネットアクセスの必要性	トラフィックパス
Amazon VPC エンドポイント (AWS PrivateLink)	いいえ	DB インスタンス → Amazon VPC エンドポイント → Bedrock サービス
NAT ゲートウェイ	はい (アウトバウンドのみ)	DB インスタンス → NAT ゲートウェイ → インターネットゲートウェイ → Bedrock サービス

Bedrock の統合を設定する

RDS for Oracle DB インスタンスの Amazon Bedrock 統合を設定するには、次の手順を実行します。

ステップ 1: Amazon VPC ネットワーク接続を設定する

「Amazon VPC ネットワークの要件」で説明されているネットワークオプションのいずれかを使用して、DB インスタンスが Bedrock ランタイムエンドポイントに到達できることを確認します。設定後に接続を確認するには、後のステップでデータベースからテストできます。

ステップ 2: Bedrock アクセス許可を持つ AWS 認証情報を取得する

次のポリシーを使用して IAM ユーザーまたはロールを作成します (または既存の ID にアタッチします)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BedrockInvoke",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream"
      ],
      "Resource": [
        "arn:aws:bedrock:region::foundation-model/*"
      ]
    }
  ]
}
```

特定のモデルへのアクセスを制限するには、ワイルドカードを特定のモデル ARN に置き換えます。次に例を示します。

```
"arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-sonnet-4-5-20250929-v1:0"
```

IAM コンソールで、この IAM ユーザーのアクセスキーを生成します。次のステップには、アクセスキー ID とシークレットアクセスキーが必要です。

ステップ 3: データベースの権限を付与する

DBA として接続し、必要な権限をアプリケーションユーザーに付与します。

```
-- Grant EXECUTE on DBMS_CLOUD and DBMS_CLOUD_AI
GRANT EXECUTE ON DBMS_CLOUD TO db_user;
GRANT EXECUTE ON DBMS_CLOUD_AI TO db_user;
```

ステップ 4: Bedrock エンドポイントのネットワーク ACL アクセスを許可する

データベースユーザーネットワーク ACL に Bedrock ランタイムエンドポイントへのアクセスを許可します。

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'bedrock-runtime.region.amazonaws.com',
    ace  => xs$ace_type(
              privilege_list => xs$name_list('http'),
              principal_name => 'db_user',
              principal_type => xs_acl.ptype_db)
  );
END;
/
```

region を自分の AWS リージョン (例: us-east-1) に置き換え、*db_user* を Bedrock モデルを呼び出す Oracle Database ユーザーに置き換えます。

ステップ 5: データベースに認証情報を作成する

アプリケーションユーザーとして接続し、DBMS_CLOUD を使用して認証情報オブジェクトを作成します。

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'AWS_BEDROCK_CRED',
    username        => 'your_AWS_access_key_id',
    password        => 'your_AWS_secret_access_key'
  );
END;
/
```

ステップ 6: Bedrock の AI プロファイルを作成する

DBMS_CLOUD_AI を使用して、認証情報を参照し、プロバイダーとして AWS を指定する AI プロファイルを作成します。

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'BEDROCK_PROFILE',
    attributes   => '{"provider": "aws",
      "credential_name": "AWS_BEDROCK_CRED",
      "model": "us.anthropic.claude-sonnet-4-5-20250929-v1:0",
      "object_list": [{"owner": "schema", "name": "table1"},
                      {"owner": "schema", "name": "table2"}]
    }'
```

```
  );
END;
/
```

us-east-1 以外のリージョンで Amazon Bedrock ランタイムエンドポイントを使用するには、プロファイル属性 JSON に region および target_language 属性を含めます。region を、Amazon Bedrock ランタイムエンドポイントがあるリージョン (例: us-west-2) に設定します。

region を設定する場合は、target_language または source_language も含める必要があります。chat や runsql などの翻訳を使用しないアクションでも、両方の属性を一緒に設定する必要があります。

target_language 値は translate アクションにのみ影響します。region なしで target_language のみを含めると、プロファイルは引き続き bedrock-runtime.us-east-1.amazonaws.com を使用します。

> **Note**
>
> Oracle DBMS_CLOUD_AI パッケージには既知の制限があるため、target_language を region と一緒に設定する必要があります。

次の例では、us-west-2 リージョンを使用するプロファイルを作成します。

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'BEDROCK_PROFILE_USW2',
    attributes   => '{"provider": "aws",
      "credential_name": "AWS_BEDROCK_CRED",
      "region": "us-west-2",
      "target_language": "en",
      "model": "us.anthropic.claude-sonnet-4-5-20250929-v1:0",
      "object_list": [{"owner": "schema", "name": "table1"},
                      {"owner": "schema", "name": "table2"}]
    }'
  );
END;
/
```

model 属性を明示的に指定する必要があります。AWS プロバイダーにはデフォルトのモデルは提供されません。モデル ID については、以下のリファレンスを参照してください。

- ベースモデル: Amazon Bedrock ベースモデル ID。
- クロスリージョン推論プロファイル: クロスリージョン推論でサポートされているリージョンとモデル。
- プロビジョンドスループット: プロビジョンドスループットの ARN を指定します。
- カスタムモデル: プロビジョンドスループットを購入し、結果のプロビジョンドモデルの ARN を指定します。

Bedrock 統合の検証

シンプルなプロンプトで接続をテストします。

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('BEDROCK_PROFILE');
SELECT AI chat what is Amazon RDS;
```

レスポンスが正常に返される場合は、Bedrock 統合が正しく設定されています。

Bedrock 統合の使用

統合を設定したら、Select AI で SQL を使用して Oracle Database から Amazon Bedrock 基盤モデルを呼び出すことができます。

`SELECT AI` アクション
(`runsql`、`showsql`、`narrate`、`chat`、`explainsql`)、`DBMS_CLOUD_AI.GENERATE` 関数、マルチターン会話、検索拡張生成 (RAG) など、Amazon Bedrock の完全な使用例については、次の Oracle ドキュメントを参照してください。

- Example: Select AI with AWS
- Manage AI Profiles – Use AWS
- DBMS_CLOUD_AI Package Reference

Amazon RDS の Oracle にデータをインポートする

Amazon RDS for Oracle DB インスタンスにデータをインポートする方法は、次の条件によって異なります。

- 保有しているデータの量
- データベース内のデータベースオブジェクトの数
- データベース内のさまざまなデータベースオブジェクト

例えば、要件に応じて、次のツールを使用できます。

- Oracle SQL Developer — シンプルな 20 MB のデータベースを 1 つインポートできます。
- Oracle Data Pump – 複雑なデータベースや、サイズが数百メガバイトまたは数テラバイトのデータベースをインポートできます。例えば、オンプレミスのデータベースから RDS for Oracle DB インスタンスにテーブルスペースを転送できます。Amazon S3 または Amazon EFS を使用して、データファイルとメタデータを転送できます。詳細については、Oracle トランスポータブル表領域を使用した移行、Amazon EFS の統合、および 「Amazon S3 統合」 を参照してください。
- AWS Database Migration Service (AWS DMS) — ダウンタイムなしでデータベースを移行できます。AWS DMS の詳細については、「AWS Database Migration Service とは」およびブログ記事「AWS DMS を使用してほぼゼロのダウンタイムで Oracle データベースを移行する」参照してください。

> **Important**
>
> 前述の移行方法のいずれかを使用する前に、データベースをバックアップすることをお勧めします。データのインポート後に、スナップショットを作成して RDS for Oracle DB インスタンスをバックアップできます。後で、スナップショットを復元できます。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

次の表は、RDS for Oracle DB インスタンスで使用できるインポート方法を比較したものです。

Amazon RDS での Oracle のインポート方法の選択

方法	次の用途に適しています	ダウンタイムの影響	複雑さ	制限事項
「Oracle SQL Developer」	100 MB 未満のシンプルなスキーマ	高	低	大規模または複雑なデータベースには適していません
Amazon S3 経由の Data Pump	ソースへのネットワークリンクなしで 100 MB ~ 5 TB	中	中	ダンプファイルあたり最大 5 TB
データベースリンク経由の Data Pump	ソースへのネットワーク接続が可能な任意のサイズ	中	中	ソースデータベースへの VPC または VPN 接続が必要です
トランスポータブルテーブルスペース	最小限のダウンタイムで済む数 TB の移行	低	高	Enterprise Edition のみ
SQL*Loader	フラットファイルのロードと ETL ワークフロー	可変	中	手動 DDL が必要。データはフラットファイル形式である必要があります
具体化されたビュー	カットオーバー前の継続的な同期	非常に小さい	中	ソースデータベースへのデータベースリンクが必要

多くのデータベースエンジンでは、ターゲットデータベースに切り替える準備ができるまで、進行中のレプリケーションを続行できます。AWS DMS を使用して、同じデータベースエンジンまたは異なるエンジンから RDS for Oracle に移行できます。別のデータベースエンジンから移行する場合は、AWS Schema Conversion Tool を使用します。これにより、AWS DMS では移行できないスキーマオブジェクトを移行できます。

トピック

- Oracle SQL Developer を使用したインポート

- Oracle トランスポータブル表領域を使用した移行
- Oracle Data Pump を使用したインポート
- Oracle エクスポート/インポートを使用したインポート
- Oracle SQL*Loader を使用したインポート
- Oracle マテリアライズドビューを使用した移行

Oracle SQL Developer を使用したインポート

Oracle SQL Developer は、Oracle によって無償で配布されるグラフィカルな Java ツールです。SQL Developer には、2 つの Oracle データベース間でデータを移行したり、MySQL などの他のデータベースから Oracle データベースへデータを移行したりするためのオプションが用意されています。このツールは、小規模なデータベースの移行に最適です。

このツールは、デスクトップコンピュータ (Windows、Linux、Mac) またはいずれか 1 つのサーバーにインストールできます。SQL Developer をインストールすると、SQL Developer を使用して移行元のデータベースと移行先のデータベースに接続できます。[Tools] メニューの [Database Copy] コマンドを使用して、Amazon RDS インスタンスにデータをコピーします。

SQL Developer をダウンロードするには、 http://www.oracle.com/technetwork/developer-tools/sql-developer にアクセスしてください。

データの移行を開始する前に、Oracle SQL Developer 製品のドキュメントを読むことをお勧めします。Oracle には、MySQL や SQL Server などの他のデータベースから移行する方法に関するドキュメントも用意されています。詳細については、Oracle のドキュメントの http://www.oracle.com/technetwork/database/migration を参照してください。

Oracle トランスポータブル表領域を使用した移行

Oracle トランスポータブル表領域機能を使用して、オンプレミスの Oracle データベースから RDS for Oracle DB インスタンスにテーブルスペースのセットをコピーできます。物理レベルでは、Amazon EFS または Amazon S3 を使用して、ソースデータファイルとメタデータファイルをターゲット DB インスタンスに転送します。トランスポータブルテーブルスペース機能は `rdsadmin.rdsadmin_transport_util` パッケージを使用します。このプロシージャの構文とセマンティクスについては、「テーブルスペースの転送」を参照してください。

表領域をトランスポートする方法を説明したブログ記事については、「Migrate Oracle Databases to AWS using transportable tablespace」と「Amazon RDS for Oracle Transportable Tablespaces using RMAN」を参照してください。

トピック

- Oracle トランスポータブル表領域の概要
- フェーズ 1: ソースホストをセットアップする
- フェーズ 2: 表領域のフルバックアップを準備する
- フェーズ 3: 増分バックアップを作成および転送する
- フェーズ 4: 表領域をトランスポートする
- フェーズ 5: 転送された表領域を検証する
- フェーズ 6: 残ったファイルをクリーンアップする

Oracle トランスポータブル表領域の概要

トランスポータブル表領域セットは、トランスポートされる表領域セットのデータファイルと、表領域メタデータを含むエクスポートダンプファイルで構成されます。トランスポータブル表領域などの物理的な移行ソリューションでは、物理ファイル (データファイル、構成ファイル、Data Pump ダンプファイル) を転送します。

トピック

- トランスポータブル表領域のメリットとデメリット
- トランスポータブル表領域の制限事項
- トランスポータブル表領域の前提条件

トランスポータブル表領域のメリットとデメリット

ダウンタイムを最小限に抑えて 1 つ以上の大きな表領域を RDS に移行する必要がある場合は、トランスポータブル表領域を使用することをお勧めします。トランスポータブル表領域には、論理移行に比べて次のような利点があります。

- ダウンタイムは他のほとんどの Oracle 移行ソリューションよりも短いです。
- トランスポータブル表領域機能は物理ファイルのみをコピーするため、論理移行で発生する可能性のあるデータ整合性エラーや論理的な破損を回避できます。
- 追加のライセンスは必要ありません。
- 例えば、Oracle Solaris プラットフォームから Linux など、さまざまなプラットフォームやエンディアンネスタイプにわたって表領域のセットを移行できます。ただし、Windows サーバーとの間で表領域を転送したり、Windows サーバーから転送することはサポートされていません。

> **Note**
>
> Linux は完全にテスト済みでサポートされています。すべての UNIX バリエーションでテストされているわけではありません。

トランスポータブル表領域を使用する場合は、Amazon S3 または Amazon EFS を使用してデータを転送できます。

- EFS を使用する場合、バックアップはインポート中も EFS ファイルシステムに残ります。ファイルは後で削除できます。この手法では、DB インスタンスに EBS ストレージをプロビジョニングする必要はありません。このため、S3 ではなく Amazon EFS を使用することをお勧めします。詳細については、「Amazon EFS の統合」を参照してください。
- S3 を使用する場合、DB インスタンスに接続された EBS ストレージに RMAN バックアップをダウンロードします。インポート中、ファイルは EBS ストレージに残ります。インポート後にこのスペースを解放できます。このスペースは DB インスタンスに割り当てられたままです。

トランスポータブル表領域の主な欠点は、Oracle Database に関する比較的高度な知識が必要なことです。詳細は、Oracle Database 管理者ガイドの「データベース間での表領域のトランスポート」を参照してください。

トランスポータブル表領域の制限事項

RDS for Oracle でこの機能を使用する場合、Oracle データベースのトランスポータブル表領域の制限が適用されます。詳細は、Oracle Database 管理者ガイドの「トランスポータブル表領域の制限」および「データのトランスポートに関する一般的な制限」を参照してください。RDS for Oracle のトランスポータブル表領域には、さらに次の制限があることに注意してください。

- ソースデータベースとターゲットデータベースのどちらも Standard Edition 2 (SE2) を使用できません。Enterprise Edition のみがサポートされています。
- Oracle Database 11g データベースをソースとして使用することはできません。RMAN クロスプラットフォームトランスポータブルテーブルスペース機能は、Oracle Database 11g がサポートしていない RMAN トランスポートメカニズムに依存しています。
- トランスポータブル表領域を使用して RDS for Oracle DB インスタンスからデータを移行することはできません。トランスポータブル表領域は、データを RDS for Oracle DB インスタンスに移行するときのみに使用できます。

- Windows オペレーティングシステムはサポートされていません。
- 表領域を下位リリースレベルのデータベースにトランスポートすることはできません。ターゲットデータベースは、ソースデータベースと同じかそれ以降のリリースレベルでなければいけません。例えば、表領域を Oracle Database 21c から Oracle Database 19c にトランスポートすることはできません。
- SYSTEM および SYSAUX などの管理上の表領域はトランスポートできません。
- PL/SQL パッケージ、Java クラス、ビュー、トリガー、シーケンス、ユーザー、ロール、一時テーブルなどの非データオブジェクトは転送できません。データ以外のオブジェクトを転送するには、手動で作成するか、Data Pump メタデータのエクスポートとインポートを使用します。詳細については、「Oracle サポートノート 1454872.1」を参照してください。
- 暗号化された表領域や暗号化された列を使用する表領域はトランスポートできません。
- Amazon S3 を使用してファイルを転送する場合、サポートされる最大ファイルサイズは 5 TiB です。
- ソースデータベースが Spatial などの Oracle オプションを使用している場合は、ターゲットデータベースで同じオプションが設定されていない限り、表領域をトランスポートできません。
- Oracle レプリカ構成の RDS for Oracle DB インスタンスに表領域をトランスポートできません。回避策としては、すべてのレプリカを削除し、表領域をトランスポートしてから、レプリカを再作成できます。

トランスポータブル表領域の前提条件

開始する前に、以下のタスクを完了します。

- My Oracle Support の次のドキュメントに記載されているトランスポータブル表領域の要件を確認してください。
  - クロスプラットフォーム増分バックアップを使用したトランスポータブル表領域のダウンタイムの短縮 (Doc ID 2471245.1)
  - トランスポータブル表領域 (TTS) 制限事項 : 詳細、リファレンス、適用バージョン (Doc ID 1454872.1)
  - トランスポータブル表領域 (TTS) に関する主な注意事項 -- よくある質問と問題 (Doc ID 1166564.1)
- エンディアンネス変換の計画 ソースプラットフォーム ID を指定すると、RDS for Oracle はエンディアンネスを自動的に変換します。プラットフォーム ID の確認方法については、「同一 Data Guard 構成の異機種間プライマリおよびフィジカルスタンバイに関する Data Guard のサポート (Doc ID 413484.1)」を参照してください。

- ターゲット DB インスタンスでトランスポータブル表領域機能が有効になっていることを確認してください。この機能は、次のクエリを実行しても ORA-20304 エラーが表示されない場合にのみ有効になります。

```
SELECT * FROM TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);
```

  トランスポータブル表領域機能が有効になっていない場合は、DB インスタンスを再起動します。詳細については、「DB インスタンスの再起動」を参照してください。
- タイムゾーンファイルがソースとターゲットのデータベース間で同じであることを確認します。
- ソースとターゲットのデータベースの文字セットが、次のいずれかの要件を満たしていることを確認します。
  - 文字セットは同じです。
  - 文字セットに互換性があります。互換性要件のリストについては、Oracle データベースドキュメントの「General Limitations on Transporting Data」を参照してください。
- Amazon S3 を使用してファイルを転送する予定がある場合は、以下を実行してください。
  - Amazon S3 バケットをファイル転送に使用でき、この Amazon S3 バケットが、DB インスタンスと同じ AWS リージョン内にあることを確認します。手順については、Amazon Simple Storage Service 入門ガイドの「バケットの作成」を参照してください。
  - Amazon RDS 統合用の Amazon S3 バケットは、「Amazon S3 と RDS for Oracle を統合する IAM アクセス許可の設定」の手順に従って準備してください。
- Amazon EFS を使用してファイルを転送する場合は、Amazon EFS の統合 の手順に従って EFS を設定したことを確認してください。
- ターゲット DB インスタンスで自動バックアップを有効にすることを強くお勧めします。メタデータのインポートステップは失敗する可能性があるため、DB インスタンスをインポート前の状態に復元できることが重要です。それによって、表領域を再度バックアップ、転送、インポートする必要がなくなります。

フェーズ 1: ソースホストをセットアップする

このステップでは、My Oracle Support から提供されているトランスポート表領域スクリプトをコピーし、必要な構成ファイルを設定します。次のステップでは、ソースホストは、ターゲットインスタンスにトランスポートされる表領域を含むデータベースを実行しています。

ソースホストを設定するには

1. Oracle ホームのオーナーとしてソースホストにログインします。
2. ORACLE_HOME および ORACLE_SID 環境変数がソースデータベースを指していることを確認してください。
3. 管理者としてデータベースにログインし、タイムゾーンバージョン、DB 文字セット、および各国語文字セットがターゲットデータベースと同じであることを確認します。

```
SELECT * FROM V$TIMEZONE_FILE;
SELECT * FROM NLS_DATABASE_PARAMETERS
  WHERE PARAMETER IN ('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');
```

4. Oracle Support ノート 2471245.1 の説明に従って、トランスポータブル表領域ユーティリティを設定します。

   セットアップには、ソースホスト上の xtt.properties ファイルの編集が含まれます。次のサンプル xtt.properties ファイルは、/dsk1/backups ディレクトリ内の 3 つの表領域のバックアップを指定しています。これらは、ターゲット DB インスタンスにトランスポートする予定の表領域です。また、エンディアンネスを自動的に変換するソースプラットフォーム ID も指定します。

   > **Note**
   >
   > 有効なプラットフォーム ID については、「同一 Data Guard 構成の異機種間プライマリおよびフィジカルスタンバイに関する Data Guard のサポート(Doc ID 413484.1)」を参照してください。

```
#linux system
platformid=13
#list of tablespaces to transport
tablespaces=TBS1,TBS2,TBS3
#location where backup will be generated
src_scratch_location=/dsk1/backups
#RMAN command for performing backup
usermantransport=1
```

フェーズ 2: 表領域のフルバックアップを準備する

このフェーズでは、初めて表領域をバックアップし、そのバックアップをターゲットホストに転送してから、プロシージャ rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces を使用して表領域を復元します。このフェーズが完了すると、最初の表領域バックアップはターゲット DB インスタンスに保存され、増分バックアップで更新できます。

トピック

- ステップ 1: ソースホストの表領域をバックアップする
- ステップ 2: バックアップファイルをターゲット DB インスタンスに転送する
- ステップ 3: ターゲット DB インスタンスに表領域をインポートする

ステップ 1: ソースホストの表領域をバックアップする

このステップでは、xttdriver.pl スクリプトを使用して表領域のフルバックアップを作成します。xttdriver.pl の出力は、TMPDIR 環境変数に格納されます。

表領域をバックアップするには

1. 表領域が読み取り専用モードの場合は、ALTER TABLESPACE 権限を持つユーザーとしてソースデータベースにログインし、表領域を読み取り/書き込みモードにします。それ以外の場合は、次のステップに進みます。

   次の例ではtbs1、tbs2、tbs3 を読み取り/書き込みモードにします。

   ```
   ALTER TABLESPACE tbs1 READ WRITE;
   ALTER TABLESPACE tbs2 READ WRITE;
   ALTER TABLESPACE tbs3 READ WRITE;
   ```

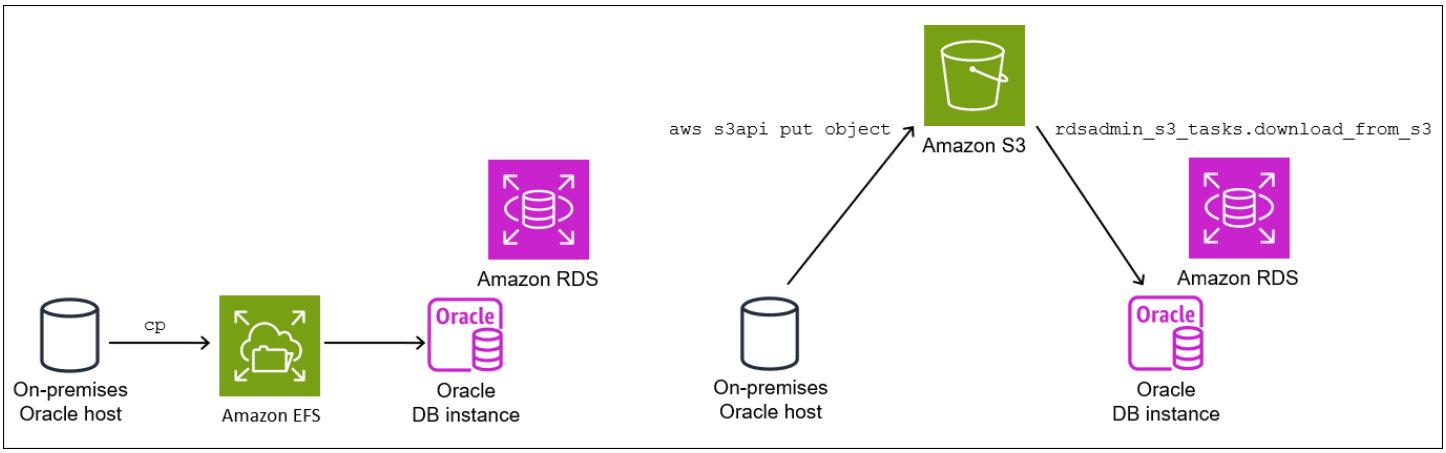
2. xttdriver.pl スクリプトを使用して表領域をバックアップします。オプションで、--debug を指定してデバッグモードでスクリプトを実行できます。

   ```
   export TMPDIR=location_of_log_files
   cd location_of_xttdriver.pl
   $ORACLE_HOME/perl/bin/perl xttdriver.pl --backup
   ```

ステップ 2: バックアップファイルをターゲット DB インスタンスに転送する

このステップでは、バックアップファイルと設定ファイルをスクラッチの場所からターゲット DB インスタンスにコピーします。以下のオプションのいずれかを選択してください。

- ソースホストとターゲットホストが Amazon EFS ファイルシステムを共有する場合は、cp などのオペレーティングシステムユーティリティを使用して、バックアップファイルと `res.txt` ファイルをスクラッチの場所から共有ディレクトリにコピーします。その後、ステップ 3: ターゲット DB インスタンスに表領域をインポートするに進みます。
- バックアップを Amazon S3 バケットにステージングする必要がある場合は、以下の手順を実行してください。



ステップ 2.2: Amazon S3 バケットにダンプファイルをアップロードする

バックアップと `res.txt` ファイルをスクラッチディレクトリから Amazon S3 バケットにアップロードします。詳細については、Amazon Simple Storage Service 開発者ガイドの「オブジェクトのアップロード」を参照してください。

ステップ 2.3: バックアップを Amazon S3 バケットからターゲット DB インスタンスにダウンロードする

このステップでは、プロシージャ `rdsadmin.rdsadmin_s3_tasks.download_from_s3` を使用して RDS for Oracle DB インスタンスにバックアップをダウンロードします。

Amazon S3 バケットからバックアップをダウンロードするには

1. SQL*Plus または Oracle SQL Developer を起動し、RDS for Oracle DB インスタンスにログインします。

2. Amazon S3 バケットからターゲット DB インスタンスにバックアップをダウンロードします。これには、Amazon S3 バケットからターゲット DB インスタンスにファイルをダウンロードするための Amazon RDS プロシージャ `rdsadmin.rdsadmin_s3_tasks.download_from_s3` を使用します。次の例では、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットから *DATA_PUMP_DIR* ディレクトリにファイルをすべてダウンロードします。

```
EXEC UTL_FILE.FREMOVE ('DATA_PUMP_DIR', 'res.txt');
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
  p_bucket_name    =>  'amzn-s3-demo-bucket',
  p_directory_name =>  'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

SELECT ステートメントでは、データ型 VARCHAR2 のタスクの ID が返ります。詳細については、「Amazon S3 バケットから Oracle DB インスタンスにファイルをダウンロードする」を参照してください。

ステップ 3: ターゲット DB インスタンスに表領域をインポートする

プロシージャ `rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` を使用して、表領域をターゲット DB インスタンスに復元します。このプロシージャは、データファイルを正しいエンディアン形式に自動的に変換します。

Linux 以外のプラットフォームからインポートする場合は、`import_xtts_tablespaces` を呼び出すときに `p_platform_id` パラメータを使用してソースプラットフォームを指定します。指定するプラットフォーム ID が、ステップ 2: ソースホストに表領域メタデータをエクスポートする の `xtt.properties` ファイルで指定されたものと一致していることを確認してください。

ターゲット DB インスタンスに表領域をインポートする

1. Oracle SQL クライアントを起動し、ターゲット RDS for Oracle DB インスタンスにマスターユーザーとしてログインします。
2. インポートする表領域とバックアップを含むディレクトリを指定して、`rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` プロシージャを実行します。

   以下の例では、表領域*TBS1*、*TBS2*、および *TBS3* をディレクトリ *DATA_PUMP_DIR* からインポートします。ソースプラットフォームは AIX ベースのシステム (64 ビット) で、プラット

フォーム ID は 6 です。プラットフォーム ID は、V$TRANSPORTABLE_PLATFORM をクエリすることで確認できます。

```
VAR task_id CLOB

BEGIN
  :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces(
        'TBS1,TBS2,TBS3',
        'DATA_PUMP_DIR',
        p_platform_id => 6);
END;
/

PRINT task_id
```

3. (オプション) テーブル rdsadmin.rds_xtts_operation_info にクエリを実行して進行状況を監視します。xtts_operation_state 列には、EXECUTING、COMPLETED、または FAILED という値が表示されます。

```
SELECT * FROM rdsadmin.rds_xtts_operation_info;
```

> **Note**
>
> 実行時間が長い操作の場合は、V$SESSION_LONGOPS、V$RMAN_STATUS、V$RMAN_OUTPUT にクエリを実行することもできます。

4. 前のステップのタスク ID を使用して、完了したインポートのログを表示します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||'&task_id'||'.log'));
```

インポートが正常に完了したことを確認してから、次の手順に進みます。

フェーズ 3: 増分バックアップを作成および転送する

このフェーズでは、ソースデータベースがアクティブな間、定期的に増分バックアップを作成して転送します。この手法を実行することで、最終的な表領域バックアップのサイズが小さくなります。複数の増分バックアップを行う場合、最後の増分バックアップの後に res.txt ファイルをコピーしてから、ターゲットインスタンスに適用する必要があります。

手順は フェーズ 2: 表領域のフルバックアップを準備する と同じですが、インポートの手順が任意である点が異なります。

フェーズ 4: 表領域をトランスポートする

このフェーズでは、読み取り専用の表領域をバックアップし、Data Pump メタデータをエクスポートし、これらのファイルをターゲットホストに転送して、表領域とメタデータの両方をインポートします。

トピック

- ステップ 1: 読み取り専用の表領域をバックアップする
- ステップ 2: ソースホストに表領域メタデータをエクスポートする
- ステップ 3: (Amazon S3 のみ) バックアップファイルとエクスポートファイルをターゲット DB インスタンスに転送する
- ステップ 4: ターゲット DB インスタンスに表領域をインポートする
- ステップ 5: ターゲット DB インスタンスに表領域メタデータをインポートする

ステップ 1: 読み取り専用の表領域をバックアップする

このステップは ステップ 1: ソースホストの表領域をバックアップする と同じですが、重要な違いが 1 つあります。最後に表領域をバックアップする前に、表領域を読み取り専用モードにします。

次の例では、tbs1、tbs2、tbs3 を読み取り専用モードにします。

> **Important**
>
> テーブルスペースを読み取り専用モードに設定すると、移行のためのダウンタイムウィンドウが開始されます。この時点から、アプリケーションはソースデータベース上のこれらのテーブルスペースに書き込むことができなくなります。このステップはメンテナンスウィンドウ中に計画します。

```
ALTER TABLESPACE tbs1 READ ONLY;
ALTER TABLESPACE tbs2 READ ONLY;
ALTER TABLESPACE tbs3 READ ONLY;
```

ステップ 2: ソースホストに表領域メタデータをエクスポートする

ソースホストで expdp ユーティリティを実行して、表領域メタデータをエクスポートします。次の例では、表領域 *TBS1*、*TBS2*、および *TBS3* を *DATA_PUMP_DIR* ディレクトリのダンプファイル *xttdump.dmp* にエクスポートします。

```
expdp username/pwd \
dumpfile=xttdump.dmp \
directory=DATA_PUMP_DIR \
statistics=NONE \
transport_tablespaces=TBS1,TBS2,TBS3 \
transport_full_check=y \
logfile=tts_export.log
```

DATA_PUMP_DIR が Amazon EFS の共有ディレクトリである場合は、スキップして ステップ 4: ターゲット DB インスタンスに表領域をインポートする に進んでください。

ステップ 3: (Amazon S3 のみ) バックアップファイルとエクスポートファイルをターゲット DB インスタンスに転送する

Amazon S3 を使用して表領域バックアップと Data Pump エクスポートファイルをステージングする場合は、次の手順を実行してください。

ステップ 3.1: バックアップとダンプファイルをソースホストから Amazon S3 バケットにアップロードする

バックアップとダンプファイルをソースホストから Amazon S3 バケットにアップロードします。詳細については、Amazon Simple Storage Service 開発者ガイドの「オブジェクトのアップロード」を参照してください。

ステップ 3.2: バックアップとダンプファイルを Amazon S3 バケットからターゲット DB インスタンスにダウンロードする

このステップでは、プロシージャ `rdsadmin.rdsadmin_s3_tasks.download_from_s3` を使用して RDS for Oracle DB インスタンスにバックアップとダンプファイルをダウンロードします。「ステップ 2.3: バックアップを Amazon S3 バケットからターゲット DB インスタンスにダウンロードする」のステップを実行してください。

ステップ 4: ターゲット DB インスタンスに表領域をインポートする

`rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` プロシージャを使用して表領域を復元します。このプロシージャの構文とセマンティクスについては、「転送されたテーブルスペースを DB インスタンスにインポートする」を参照してください。

> **Important**
>
> 最後の表領域のインポートが完了したら、次のステップは Oracle Data Pump メタデータのインポートです。インポートが失敗した場合、DB インスタンスを失敗前の状態に戻すことが重要です。そのため、Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成 の手順に従って DB インスタンスの DB スナップショットを作成することをお勧めします。スナップショットにはインポートされたすべての表領域が含まれるため、インポートが失敗した場合でも、バックアップとインポートのプロセスを繰り返す必要はありません。
>
> ターゲット DB インスタンスで自動バックアップがオンになっていて、メタデータをインポートする前に有効なスナップショットが開始されたことを Amazon RDS が検出しない場合、RDS はスナップショットの作成を試みます。インスタンスのアクティビティに応じて、このスナップショットは成功する場合と成功しない場合があります。有効なスナップショットが検出されないか、スナップショットを開始できない場合、メタデータのインポートはエラーを出して終了します。

ターゲット DB インスタンスに表領域をインポートする

1. Oracle SQL クライアントを起動し、ターゲット RDS for Oracle DB インスタンスにマスターユーザーとしてログインします。
2. インポートする表領域とバックアップを含むディレクトリを指定して、`rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces` プロシージャを実行します。

   以下の例では、表領域*TBS1*、*TBS2*、および *TBS3* をディレクトリ *DATA_PUMP_DIR* からインポートします。

```
BEGIN

 :task_id:=rdsadmin.rdsadmin_transport_util.import_xtts_tablespaces('TBS1,TBS2,TBS3','DATA_
END;
/
```

```
PRINT task_id
```

3. (オプション) テーブル rdsadmin.rds_xtts_operation_info にクエリを実行して進行状況を監視します。xtts_operation_state 列には、EXECUTING、COMPLETED、または FAILED という値が表示されます。

```
SELECT * FROM rdsadmin.rds_xtts_operation_info;
```

> **Note**
>
> 実行時間が長い操作の場合は、V$SESSION_LONGOPS、V$RMAN_STATUS、V$RMAN_OUTPUT にクエリを実行することもできます。

4. 前のステップのタスク ID を使用して、完了したインポートのログを表示します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP',
 'dbtask-'||'&task_id'||'.log'));
```

インポートが正常に完了したことを確認してから、次の手順に進みます。

5. Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成 の指示に従って DB スナップショットを手動で作成します。

ステップ 5: ターゲット DB インスタンスに表領域メタデータをインポートする

このステップでは、rdsadmin.rdsadmin_transport_util.import_xtts_metadata プロシージャを使用して RDS for Oracle DB インスタンスにトランスポータブル表領域メタデータをインポートします。このプロシージャの構文とセマンティクスについては、「転送可能テーブルスペースメタデータを DB インスタンスにインポートする」を参照してください。操作中、インポートのステータスがテーブル rdsadmin.rds_xtts_operation_info に表示されます。

> **Important**
>
> メタデータをインポートする前に、表領域をインポートした後に DB スナップショットが正常に作成されたことを確認することを強くお勧めします。インポートステップが失敗した場合は、DB インスタンスを復元し、インポートエラーに対処してから、インポートを再試行してください。

データポンプメタデータを RDS for Oracle DB インスタンスにインポートします

1. Oracle SQL クライアントを起動し、ターゲット DB インスタンスにマスターユーザーとしてログインします。
2. トランスポートされた表領域にスキーマを所有するユーザーがまだ存在しない場合は、それらのユーザーを作成します。

```
CREATE USER tbs_owner IDENTIFIED BY password;
```

3. ダンプファイルの名前とディレクトリの場所を指定して、メタデータをインポートします。

```
BEGIN

 rdsadmin.rdsadmin_transport_util.import_xtts_metadata('xttdump.dmp','DATA_PUMP_DIR');
END;
/
```

4. (オプション) トランスポータブル表領域の履歴テーブルにクエリを実行して、メタデータインポートのステータスを確認します。

```
SELECT * FROM rdsadmin.rds_xtts_operation_info;
```

このオペレーションが完了すると、表領域は読み取り専用モードになります。

5. (オプション) ログファイルを表示します。

次の例では、BDUMP ディレクトリの内容を一覧表示し、インポートログをクエリします。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'BDUMP'));

SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file(
  p_directory => 'BDUMP',
  p_filename => 'rds-xtts-
import_xtts_metadata-2023-05-22.01-52-35.560858000.log'));
```

フェーズ 5: 転送された表領域を検証する

このオプションのステップでは、rdsadmin.rdsadmin_rman_util.validate_tablespace プロシージャを使用してトランスポートされた表領域を検証し、表領域を読み取り/書き込みモードにします。

転送されたデータを検証するには

1. SQL*Plus または SQL Developer を起動し、ターゲット DB インスタンスにマスターユーザーとしてログインします。
2. `rdsadmin.rdsadmin_rman_util.validate_tablespace` プロシージャを使用して表領域を検証します。

```
SET SERVEROUTPUT ON
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS1',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS2',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
    rdsadmin.rdsadmin_rman_util.validate_tablespace(
        p_tablespace_name     => 'TBS3',
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_rman_to_dbms_output => TRUE);
END;
/
```

3. 表領域を読み取り/書き込みモードにします。

```
ALTER TABLESPACE TBS1 READ WRITE;
ALTER TABLESPACE TBS2 READ WRITE;
ALTER TABLESPACE TBS3 READ WRITE;
```

フェーズ 6: 残ったファイルをクリーンアップする

このオプションのステップでは、不要なファイルをすべて削除します。`rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files` プロシージャを使用して、テーブルスペースのインポート後に孤立したデータファイルを一覧表示して、`rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import` プロシージャを使用して削除します。これらのプロシージャの構文とセマンティクスについては、「テーブルスペースのインポート後の孤立ファイルを一覧表示する」および「テーブルスペースのインポート後に孤立したデータファイルを削除する」を参照してください。

残ったファイルをクリーンアップするには

1. 次のように *DATA_PUMP_DIR* の古いバックアップを削除します。

   a. rdsadmin.rdsadmin_file_util.listdir を実行してバックアップファイルを一覧表示します。

   ```
   SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory =>
    'DATA_PUMP_DIR'));
   ```

   b. UTL_FILE.FREMOVE を呼び出して、バックアップを 1 つずつ削除します。

   ```
   EXEC UTL_FILE.FREMOVE ('DATA_PUMP_DIR', 'backup_filename');
   ```

2. 表領域をインポートしたが、これらの表領域のメタデータをインポートしなかった場合は、孤立したデータファイルを次のように削除できます。

   a. 削除する必要のある孤立したデータファイルを一覧表示します。次の例では、rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files プロシージャを呼び出します。

   ```
   SQL> SELECT * FROM
    TABLE(rdsadmin.rdsadmin_transport_util.list_xtts_orphan_files);

   FILENAME       FILESIZE
   -------------- ---------
   datafile_7.dbf 104865792
   datafile_8.dbf 104865792
   ```

   b. rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import プロシージャを実行して、孤立したファイルを削除します。

   ```
   BEGIN

    rdsadmin.rdsadmin_transport_util.cleanup_incomplete_xtts_import('DATA_PUMP_DIR');
   END;
   /
   ```

   クリーンアップ操作により、BDUMP ディレクトリに名前形式 rds-xtts-delete_xtts_orphaned_files-*YYYY-MM-DD.HH24-MI-SS.FF*.log を使用するログファイルが生成されます。

c. 前のステップで生成されたログファイルを読み込みます。次の例ではログ rds-xtts-delete_xtts_orphaned_files-*2023-06-01.09-33-11.868894000*.log を読み取ります。

```
SELECT *
FROM TABLE(rdsadmin.rds_file_util.read_text_file(
       p_directory => 'BDUMP',
       p_filename  => 'rds-xtts-
delete_xtts_orphaned_files-2023-06-01.09-33-11.868894000.log'));

TEXT
--------------------------------------------------------------------------------
orphan transported datafile datafile_7.dbf deleted.
orphan transported datafile datafile_8.dbf deleted.
```

3. 表領域をインポートし、これらの表領域のメタデータをインポートしたものの、互換性エラーやその他の Oracle Data Pump の問題が発生した場合は、次のように部分的にトランスポートされたデータファイルをクリーンアップします。

a. DBA_TABLESPACES クエリを実行して、部分的にトランスポートされたデータファイルを含む表領域を一覧表示します。

```
SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE PLUGGED_IN='YES';

TABLESPACE_NAME
--------------------------------------------------------------------------------
TBS_3
```

b. 表領域と部分的にトランスポートされたデータファイルを削除します。

```
DROP TABLESPACE TBS_3 INCLUDING CONTENTS AND DATAFILES;
```

Oracle Data Pump を使用したインポート

Oracle Data Pump は、Oracle データのダンプファイルへのエクスポートおよび別の Oracle データベースへのインポートを行うことができるユーティリティです。Oracle Data Pump は、Oracle エクスポート/インポートユーティリティとして長期間使用されてきました。Oracle Data Pump は、Oracle データベースから Amazon RDS DB インスタンスに大量のデータを移行する際に推奨される方法でもあります。

このセクションの例では、Oracle データベースにデータをインポートする 1 つの方法を示していますが、Oracle Data Pump では他の手法もサポートしています。詳細については、Oracle Database のドキュメントを参照してください。

このセクションの例では DBMS_DATAPUMP パッケージを使用します。同じタスクは、Oracle Data Pump コマンドラインユーティリティの impdp および expdp を使用して実行できます。これらのユーティリティは、Oracle インスタントクライアントを含む Oracle クライアントインストールの一部としてリモートホストにインストールできます。詳細については、「Oracle Instant Client を使用した Amazon RDS for Oracle DB インスタンスの Data Pumpのインポートまたはエクスポートを実行する方法」を参照してください。

トピック

- Oracle Data Pump の概要
- Oracle Data Pump と Amazon S3 バケットを使用したデータのインポート
- Oracle Data Pump とデータベースリンクを使用したデータのインポート

Oracle Data Pump の概要

Oracle Data Pump は、次のコンポーネントで構成されています。

- コマンドラインクライアントの expdp および impdp
- DBMS_DATAPUMP PL/SQL パッケージ
- DBMS_METADATA PL/SQL パッケージ

以下のシナリオで Oracle Data Pump を使用できます。

- Oracle データベース (オンプレミスまたは Amazon EC2 インスタンス) から RDS for Oracle DB インスタンスにデータをインポートする。
- RDS for Oracle DB インスタンスから Oracle データベース (オンプレミスまたは Amazon EC2 インスタンス) にデータをインポートする。
- RDS for Oracle DB インスタンス間でデータをインポートする (例: EC2-Classic から VPC へのデータ移行)。

Oracle Data Pump ユーティリティをダウンロードするには、Oracle Technical Network ウェブサイトの「Oracle Database ソフトウェア・ダウンロード」を参照してください。Oracle Database の

バージョン間で移行する場合の互換性に関する考慮事項については、Oracle Database のドキュメントを参照してください。

Oracle Data Pump のワークフロー

通常、Oracle Data Pump は、以下の段階を踏んで使用します。

1. ソースデータベースのダンプファイルにデータをエクスポートします。
2. ダンプファイルを、ターゲットの RDS for Oracle DB インスタンスにアップロードします。ダンプファイルを転送するには、Amazon S3 バケットを使用するか、2 つのデータベース間のデータベースリンクを使用します。
3. ダンプファイルから RDS for Oracle DB インスタンスにデータをインポートします。

Oracle Data Pump のベストプラクティス

Oracle Data Pump を使用して RDS for Oracle インスタンスにデータをインポートする場合は、次のベストプラクティスをお勧めします。

- 特定のスキーマやオブジェクトをインポートするには、`schema` または `table` モードでインポートを実行します。
- インポートするスキーマをアプリケーションに必要なスキーマに制限します。
- `full` モードでのインポートまたはシステムが管理するコンポーネントのスキーマのインポートは行わないでください。

  RDS for Oracle では SYS または SYSDBA 管理ユーザーへのアクセスが許可されていないため、これらのアクションによって Oracle データディレクトリが損傷し、データベースの安定性が影響を受ける可能性があります。
- 大量のデータをロードする場合は、以下の操作を実行します。
  1. ダンプファイルを、ターゲットの RDS for Oracle DB インスタンスに転送します。
  2. インスタンスのDB スナップショットを取得します。
  3. インポートをテストして、これが成功することを確認します。

  データベースコンポーネントが無効の場合は、DB インスタンスを削除後、DB スナップショットから再作成します。復元された DB インスタンスには、DB スナップショットの作成時に DB インスタンス上でステージングされたダンプファイルがすべて含まれています。
- Oracle Data Pump エクスポートパラメータ `TRANSPORT_TABLESPACES`、`TRANSPORTABLE`、または `TRANSPORT_FULL_CHECK` を使用して作成されたダンプファイルはインポートしないでくだ

さい。RDS for Oracle DB インスタンスでは、これらのダンプファイルのインポートはサポートされていません。

- SYS、SYSTEM、RDSADMIN、RDSSEC、RDS_DATAGUARD の Oracle スケジューラオブジェクトを含み、以下のカテゴリに属するダンプファイルをインポートしないでください。
  - ジョブ
  - プログラム
  - スケジュール
  - チェーン
  - Rules
  - 評価コンテキスト
  - ルールセット

  RDS for Oracle DB インスタンスでは、これらのダンプファイルのインポートはサポートされていません。
- サポートされていない Oracle Scheduler オブジェクトを除外するには、Data Pump エクスポート時に追加のディレクティブを使用します。DBMS_DATAPUMP を使用する場合は、METADATA_FILTER の前に DBMS_DATAPUMP.START_JOB をさらに追加します。

```
DBMS_DATAPUMP.METADATA_FILTER(
  v_hdnl,
  'EXCLUDE_NAME_EXPR',
  q'[IN (SELECT NAME FROM SYS.OBJ$
         WHERE TYPE# IN (66,67,74,79,59,62,46)
         AND OWNER# IN
           (SELECT USER# FROM SYS.USER$
            WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
            )
        )
  ]',
  'PROCOBJ'
);
```

expdp を使用する場合は、次の例に示す exclude ディレクティブを含むパラメータファイルを作成します。その後、PARFILE=*parameter_file* コマンドで expdp を使用します。

```
exclude=procobj:"IN
  (SELECT NAME FROM sys.OBJ$
```

```
  WHERE TYPE# IN (66,67,74,79,59,62,46)
  AND OWNER# IN
    (SELECT USER# FROM SYS.USER$
     WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
    )
)"
```

Oracle Data Pump と Amazon S3 バケットを使用したデータのインポート

次のインポートプロセスでは、Oracle Data Pump と Amazon S3 バケットを使用します。ステップは次のとおりです。

1. Oracle DBMS_DATAPUMP パッケージを使用して、ソースデータベースのデータをエクスポートします。
2. ダンプファイルを Amazon S3 バケットに配置します。
3. Amazon S3 バケットから、ターゲットの Amazon RDS for Oracle DB インスタンスの DATA_PUMP_DIR ディレクトリにダンプファイルをダウンロードします。
4. DBMS_DATAPUMP パッケージを使用して、コピーしたダンプファイルのデータを RDS for Oracle DB インスタンス内にインポートします。

トピック

- Oracle Data Pump と Amazon S3 バケットを使用したデータのインポートの要件
- ステップ 1: RDS for Oracle のターゲットの DB インスタンスのデータベースユーザーに特権を付与する
- ステップ 2: DBMS_DATAPUMP を使用してデータをダンプファイルにエクスポートする
- ステップ 3: Amazon S3 バケットにダンプファイルをアップロードする
- ステップ 4: ダンプファイルを Amazon S3 バケットからターゲット DB インスタンスにダウンロードする
- ステップ 5: DBMS_DATAPUMP を使用してダンプファイルをターゲット DB インスタンスにインポートする
- ステップ 6: クリーンアップ

Oracle Data Pump と Amazon S3 バケットを使用したデータのインポートの要件

このプロセスには、次の要件があります。

- Amazon S3 バケットをファイル転送に使用でき、この Amazon S3 バケットが、DB インスタンスと同じ AWS リージョン リージョン内にあることを確認します。手順については、Amazon Simple Storage Service 入門ガイドの「バケットの作成」を参照してください。
- Amazon S3 バケットにアップロードするオブジェクトは、5 TB 以下にする必要があります。Amazon S3 でオブジェクトを操作する方法については、Amazon Simple Storage Service ユーザーガイドを参照してください。

> **Note**
>
> ダンプファイルが5 TBを超える場合、並列オプションを使用して Oracle Data Pump エクスポートを実行できます。このオペレーションは、個々のファイルの 5 TB の制限を超えないように複数のダンプファイルにデータを分散します。

- Amazon RDS 統合用の Amazon S3 バケットは、「Amazon S3 と RDS for Oracle を統合する IAM アクセス許可の設定」の手順に従って準備してください。
- 移行元のインスタンスと移行先の DB インスタンスにダンプファイルを保存するための十分なストレージ領域が必要です。

> **Note**
>
> このプロセスでは、`DATA_PUMP_DIR` ディレクトリ (すべての Oracle DB インスタンスで事前に設定されているディレクトリ) にダンプファイルをインポートします。このディレクトリはデータファイルと同じストレージボリュームにあります。ダンプファイルをインポートした場合、既存の Oracle データファイルのスペース占有率は高くなります。そのため、DB インスタンスではスペースの追加占有に対応できることを確認する必要があります。インポートしたダンプファイルは、`DATA_PUMP_DIR` ディレクトリから自動的に削除またはパージされることはありません。インポートしたダンプファイルを削除するには、Oracle ウェブサイトにある UTL_FILE.FREMOVE を使用します。

ステップ 1: RDS for Oracle のターゲットの DB インスタンスのデータベースユーザーに特権を付与する

このステップでは、データのインポート先となるスキーマを作成し、ユーザーに必要な特権を付与します。

RDS for Oracle ターゲットインスタンスでユーザーを作成し、必要な特権を付与するには

1. SQL*Plus や Oracle SQL Developer を使用して、データをインポートする先の RDS for Oracle DB インスタンスにマスターユーザーとしてログインします。DB インスタンスへの接続方法については、「Oracle DB インスタンスへの接続」を参照ください。
2. データをインポートする前に、必要なテーブルスペースを作成します。詳細については、「RDS for Oracle でのテーブルスペースの作成とサイズ設定」を参照してください。
3. データのインポート先のユーザーアカウントが存在しない場合は、ユーザーアカウントを作成し、必要なアクセス許可とロールを付与します。データを複数のユーザースキーマにインポートする場合は、各ユーザーアカウントを作成し、それぞれ必要な権限およびロールを付与します。

   例えば、以下の SQL ステートメントでは、新しいユーザーを作成して、ユーザーが所有するスキーマ内にデータをインポートするために必要な特権とロールを付与します。*schema_1* を、このステップおよび次のステップのスキーマ名に置き換えます。

   ```
   CREATE USER schema_1 IDENTIFIED BY my_password;
   GRANT CREATE SESSION, RESOURCE TO schema_1;
   ALTER USER schema_1 QUOTA 100M ON users;
   ```

   > **Note**
   >
   > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

   前のステートメントでは、新規ユーザーに特権 CREATE SESSION とロール RESOURCE を付与します。インポートするデータベースオブジェクトによっては、特権とロールの追加が必要になる場合があります。

ステップ 2: DBMS_DATAPUMP を使用してデータをダンプファイルにエクスポートする

ダンプファイルを作成するには、DBMS_DATAPUMP パッケージを使用します。

Oracle データをダンプファイルにエクスポートするには

1. 管理ユーザーとして、SQL Plus または Oracle SQL Developer を使用してソースの RDS for Oracle DB インスタンスに接続します。移行元のデータベースが RDS for Oracle DB インスタンスである場合は、Amazon RDS マスターユーザーとして接続します。

2. DBMS_DATAPUMP プロシージャを呼び出して、データをエクスポートします。

   次のスクリプトでは、DATA_PUMP_DIR ディレクトリ内の sample.dmp という名前のダンプファイルに *SCHEMA_1* スキーマをエクスポートします。*SCHEMA_1* をエクスポートするスキーマの名前に置き換えます。

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'EXPORT',
    job_mode  => 'SCHEMA',
    job_name  => null
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl         ,
    filename  => 'sample.dmp'   ,
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file
  );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_exp.log',
    directory => 'DATA_PUMP_DIR' ,
    filetype  => dbms_datapump.ku$_file_type_log_file
  );
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl,
    'EXCLUDE_NAME_EXPR',
    q'[IN (SELECT NAME FROM SYS.OBJ$
           WHERE TYPE# IN (66,67,74,79,59,62,46)
           AND OWNER# IN
             (SELECT USER# FROM SYS.USER$
              WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
             )
          )
    ]',
    'PROCOBJ'
  );
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
```

```
/
```

> **Note**
>
> Data Pump は非同期的にジョブを開始します。Data Pump ジョブのモニタリングについては、Oracle ドキュメントの「ジョブステータスのモニタリング」を参照してください。

3. (オプション) `rdsadmin.rds_file_util.read_text_file` プロシージャを呼び出してエクスポートログの内容を表示します。詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

ステップ 3: Amazon S3 バケットにダンプファイルをアップロードする

Amazon RDS プロシージャ `rdsadmin.rdsadmin_s3_tasks.upload_to_s3` を使用して、Amazon S3 バケットにダンプファイルをコピーします。次の例では、`DATA_PUMP_DIR` ディレクトリのすべてのファイルを、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットにアップロードします。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
  p_bucket_name    =>  'amzn-s3-demo-bucket',
  p_directory_name =>  'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

SELECT ステートメントでは、データ型 VARCHAR2 のタスクの ID が返ります。詳細については、「RDS for Oracle DB インスタンスから Amazon S3 バケットにファイルをアップロードする」を参照してください。

ステップ 4: ダンプファイルを Amazon S3 バケットからターゲット DB インスタンスにダウンロードする

Amazon RDS プロシージャ `rdsadmin.rdsadmin_s3_tasks.download_from_s3` を使用して、このステップを実行します。ファイルをディレクトリにダウンロードするとき、ディレクトリに同じ名前のファイルが既に存在する場合、プロシージャ `download_from_s3` はダウンロードをスキップします。ダウンロードディレクトリからファイルを削除するには、Oracle ウェブサイトにある UTL_FILE.FREMOVE を使用します。

ダンプファイルをダウンロードするには

1. SQL*Plus または Oracle SQL Developer を起動し、Amazon RDS ターゲットの Oracle DB インスタンスにマスターとしてログインします。
2. Amazon RDS プロシージャ `rdsadmin.rdsadmin_s3_tasks.download_from_s3` を使用してダンプファイルをダウンロードします。

   次の例では、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットからディレクトリ DATA_PUMP_DIR にすべてのファイルをダウンロードします。

   ```
   SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
     p_bucket_name    =>  'amzn-s3-demo-bucket',
     p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
   ```

   SELECT ステートメントでは、データ型 VARCHAR2 のタスクの ID が返ります。詳細については、「Amazon S3 バケットから Oracle DB インスタンスにファイルをダウンロードする」を参照してください。

ステップ 5: DBMS_DATAPUMP を使用してダンプファイルをターゲット DB インスタンスにインポートする

DBMS_DATAPUMP を使用して、RDS for Oracle DB インスタンスにスキーマをインポートします。METADATA_REMAP などの追加オプションが必要になる場合があります。

ターゲット DB インスタンスにデータをインポートするには

1. SQL*Plus または SQL Developer を起動し、RDS for Oracle DB インスタンスにマスターユーザーとしてログインします。
2. DBMS_DATAPUMP プロシージャを呼び出して、データをインポートします。

   次の例では、sample.dmp からターゲット DB インスタンスに *SCHEMA_1* データをインポートします。

   ```
   DECLARE
     v_hdnl NUMBER;
   BEGIN
     v_hdnl := DBMS_DATAPUMP.OPEN(
       operation => 'IMPORT',
   ```

```
    job_mode  => 'SCHEMA',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump ジョブは非同期的に開始されます。Data Pump ジョブのモニタリングについては、Oracle ドキュメントの「ジョブステータスのモニタリング」を参照してください。インポートログの内容は、rdsadmin.rds_file_util.read_text_file の手順を使用して表示できます。詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

Data Pump インポートジョブの進行状況をモニタリングするには、次のクエリを実行します。

```
SELECT job_name, operation, job_mode, state, attached_sessions FROM
 DBA_DATAPUMP_JOBS WHERE owner_name = USER;
```

3. ターゲット DB インスタンスのスキーマテーブルを一覧表示して、データのインポートを検証します。

   例えば、次のクエリでは、*SCHEMA_1* のテーブル数が返ります。

```
SELECT COUNT(*) FROM DBA_TABLES WHERE OWNER='SCHEMA_1';
```

Data Pump のインポートのトラブルシューティング

Data Pump インポートジョブのステータスを確認するには、インポートログファイルをクエリします。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('DATA_PUMP_DIR',
 'sample_imp.log'));
```

一般的なエラーは次のとおりです。

- ORA-39083: オブジェクトタイプの作成に失敗しました。通常、権限またはテーブルスペースの不足が原因で発生します。必要な権限を付与するか、METADATA_REMAP を使用してテーブルスペースを再マッピングします。
- ORA-39166: オブジェクトが見つかりません。ダンプファイルにソーススキーマまたはオブジェクトが存在しません。
- ORA-31693: テーブルデータオブジェクトのロードに失敗しました。テーブルスペースのクォータまたはスペースの問題を確認します。
- ORA-31634: ジョブは既に存在します。多くの場合、以前のジョブによって孤立したマスターテーブルが残されたことが原因で、同じ名前の Data Pump ジョブが既に存在します。DBA_DATAPUMP_JOBS をクエリして NOT RUNNING 状態のジョブを検索し、残存しているマスターテーブル (DROP TABLE *job_name*; など) を削除するか、DBMS_DATAPUMP.OPEN 呼び出しで一意の job_name を指定します。

部分的に失敗したインポートを再実行するには、インポートパラメータに TABLE_EXISTS_ACTION => 'REPLACE' を追加して既存のオブジェクトを上書きします。

ステップ 6: クリーンアップ

データをインポートしたら、保管が不要になったファイルは削除できます。

不要なファイルを削除するには

1. SQL*Plus または SQL Developer を起動し、RDS for Oracle DB インスタンスにマスターユーザーとしてログインします。
2. 次のコマンドを使用して DATA_PUMP_DIR のファイルを一覧表示します。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR')) ORDER BY
 MTIME;
```

3. DATA_PUMP_DIR 内の不要になったファイルを削除するには、次のコマンドを使用します。

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','filename');
```

例えば、次のコマンドは、sample.dmp という名前のファイルが削除されます。

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample.dmp');
```

Oracle Data Pump とデータベースリンクを使用したデータのインポート

次のインポートプロセスでは、Oracle Data Pump と Oracle DBMS_FILE_TRANSFER パッケージを使用します。ステップは次のとおりです。

1. ソースの Oracle データベース (オンプレミスデータベース、Amazon EC2 インスタンス、または RDS for Oracle DB インスタンス) に接続します。
2. DBMS_DATAPUMP を使用してデータをエクスポートします。
3. DBMS_FILE_TRANSFER.PUT_FILE を使用して、Oracle インスタンスのダンプファイルを、データベースリンクを使用して接続されているターゲットの RDS for Oracle DB インスタンスの DATA_PUMP_DIR ディレクトリにコピーします。
4. DBMS_DATAPUMP パッケージを使用して、コピーしたダンプファイルのデータを RDS for Oracle DB インスタンス内にインポートします。

Oracle Data Pump と DBMS_FILE_TRANSFER バケットを使用したインポートプロセスでは、次のステップを使用します。

トピック

- Oracle Data Pump とデータベースリンクを使用したデータのインポートの要件
- ステップ 1: RDS for Oracle のターゲットの DB インスタンスのユーザーに特権を付与する
- ステップ 2: ソースデータベースのユーザーに特権を付与する
- ステップ 3: DBMS_DATAPUMP を使用してダンプファイルを作成する
- ステップ 4: 移行先の DB インスタンスへのデータベースリンクを作成する
- ステップ 5: DBMS_FILE_TRANSFER を使用して、エクスポートされたダンプファイルをターゲットの DB インスタンスにコピーする

- ステップ 6: DBMS_DATAPUMP を使用してターゲットの DB インスタンスにデータファイルをインポートする
- ステップ 7: クリーンアップ

Oracle Data Pump とデータベースリンクを使用したデータのインポートの要件

このプロセスには、次の要件があります。

- DBMS_FILE_TRANSFER パッケージと DBMS_DATAPUMP パッケージに対する実行権限が必要です。
- 移行元の DB インスタンスの DATA_PUMP_DIR ディレクトリに対する書き込み権限が必要です。
- 移行元のインスタンスと移行先の DB インスタンスにダンプファイルを保存するための十分なストレージ領域が必要です。

> **Note**
>
> このプロセスでは、DATA_PUMP_DIR ディレクトリ (すべての Oracle DB インスタンスで事前に設定されているディレクトリ) にダンプファイルをインポートします。このディレクトリはデータファイルと同じストレージボリュームにあります。ダンプファイルをインポートした場合、既存の Oracle データファイルのスペース占有率は高くなります。そのため、DB インスタンスではスペースの追加占有に対応できることを確認する必要があります。インポートしたダンプファイルは、DATA_PUMP_DIR ディレクトリから自動的に削除またはパージされることはありません。インポートしたダンプファイルを削除するには、Oracle ウェブサイトにある UTL_FILE.FREMOVE を使用します。

ステップ 1: RDS for Oracle のターゲットの DB インスタンスのユーザーに特権を付与する

RDS for Oracle のターゲットの DB インスタンスのユーザーに特権を付与するには、次のステップを行います。

1. SQL Plus または Oracle SQL Developer を使用して、データをインポートする RDS for Oracle DB インスタンスに接続します。Amazon RDS マスターユーザーとして接続します。DB インスタンスへの接続方法については、「Oracle DB インスタンスへの接続」を参照してください。
2. データをインポートする前に、必要なテーブルスペースを作成します。詳細については、「RDS for Oracle でのテーブルスペースの作成とサイズ設定」を参照してください。

3. データのインポート先のユーザーアカウントが存在しない場合は、ユーザーアカウントを作成し、必要なアクセス許可とロールを付与します。データを複数のユーザースキーマにインポートする場合は、各ユーザーアカウントを作成し、それぞれ必要な特権およびロールを付与します。

   例えば、以下のコマンドでは、*schema_1* という名前の新しいユーザーを作成して、このユーザーのスキーマ内にデータをインポートするために必要なアクセス許可とロールを付与します。

   ```
   CREATE USER schema_1 IDENTIFIED BY my-password;
   GRANT CREATE SESSION, RESOURCE TO schema_1;
   ALTER USER schema_1 QUOTA 100M ON users;
   ```

   > **Note**
   >
   > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

   この前の例では、新規ユーザーに特権 CREATE SESSION とロール RESOURCE を付与します。インポートするデータベースオブジェクトによっては、特権とロールの追加が必要になる場合があります。

   > **Note**
   >
   > *schema_1* を、このステップおよび次のステップのスキーマ名に置き換えます。

ステップ 2: ソースデータベースのユーザーに特権を付与する

SQL *Plus または Oracle SQL Developer を使用して、インポートするデータが含まれている RDS for Oracle DB インスタンスに接続します。必要に応じて、ユーザーアカウントを作成し、必要なアクセス許可を付与します。

> **Note**
>
> 移行元のデータベースが Amazon RDS インスタンスの場合、このステップは省略できます。エクスポートを行うには、Amazon RDS マスターユーザーアカウントを使用します。

次のコマンドでは、新しいユーザーを作成し、必要なアクセス許可を付与します。

```
CREATE USER export_user IDENTIFIED BY my-password;
GRANT CREATE SESSION, CREATE TABLE, CREATE DATABASE LINK TO export_user;
ALTER USER export_user QUOTA 100M ON users;
GRANT READ, WRITE ON DIRECTORY data_pump_dir TO export_user;
GRANT SELECT_CATALOG_ROLE TO export_user;
GRANT EXECUTE ON DBMS_DATAPUMP TO export_user;
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO export_user;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

ステップ 3: DBMS_DATAPUMP を使用してダンプファイルを作成する

ダンプファイルを作成するには、次の手順に従います。

1. 管理ユーザーまたはステップ 2 で作成したユーザーとして、SQL*Plus または Oracle SQL Developer を使用してソースの Oracle インスタンスに接続します。移行元のデータベースが Amazon RDS for Oracle DB インスタンスである場合は、Amazon RDS マスターユーザーとして接続します。
2. Oracle Data Pump ユーティリティを使用してダンプファイルを作成します。

   次のスクリプトでは、DATA_PUMP_DIR ディレクトリに sample.dmp というダンプファイルを作成します。

   ```
   DECLARE
     v_hdnl NUMBER;
   BEGIN
     v_hdnl := DBMS_DATAPUMP.OPEN(
       operation => 'EXPORT' ,
       job_mode  => 'SCHEMA' ,
       job_name  => null
     );
     DBMS_DATAPUMP.ADD_FILE(
       handle    => v_hdnl,
       filename  => 'sample.dmp'    ,
       directory => 'DATA_PUMP_DIR' ,
       filetype  => dbms_datapump.ku$_file_type_dump_file
     );
   ```

```
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl           ,
    filename  => 'sample_exp.log' ,
    directory => 'DATA_PUMP_DIR'  ,
    filetype  => dbms_datapump.ku$_file_type_log_file
  );
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl              ,
    'SCHEMA_EXPR'       ,
    'IN (''SCHEMA_1'')'
  );
  DBMS_DATAPUMP.METADATA_FILTER(
    v_hdnl,
    'EXCLUDE_NAME_EXPR',
    q'[IN (SELECT NAME FROM sys.OBJ$
           WHERE TYPE# IN (66,67,74,79,59,62,46)
           AND OWNER# IN
             (SELECT USER# FROM SYS.USER$
              WHERE NAME IN ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')
             )
          )
    ]',
    'PROCOBJ'
  );
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump ジョブは非同期的に開始されます。Data Pump ジョブのモニタリングについては、Oracle ドキュメントの「ジョブステータスのモニタリング」を参照してください。エクスポートログの内容は、`rdsadmin.rds_file_util.read_text_file` の手順を使用して表示できます。詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

ステップ 4: 移行先の DB インスタンスへのデータベースリンクを作成する

移行元の DB インスタンスと移行先の DB インスタンスの間にデータベースリンクを作成します。データベースリンクを作成してエクスポートダンプファイルを転送するには、DB インスタンスとのネットワーク接続がローカルの Oracle インスタンスに必要です。

このステップでは、前のステップと同じユーザーアカウントを使用して接続します。

同じ VPC 内またはピア接続された VPC 内の 2 つの DB インスタンス間のデータベースリンクを作成する場合、2 つの DB インスタンス間には有効なルートがある必要があります。各 DB インスタンスのセキュリティグループは他の DB インスタンスの受信と送信を許可する必要があります。セキュリティグループのインバウンドルールとアウトバウンドルールは、同じ VPC またはピアリング接続先 VPC からセキュリティグループを参照できます。詳細については、「VPC の DB インスタンスで使用するデータベースリンクの調整」を参照してください。

次のコマンドでは、ターゲットの DB インスタンスの Amazon RDS マスターユーザーに接続する to_rds という名前のデータベースリンクを作成します。

```
CREATE DATABASE LINK to_rds
  CONNECT TO <master_user_account> IDENTIFIED BY <password>
  USING '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>)
         (PORT=<listener port>))(CONNECT_DATA=(SID=<remote SID>)))';
```

ステップ 5: DBMS_FILE_TRANSFER を使用して、エクスポートされたダンプファイルをターゲットの DB インスタンスにコピーする

DBMS_FILE_TRANSFER を使用して、移行元のデータベースインスタンスから移行先の DB インスタンスにダンプファイルをコピーします。次のスクリプトでは、ソースのインスタンスから to_rds という名前のターゲットのデータベースリンク (前のステップで作成) に sample.dmp というダンプファイルをコピーします。

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object       => 'DATA_PUMP_DIR',
    source_file_name              => 'sample.dmp',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name         => 'sample_copied.dmp',
    destination_database          => 'to_rds' );
END;
/
```

ステップ 6: DBMS_DATAPUMP を使用してターゲットの DB インスタンスにデータファイルをインポートする

Oracle Data Pump を使用してスキーマを DB インスタンスにインポートします。METADATA_REMAP などの追加オプションが必要になる場合があります。

Amazon RDS マスターユーザーアカウントで DB インスタンスに接続し、インポートを実行します。

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'SCHEMA',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_copied.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

> **Note**
>
> Data Pump ジョブは非同期的に開始されます。Data Pump ジョブのモニタリングについては、Oracle ドキュメントの「ジョブステータスのモニタリング」を参照してください。インポートログの内容は、rdsadmin.rds_file_util.read_text_file の手順を使用して表示できます。詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

DB インスタンスでユーザーのテーブルを表示することで、データのインポートを検証できます。例えば、次のクエリでは、*schema_1* のテーブル数が返ります。

```
SELECT COUNT(*) FROM DBA_TABLES WHERE OWNER='SCHEMA_1';
```

ステップ 7: クリーンアップ

データをインポートしたら、保管が不要になったファイルは削除できます。次のコマンドを使用して DATA_PUMP_DIR のファイルを一覧表示できます。

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR')) ORDER BY MTIME;
```

DATA_PUMP_DIR 内の不要になったファイルを削除するには、次のコマンドを使用します。

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','<file name>');
```

例えば、次のコマンドは、"sample_copied.dmp" という名前のファイルが削除されます。

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample_copied.dmp');
```

Oracle エクスポート/インポートを使用したインポート

次のような場合、Oracle エクスポート/インポートユーティリティを使用して移行することも検討できます。

- データサイズが小さい。
- 2 進浮動小数点数や倍精度浮動小数点数などのデータ型を必要としない。

インポートプロセスにより、必要なスキーマオブジェクトが作成されます。したがって、オブジェクトを事前に作成するためのスクリプトを実行する必要はありません。

Oracle エクスポートユーティリティとインポートユーティリティをインストールする最も簡単な方法は、Oracle Instant Client をインストールすることです。ソフトウェアをダウンロードするには、https://www.oracle.com/database/technologies/instant-client.html にアクセスしてください。ドキュメントについては、Oracle Database Utilities マニュアルの「Instant Client for SQL*Loader、エクスポート、およびインポート」を参照してください。

テーブルをエクスポートしてからインポートするには

1. exp コマンドを使用して、移行元のデータベースからテーブルをエクスポートします。

   次のコマンドは、tab1、tab2、tab3 という名前のテーブルをエクスポートします。ダンプファイルは exp_file.dmp です。

   ```
   exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
   ```

   エクスポートでは、指定されたテーブルのスキーマとデータを含むバイナリダンプファイルが作成されます。
2. imp コマンドを使用して、このスキーマとデータを移行先のデータベースにインポートします。

   次のコマンドは、ダンプファイル exp_file.dmp からテーブル tab1、tab2、および tab3 をインポートします。

   ```
   imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
   TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
   ```

ユーザーのニーズに合うと考えられるエクスポートまたはインポートの使用方法は、他にもあります。詳細については、Oracle Database のドキュメントを参照してください。

Oracle SQL*Loader を使用したインポート

含まれるオブジェクトの数が制限されている大規模なデータベースには、Oracle SQL*Loader が適しているかもしれません。移行元のデータベースからのエクスポートと移行先のデータベースへの読み込みのプロセスは、スキーマに固有のものであるため、次の例では、サンプルのスキーマオブジェクトを作成し、移行元からエクスポートして、移行先のデータベースにデータを読み込みます。

Oracle SQL*Loader をインストールする最も簡単な方法は、Oracle Instant Client をインストールすることです。ソフトウェアをダウンロードするには、https://www.oracle.com/database/technologies/instant-client.html にアクセスしてください。ドキュメントについては、Oracle Database Utilities マニュアルの「Instant Client for SQL*Loader、エクスポート、およびインポート」を参照してください。

Oracle SQL*Loader を使用してデータをインポートするには

1. 次の SQL ステートメントを使用して、サンプルの移行元テーブルを作成します。

```
CREATE TABLE customer_0 TABLESPACE users
   AS (SELECT ROWNUM id, o.*
       FROM   ALL_OBJECTS o, ALL_OBJECTS x
       WHERE  ROWNUM <= 1000000);
```

2. 移行先の RDS for Oracle DB インスタンスで、データを読み込むための移行先テーブルを作成します。WHERE 1=2 句を使用すると、ALL_OBJECTS の構造体がコピーされますが、どの行もコピーされません。

```
CREATE TABLE customer_1 TABLESPACE users
  AS (SELECT 0 AS ID, OWNER, OBJECT_NAME, CREATED
      FROM   ALL_OBJECTS
      WHERE  1=2);
```

3. 移行元のデータベースからテキストファイルにデータを エクスポートします。以下の例では SQL*Plus を使用しています。移行するデータについて、データベース内のすべてのオブジェクトを エクスポートするためのスクリプトの生成が必要になる場合があります。

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD HH24:MI:SS'

SET LINESIZE 800 HEADING OFF FEEDBACK OFF ARRAY 5000 PAGESIZE 0
SPOOL customer_0.out
SET MARKUP HTML PREFORMAT ON
SET COLSEP ','

SELECT id, owner, object_name, created
FROM   customer_0;

SPOOL OFF
```

4. データの詳細について記述した制御ファイルを作成します。このステップを実行するためのスクリプトを記述する必要がある場合があります。

```
cat << EOF > sqlldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by '"'
(
  id           POSITION(01:10)    INTEGER EXTERNAL,
```

```
  owner         POSITION(12:41)    CHAR,
  object_name   POSITION(43:72)    CHAR,
  created       POSITION(74:92)    date "YYYY/MM/DD HH24:MI:SS"
)
EOF
```

必要に応じて、ステージング領域 (Amazon EC2 インスタンスなど) に上のコードで生成したファイルをコピーします。

5. 移行先のデータベース用の適切なユーザー名とパスワードで SQL*Loader を使用して、データをインポートします。

```
sqlldr cust_dba@targetdb CONTROL=sqlldr_1.ctl BINDSIZE=10485760 READSIZE=10485760
 ROWS=1000
```

Oracle マテリアライズドビューを使用した移行

大規模なデータセットを効率的に移行するために、Oracle マテリアライズドビューのレプリケーションを使用することができます。レプリケーションを使用すると、移行先テーブルと移行元テーブルとの同期を継続的に維持できます。このため、必要に応じて、後から Amazon RDS に切り替えることができます。

マテリアライズドビューを使用して移行する前に、以下の前提条件を満たしていることを確認してください。

- 移行先のデータベースから移行元のデータベースへのアクセスを設定します。次の例では、移行元データベースでアクセスルールが有効になっており、移行先の RDS for Oracle データベースが SQL*Net を経由して移行元にアクセスすることが許可されています。
- RDS for Oracle DB インスタンスから移行元のデータベースへのデータベースリンクを作成します。

マテリアライズドビューを使用してデータを移行するには

1. 同じパスワードで認証できるユーザーアカウントを、移行元と移行先の RDS for Oracle インスタンスの両方に作成します。次の例では、dblink_user という名前のユーザーを作成します。

```
CREATE USER dblink_user IDENTIFIED BY my-password
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp;
```

```
GRANT CREATE SESSION TO dblink_user;

GRANT SELECT ANY TABLE TO dblink_user;

GRANT SELECT ANY DICTIONARY TO dblink_user;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

2. 新しく作成したユーザーを使用して、移行先の RDS for Oracle インスタンスから移行元のインスタンスへのデータベースリンクを作成します。

```
CREATE DATABASE LINK remote_site
  CONNECT TO dblink_user IDENTIFIED BY my-password
  USING '(description=(address=(protocol=tcp) (host=my-host)
    (port=my-listener-port)) (connect_data=(sid=my-source-db-sid)))';
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

3. リンクをテストします。

```
SELECT * FROM V$INSTANCE@remote_site;
```

4. 移行元のインスタンスで、プライマリキーを持つサンプルテーブルとマテリアライズドビューのログを作成します。

```
CREATE TABLE customer_0 TABLESPACE users
  AS (SELECT ROWNUM id, o.*
      FROM   ALL_OBJECTS o, ALL_OBJECTS x
      WHERE  ROWNUM <= 1000000);

ALTER TABLE customer_0 ADD CONSTRAINT pk_customer_0 PRIMARY KEY (id) USING INDEX;
```

```
CREATE MATERIALIZED VIEW LOG ON customer_0;
```

5. 移行先の RDS for Oracle DB インスタンスで、マテリアライズドビューを作成します。

```
CREATE MATERIALIZED VIEW customer_0
  BUILD IMMEDIATE REFRESH FAST
  AS (SELECT *
      FROM   cust_dba.customer_0@remote_site);
```

6. 移行先の RDS for Oracle DB インスタンスで、マテリアライズドビューを更新します。

```
EXEC DBMS_MVIEW.REFRESH('CUSTOMER_0', 'f');
```

7. マテリアライズドビューを削除し、PRESERVE TABLE 句を含めて、マテリアライズドビューコンテナテーブルとその内容を保持します。

```
DROP MATERIALIZED VIEW customer_0 PRESERVE TABLE;
```

保持したテーブル名は、削除したマテリアライズドビューと同じです。

Amazon RDS for Oracle でのリードレプリカの使用

Oracle DB インスタンス間のレプリケーションを設定するには、レプリカデータベースを作成します。Amazon RDS リードレプリカの概要については、Amazon RDS リードレプリカの概要 を参照してください。Oracle レプリカと他の DB エンジンの違いの概要については、DB エンジンのリードレプリカ間の違い を参照してください。

トピック

- RDS for Oracle レプリカの概要
- RDS for Oracle レプリカの要件と考慮事項
- Oracle レプリカの作成の準備
- マウントモードでの RDS for Oracle レプリカの作成
- RDS for Oracle レプリカモードの変更
- RDS for Oracle レプリカのバックアップの使用
- Oracle Data Guard のスイッチオーバー操作の実行
- RDS for Oracle レプリカのトラブルシューティング
- RDS for Oracle を使用した REDO トランスポート圧縮

RDS for Oracle レプリカの概要

Oracle レプリカデータベースは、プライマリデータベースの物理コピーです。読み取り専用モードの Oracle レプリカは、リードレプリカと呼ばれます。マウントモードの Oracle レプリカは、マウントされたレプリカと呼ばれます。Oracle データベースでは、レプリカへの書き込みは許可されませんが、レプリカを昇格して書き込み可能にすることができます。昇格したリードレプリカには、昇格をリクエストされた時点までのレプリケートされたデータがあります。

次のビデオでは、RDS for Oracle の災害対策ついて紹介します。

詳細については、ブログ記事「Amazon RDS for Oracle クロスリージョン自動バックアップによる管理された災害対策 – パート 1」および「Amazon RDS for Oracle クロスリージョン自動バックアップによる管理された災害対策 – パート 2」を参照してください。

トピック

- 読み取り専用レプリカとマウントされたレプリカ
- CDB のレプリカを読み取る

- アーカイブされた REDO ログの保持
- Oracle レプリケーション中の停止

読み取り専用レプリカとマウントされたレプリカ

Oracle レプリカを作成または変更する場合、次のモードのいずれかにすることができます。

[Read-only]

これがデフォルトです。Active Data Guard は、ソースデータベースからすべてのリードレプリカデータベースに変更を送信し、適用します。

1 つのソース DB インスタンスから最大 5 つのリードレプリカを作成できます。すべての DB エンジンに適用されるリードレプリカの一般的な情報については、「DB インスタンスのリードレプリカの操作」を参照してください。Oracle Data Guard の詳細については、Oracle ドキュメントの「Oracle Data Guard の概要および管理」を参照してください。

マウント

この場合、レプリケーションでは Oracle Data Guard が使用されますが、レプリカデータベースはユーザー接続を受け付けません。マウントされたレプリカの主な用途は、クロスリージョンの災害対策です。

マウントされたレプリカは、読み取り専用のワークロードを処理できません。マウントされたレプリカは、アーカイブログ保持ポリシーに関係なく、適用後にアーカイブ REDO ログファイルを削除します。

同じソース DB インスタンスに対して、マウントされた DB レプリカと読み取り専用 DB レプリカを組み合わせて作成できます。読み取り専用レプリカをマウントモードに変更したり、マウントされたレプリカを読み取り専用モードに変更したりできます。いずれの場合も、Oracle データベースはアーカイブログの保持設定を維持します。

次の表は、適切なモードを選択するのに役立つ、マウントされたレプリカと読み取り専用レプリカの違いをまとめたものです。

マウントされたレプリカモードと読み取り専用レプリカモードの比較

機能	マウントモード	読み取り専用モード
主なユースケース	読み取りオフロードなしのディザスタリカバリ	読み取りオフロードによるディザスタリカバリ
クエリアクセス	いいえ (レプリカはユーザー接続を受け入れません)	はい (レプリカで SELECT クエリを実行できます)
ライセンス要件	Oracle Data Guard (Active Data Guard ライセンスは必要ありません)	Active Data Guard のライセンスが必要
リソースの使用状況	低 (クエリ処理オーバーヘッドなし)	高 (読み取りワークロードに対応)
いつ選択するか	フェイルオーバーにのみスタンバイが必要で、コストを最小限に抑えたい	プライマリからレポートクエリまたは分析クエリをオフロードする必要あり

CDB のレプリカを読み取る

RDS for Oracle は、シングルテナント設定とマルチテナント設定の両方で、Oracle Database 19c、21c および 26ai CDB について、Data Guard リードレプリカをサポートしています。CDB 以外と同様に、CDB でもリードレプリカを作成、管理、および昇格できます。マウントされたレプリカもサポートされています。次の利点を得ることができます。

- マネージドディザスタリカバリ、高可用性、レプリカへの読み取り専用アクセス
- 別の AWS リージョン でリードレプリカを作成する機能。
- 既存の RDS リードレプリカ API との統合: CreateDBInstanceReadReplica、PromoteReadReplica、および SwitchoverReadReplica

この機能を使用するには、レプリカとプライマリ DB インスタンスの両方に Active Data Guard ライセンスと Oracle Database Enterprise Edition ライセンスが必要です。CDB アーキテクチャの使用に関連する追加コストはありません。お支払いいただくのは DB インスタンスに対してのみです。

CDB アーキテクチャのシングルテナント設定とマルチテナント設定の詳細については、「RDS for Oracle CDB の概要」を参照してください。

アーカイブされた REDO ログの保持

プライマリ DB インスタンスにクロスリージョンのリードレプリカがない場合、Amazon RDS for Oracle は、ソース DB インスタンスで最低 2 時間のアーカイブ REDO ログを保持します。これは、`rdsadmin.rdsadmin_util.set_configuration` の `archivelog retention hours` の設定に関係なく当てはまります。

RDSは、2時間後、またはアーカイブログの保持時間の設定が経過した後のいずれか長い方の時間後に、ソース DBインスタンスからログを削除します。ログがデータベースに正常に適用された場合にのみ、アーカイブログの保持時間設定が経過すると、RDS はログをリードレプリカから削除します。

プライマリ DB インスタンスには、1 つ以上のクロスリージョンのリードレプリカが存在する場合があります。その場合 Amazon RDS for Oracle は、ソース DB インスタンスのトランザクションログが転送され、すべてのクロスリージョンリードレプリカに適用されるまで保持します。`rdsadmin.rdsadmin_util.set_configuration` については、アーカイブ REDO ログの保持を参照してください。

Oracle レプリケーション中の停止

リードレプリカを作成すると、Amazon RDS はソース DB インスタンスの DB スナップショットを取得し、レプリケーションを開始します。DB スナップショットオペレーションが始まると、ソース DB インスタンスでごく短時間の I/O 停止が発生します。通常、I/O 停止は約 1 秒続きます。ソース DB インスタンスがマルチ AZ 配置の場合は、I/O 停止を回避できます。スナップショットがセカンダリ DB インスタンスから取得されるためです。

DB スナップショットは Oracle レプリカになります。Amazon RDS は、サービスを中断することなく、ソースデータベースとレプリカに対して必要なパラメータとアクセス許可を設定します。同様に、レプリカを削除する場合も、停止は発生しません。

RDS for Oracle レプリカの要件と考慮事項

Oracle レプリカを作成する前に、以下の要件と考慮事項を確認してください。

トピック

- RDS for Oracle レプリカのバージョンとライセンス要件

- RDS for Oracle レプリカのオプショングループの制限事項
- RDS for Oracle レプリカのバックアップと復元に関する考慮事項
- RDS for Oracle レプリカに関する Oracle Data Guard の要件と制限事項
- RDS for Oracle レプリカのマルチテナント設定の制限
- RDS for Oracle レプリカに関するその他の考慮事項

RDS for Oracle レプリカのバージョンとライセンス要件

RDS for Oracle レプリカを作成する前に、次の点を考慮してください。

- レプリカが読み取り専用モードの場合は、Active Data Guard ライセンスがあることを確認してください。レプリカをマウントモードにした場合、Active Data Guard ライセンスは必要ありません。マウントされたレプリカをサポートするのは、Oracle DB エンジンだけです。
- Oracle レプリカは、Oracle Enterprise Edition (EE) でのみサポートされています。
- 非 CDB の Oracle レプリカは、Oracle Database 19c を実行している非 CDB インスタンスを使用して作成された DB インスタンスでのみサポートされます。
- Oracle レプリカは、2 つ以上の vCPU を持つ DB インスタンスクラスで実行されている DB インスタンスでのみ使用できます。ソース DB インスタンスは、db.t3.small インスタンスクラスを使用できません。
- ソース DB インスタンスとそのすべてのレプリカの Oracle DB エンジンバージョンは同じである必要があります。Amazon RDS では、レプリカのメンテナンスウィンドウに関係なく、ソース DB インスタンスのアップグレード後すぐにレプリカのアップグレードが行われます。クロスリージョンレプリカのメジャーバージョンアップグレードの場合、Amazon RDS は自動的に以下を実行します。
  - ターゲットバージョンのオプショングループを生成します。
  - 元のオプショングループから新しいオプショングループにすべてのオプションとオプション設定をコピーします。
  - アップグレードされたクロスリージョンレプリカを新しいオプショングループに関連付けます。

  DB エンジンバージョンのアップグレードの詳細については、「RDS for Oracle DB エンジンのアップグレード」を参照してください。

RDS for Oracle レプリカのオプショングループの制限事項

RDS for Oracle レプリカのオプショングループを使用する場合は、次の点を考慮してください。

- ソースとレプリカが同じ AWS リージョンにある場合、ソース DB インスタンスオプショングループとは異なるレプリカオプショングループを使用することはできません。

  ソースオプショングループまたはソースオプショングループメンバーシップへの変更はレプリカに反映されます。これらの変更は、レプリカのメンテナンスウィンドウに関係なく、出典 DB インスタンスに適用された後すぐにレプリカに適用されます。オプショングループの詳細については、「オプショングループを使用する」を参照してください。
- RDS for Oracle クロスリージョンレプリカを、レプリカ用に自動的に作成された専用オプショングループから削除することはできません。
- RDS for Oracle クロスリージョンレプリカの専用オプショングループを他の DB インスタンスに追加することはできません。
- RDS for Oracle クロスリージョンレプリカの専用オプショングループからは、レプリケートされていないオプションを追加または削除することはできません。ただし、次のオプションは除きます。
  - `NATIVE_NETWORK_ENCRYPTION`
  - `OEM`
  - `OEM_AGENT`
  - `SSL`

  RDS for Oracle クロスリージョンレプリカに他のオプションを追加するには、ソース DB インスタンスのオプショングループに追加します。オプションは、すべての出典 DB インスタンスのレプリカにもインストールされます。ライセンス供与オプションについては、レプリカに十分なライセンスがあることを確認してください。

  RDS for Oracle クロスリージョンレプリカを昇格するとき、昇格されたレプリカは、オプションの管理を含め、他の Oracle DB インスタンスと同じように動作します。レプリカは、明示的にまたはソース DB インスタンスを削除して暗黙的に昇格できます。

  オプショングループの詳細については、「オプショングループを使用する」を参照してください。
- RDS for Oracle クロスリージョンレプリカに `EFS_INTEGRATION` オプションを追加することはできません。

RDS for Oracle レプリカのバックアップと復元に関する考慮事項

RDS for Oracle レプリカを作成する前に、次の点を考慮してください。

- RDS for Oracle レプリカのスナップショットを作成したり、自動バックアップを有効にしたりするには、必ずバックアップ保持期間を手動で設定してください。自動バックアップは、デフォルトでは有効になっていません。
- レプリカのバックアップを復元するときは、バックアップが実行された時刻ではなく、データベース時刻に復元することになります。データベース時刻は、バックアップのデータにトランザクションが最後に適用された時刻を指します。レプリカはプライマリよりも数分または数時間遅れることがあるため、この違いは重大です。

  差を確認するには、`describe-db-snapshots` コマンドを使用します。レプリカのバックアップのデータベース時間である `SnapshotDatabaseTime` と、プライマリデータベースで最後に適用されたトランザクションである `OriginalSnapshotCreateTime` を比較します。

RDS for Oracle レプリカに関する Oracle Data Guard の要件と制限事項

RDS for Oracle レプリカを作成する前に、次の要件と制限事項に注意してください。

- プライマリ DB インスタンスがマルチテナントアーキテクチャのシングルテナントまたはマルチテナント構成を使用している場合は、次の点を考慮してください。
  - Enterprise Edition では Oracle Database 19c 以降を使用する必要があります。
  - プライマリ CDB インスタンスは `ACTIVE` ライフサイクルにある必要があります。
  - CDB 以外のプライマリインスタンスを CDB インスタンスに変換し、そのレプリカを同じ操作で変換することはできません。代わりに、CDB 以外のレプリカを削除し、プライマリ DB インスタンスを CDB に変換してから、新しいレプリカを作成してください。
- プライマリ DB インスタンスのログイントリガーで、`RDS_DATAGUARD` ユーザーへのアクセス、および `AUTHENTICATED_IDENTITY` の値が `RDS_DATAGUARD` または `rdsdb` であるすべてのユーザーへのアクセスを許可する必要があります。また、トリガーで `RDS_DATAGUARD` ユーザーの現在のスキーマを設定しないでください。
- Data Guard ブローカープロセスからの接続のブロックを回避するには、制限セッションを有効にしないでください。制限セッションの詳細については、「制限セッションの有効化と無効化」を参照してください。

RDS for Oracle レプリカのマルチテナント設定の制限

RDS for Oracle レプリカでマルチテナント設定を使用する場合は、次の制限に注意してください。

- テナントデータベースを作成、削除、または変更できるのは、プライマリ DB インスタンス上のみです。これらの変更は、レプリカに自動的に伝播されます。
- RDS for Oracle のプライマリ、ソース、またはレプリカのテナントデータベースをカスタム文字セットを使用して作成することはできません。カスタム文字セットが必要な場合は、DB インスタンスのリードレプリカを作成する前にテナントデータベースを作成します。

RDS for Oracle レプリカに関するその他の考慮事項

RDS for Oracle レプリカを作成する前に、次の点を考慮してください。

- 追加のストレージボリュームがある DB インスタンスの RDS for Oracle レプリカを作成する場合、RDS はレプリカに追加のストレージボリュームを自動的に設定します。ただし、プライマリ DB インスタンスのストレージボリュームに加えられた後続の変更は、レプリカに自動的に適用されません。
- プライマリ DB インスタンスにストレージボリュームを追加した場合、RDS はレプリカにストレージボリュームを自動的に追加しません。追加のストレージボリュームを追加するには、レプリカを変更する必要があります。
- プライマリ DB インスタンスのストレージサイズや IOPS などのストレージボリューム設定を変更しても、RDS はレプリカのストレージボリュームを自動的に変更しません。ストレージボリュームの設定を更新するには、レプリカを変更する必要があります。
- ボリューム間でデータファイルの場所を管理する場合、プライマリインスタンスで行われた変更はレプリカに自動的に同期されないことに注意してください。
  - 読み取り専用レプリカの場合: パラメータグループ設定を使用してデフォルトのファイルの場所を制御するか、作成後に手動でファイルを移動できます。
  - マウントされたレプリカの場合: プライマリデータベース内のデータファイルの場所を手動で変更するには、それらの変更を反映するためにマウントされたレプリカを再作成する必要があります。これを回避するには、パラメータグループ設定を使用してデフォルトのファイルの場所を管理することをお勧めします。
- DB インスタンスが 1 つ以上のクロスリージョンレプリカのソースである場合、ソース DB は、すべてのクロスリージョンレプリカに適用されるまで、アーカイブされた REDO ログファイルに保持されます。アーカイブされた REDO ログにより、ストレージの消費が増える場合があります。
- RDS 自動化の中断を避けるために、システムトリガーでは、特定のユーザーがプライマリデータベースとレプリカデータベースへのログオンを許可する必要があります。システムトリガーには、DDL、ログオン、およびデータベースロールトリガーが含まれます。以下のサンプルコードに記載されているユーザーを除外するために、トリガーにコードを追加することをお勧めします。

```
-- Determine who the user is
SELECT SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') INTO CURRENT_USER FROM DUAL;
-- The following users should always be able to login to either the Primary or
 Replica
IF CURRENT_USER IN ('master_user', 'SYS', 'SYSTEM', 'RDS_DATAGUARD', 'rdsdb') THEN
RETURN;
END IF;
```

- ブロック変更の追跡は、読み取り専用レプリカではサポートされますが、マウントされたレプリカではサポートされません。マウントされたレプリカを読み取り専用レプリカに変更し、ブロック変更の追跡を有効にすることができます。詳細については、「ブロック変更追跡の有効化/無効化」を参照してください。
- ソースデータベースが Secrets Manager でマスターユーザー認証情報を管理する場合、Oracle リードレプリカを作成することはできません。

Oracle レプリカの作成の準備

レプリカの使用をスタートする前に、次のタスクを実行します。

トピック

- 自動バックアップの有効化
- 強制ログ記録モードの有効化
- ログ記録設定の変更
- MAX_STRING_SIZE パラメータの設定
- コンピューティングとストレージのリソース計画

自動バックアップの有効化

DB インスタンスがソース DB インスタンスとして機能するには、必ずソース DB インスタンスで自動バックアップを有効にします。この手順の実行方法については、「自動バックアップの有効化」を参照してください。

強制ログ記録モードの有効化

強制ログ記録モードを有効化することを推奨します。強制ログ記録モードでは、NOLOGGING がデータ定義言語 (DDL) ステートメントとともに使用されている場合でも、Oracle データベースは REDO レコードを書き込みます。

強制ログ記録モードを有効にするには

1. SQL Developer などのクライアントツールを使用して、Oracle データベースにログインします。
2. 次の手順を実行して、強制ログモードを有効にします。

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

この手順の詳細については、「強制ログ作成の設定」を参照してください。

ログ記録設定の変更

サイズ m の n 個のオンライン REDO ログの場合、RDS はプライマリ DB インスタンスとすべてのレプリカにサイズ m の n+1 個のスタンバイログを自動的に作成します。プライマリのログ記録設定を変更するたびに、その変更はレプリカに自動的に反映されます。

ログ記録設定を変更する場合は、以下のガイドラインを考慮してください。

- DB インスタンスをレプリカのソースにする前に変更を完了することをお勧めします。RDS for Oracle は、ソースになった後のインスタンスの更新もサポートしています。
- プライマリ DB インスタンスのログ記録設定を変更する前に、各レプリカに新しい設定に対応するのに十分なストレージがあることを確認してください。

DB インスタンスのログ設定を変更するには、Amazon RDS の手順 `rdsadmin.rdsadmin_util.add_logfile` と `rdsadmin.rdsadmin_util.drop_logfile` を使用します。詳細については、「オンライン REDO ログの追加」および「オンライン REDO ログの削除」を参照してください。

MAX_STRING_SIZE パラメータの設定

Oracle レプリカを作成する前に、`MAX_STRING_SIZE` パラメータの設定が、ソース DB インスタンスおよびそのレプリカと同じであることを確認します。そのためには、同じパラメータグループと関連付けます。ソースとレプリカのパラメータグループが異なる場合は、`MAX_STRING_SIZE` を同じ値に設定できます。このパラメータの設定の詳細については、「新しい DB インスタンスで拡張データ型を有効にする」を参照してください。

コンピューティングとストレージのリソース計画

ソース DB インスタンスとそのレプリカのサイズが、運用負荷に合わせる上でコンピューティングとストレージの観点から適切に設定されていることを確認してください。レプリカのコンピューティング、ネットワーク、またはストレージがリソースの容量に達すると、レプリカはソースからの変更の受信または適用を停止します。Amazon RDS for Oracle が、ソース DB インスタンスとそのレプリカ間のレプリカラグの軽減のために介入することはありません。レプリカのストレージや CPU リソースは、そのソースや他のレプリカとは独立して変更することができます。

マウントモードでの RDS for Oracle レプリカの作成

デフォルトでは、Oracle レプリカは読み取り専用です。マウントモードでレプリカを作成するには、コンソール、AWS CLI、RDS API のいずれかを使用します。

コンソール

ソース Oracle DB インスタンスからマウントされたレプリカを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. マウントされたレプリカのソースとして使用する Oracle DB インスタンスを選択します。
4. アクションでレプリカの作成を選択します。
5. レプリカモードで、マウントを選択します。
6. 使用する設定を選択します。DB インスタンス識別子に、リードレプリカの名前を入力します。必要に応じて他の設定を変更します。
7. リージョンで、マウントされたレプリカを起動するリージョンを選択します。
8. インスタンスサイズとストレージタイプを選択します。リードレプリカでもソース DB インスタンスと同じ DB インスタンスクラスとストレージタイプを使用することをお勧めします。
9. マルチ AZ 配置でスタンバイインスタンスの作成を選択して、マウントされたレプリカのフェイルオーバーをサポートするために別のアベイラビリティーゾーンにレプリカのスタンバイを作成します。ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マウントされたレプリカをマルチ AZ DB インスタンスとして作成できます。
10. 使用する他の設定を選択します。
11. レプリカの作成を選択します。

データベースページで、マウントされたレプリカにロールとして [レプリカ] が割り当てられます。

AWS CLI

マウントモードで Oracle レプリカを作成するには、`--replica-mode` コマンド create-db-instance-read-replica で `mounted` を AWS CLI に設定します。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance \
    --replica-mode mounted
```

Windows の場合:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance ^
    --replica-mode mounted
```

読み取り専用レプリカをマウント状態に変更するには、`--replica-mode` コマンド modify-db-instance で `mounted` を AWS CLI に設定します。マウントされたレプリカを読み取り専用モードにするには、`--replica-mode` を `open-read-only` に設定します。

RDS API

マウントモードで Oracle レプリカを作成するには、RDS API オペレーション CreateDBInstanceReadReplica で `ReplicaMode=mounted` を指定します。

RDS for Oracle レプリカモードの変更

既存のレプリカのレプリカモードを変更するには、コンソール、AWS CLI、RDS API のいずれかを使用します。マウントモードに変更すると、レプリカはすべてのアクティブな接続を切断します。読み取り専用モードに変更すると、Amazon RDS は Active Data Guard を初期化します。

変更には数分かかる場合があります。オペレーション中、DB インスタンスのステータスは modifying に変わります。ステータス変更の詳細については、「Amazon RDS DB インスタンスのステータスの表示」を参照してください。

コンソール

Oracleレプリカのレプリカモードをマウントから読み取り専用に変更するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. マウントされたレプリカのデータベースを選択します。
4. [Modify] を選択します。
5. [レプリカモード] で、[読み取り専用] を選択します。
6. 変更する他の設定を選択します。
7. [Continue] を選択します。
8. [変更のスケジューリング] で、[すぐに適用] を選択します。
9. [DB インスタンスの変更] を選択します。

AWS CLI

リードレプリカをマウントモードに変更するには、`--replica-mode` コマンド modify-db-instance で `mounted` を AWS CLI に設定します。マウントされたレプリカを読み取り専用モードに変更するには、`--replica-mode` を `open-read-only` に設定します。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier myreadreplica \
    --replica-mode open-read-only
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier myreadreplica ^
    --replica-mode open-read-only
```

RDS API

読み取り専用レプリカをマウントモードに変更するには、ModifyDBInstance で `ReplicaMode=mounted` を設定します。マウントされたレプリカを読み取り専用モードに変更するには、`ReplicaMode=open-read-only` を設定します。

RDS for Oracle レプリカのバックアップの使用

RDS for Oracle レプリカのバックアップを作成および復元できます。自動バックアップと手動スナップショットの両方がサポートされています。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。以下のセクションでは、プライマリと RDS for Oracle レプリカのバックアップ管理の主な違いについて説明します。

RDS for Oracle レプリカのバックアップを有効にする

Oracle レプリカでは、デフォルトで自動バックアップが有効になっていません。バックアップ保持期間を 0 以外の正の値に設定することで、自動バックアップを有効にします。

コンソール

自動バックアップをすぐに有効にするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスまたはマルチ AZ DB クラスターを選択します。
3. [Modify] (変更) を選択します。
4. [バックアップ保持期間] で、ゼロ以外の正の値 (3 日など) を選択します。
5. [続行] を選択します。
6. [すぐに適用] を選択します。
7. [DB インスタンスの変更] または [クラスターの変更] を選択して変更を保存し、自動バックアップを有効にします。

AWS CLI

自動バックアップを有効にするには、AWS CLI の modify-db-instance コマンドまたは modify-db-cluster コマンドを使用します。

以下のパラメータを含めます。

- `--db-instance-identifier` (またはマルチ AZ DB クラスターの場合は `--db-cluster-identifier`)
- `--backup-retention-period`
- `--apply-immediately` または `--no-apply-immediately`

次の例では、バックアップ保持期間を 3 日に設定して、自動バックアップを有効にします。変更はすぐに適用されます。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier my_db_instance  \
    --backup-retention-period 3 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier my_db_instance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

RDS API

自動バックアップを有効にするには、以下の必須パラメータを指定して RDS API の ModifyDBInstance オペレーションまたは ModifyDBCluster オペレーションを使用します。

- `DBInstanceIdentifier-または-DBClusterIdentifier`
- `BackupRetentionPeriod`

RDS for Oracle レプリカのバックアップの復元

Oracle レプリカのバックアップは、プライマリインスタンスのバックアップを復元する場合と同様に復元できます。詳細については次を参照してください:

- DB インスタンスへの復元
- Amazon RDS の DB インスタンスを特定の時点に復元する

レプリカバックアップを復元する際の主な考慮事項は、復元する時点を決定することです。データベース時刻は、バックアップのデータにトランザクションが最後に適用された時刻を指します。レプリカのバックアップを復元するときは、バックアップが完了した時刻ではなく、データベースの時刻に復元することになります。RDS for Oracle レプリカはプライマリよりも数分または数時間遅れることがあるため、この違いは重大です。このため、レプリカのバックアップのデータベースの時刻、つまり復元する時点は、バックアップ作成時刻よりもはるかに早い場合があります。

データベースの時刻と作成時刻の差を確認するには、describe-db-snapshots コマンドを使用します。レプリカのバックアップのデータベース時間である SnapshotDatabaseTime と、プライマリデータベースで最後に適用されたトランザクションである OriginalSnapshotCreateTime を比較します。次の例は、2 つの時刻の差を示しています。

```
aws rds describe-db-snapshots \
    --db-instance-identifier my-oracle-replica
    --db-snapshot-identifier my-replica-snapshot

{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "my-replica-snapshot",
            "DBInstanceIdentifier": "my-oracle-replica",
            "SnapshotDatabaseTime": "2022-07-26T17:49:44Z",
            ...
            "OriginalSnapshotCreateTime": "2022-07-26T19:49:44Z"
        }
    ]
}
```

Oracle Data Guard のスイッチオーバー操作の実行

スイッチオーバーでは、プライマリデータベースとスタンバイデータベースとの間でロールが入れ替わります。スイッチオーバー中、元のプライマリデータベースはスタンバイロールに移行し、元のスタンバイデータベースはプライマリロールに移行します。

Oracle Data Guard 環境では、プライマリデータベースは 1 つ以上のスタンバイデータベースをサポートします。プライマリデータベースからスタンバイデータベースへ、マネージド型のスイッチオーバーベースのロール移行を実行できます。

トピック

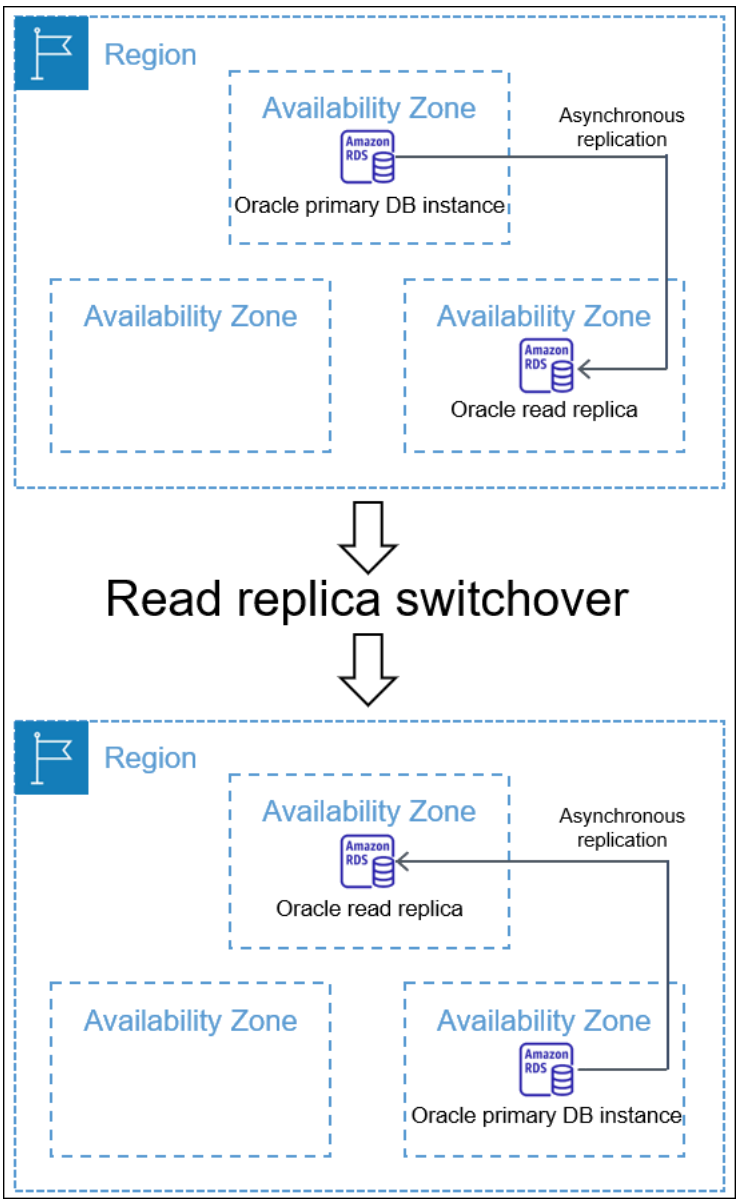
- Oracle Data Guard のスイッチオーバー機能の概要

- Oracle Data Guard のスイッチオーバーの要件
- Oracle Data Guard のスイッチオーバーの開始
- Oracle Data Guard のスイッチオーバーのモニタリング

Oracle Data Guard のスイッチオーバー機能の概要

Amazon RDS は、Oracle データベースレプリカの完全マネージド型のスイッチオーバーベースのロール移行をサポートしています。マウントされているか、読み取り専用で開いているスタンバイデータベースへのスイッチオーバーのみを開始できます。

レプリカは、個別の AWS リージョンに配置するか、単一のリージョン内における複数の異なるアベイラビリティーゾーン (AZ) に配置できます。すべての AWS リージョンがサポートされています。



スイッチオーバーは、リードレプリカの昇格とは異なります。スイッチオーバーでは、ソース DB インスタンスとレプリカ DB インスタンスの役割が入れ替わります。昇格では、リードレプリカはソース DB インスタンスになりますが、ソース DB インスタンスはレプリカになりません。詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

トピック

- Oracle Data Guard スイッチオーバーの利点
- サポートされている Oracle Database のバージョン
- Oracle Data Guard のスイッチオーバーの費用
- Oracle Data Guard のスイッチオーバーの仕組み

Oracle Data Guard スイッチオーバーの利点

RDS for Oracle のリードレプリカと同様に、マネージド型スイッチオーバーは Oracle Data Guard に依存しています。このオペレーションは、データ損失がゼロになるように設計されています。Amazon RDS は、スイッチオーバーの次の要素を自動化します。

- 新しいスタンバイデータベースを元のスタンバイデータベースと同じ状態 (マウントされているか、読み取り専用)、プライマリデータベースと指定されたスタンバイデータベースのロールを逆にする
- データの整合性を確保する
- 移行後もレプリケーション構成を維持する
- 新しいスタンバイデータベースのロールが元のプライマリロールに戻ることができることで、逆転の繰り返しをサポートする

サポートされている Oracle Database のバージョン

Oracle Data Guard スイッチオーバーは、Oracle Database 19c 以降のリリースでサポートされています。

Oracle Data Guard のスイッチオーバーの費用

Oracle Data Guard のスイッチオーバー機能には追加コストはかかりません。Oracle Database Enterprise Edition には、マウントモードのスタンバイデータベースのサポートが含まれています。スタンバイデータベースを読み取り専用モードで開くには、Oracle Active Data Guard オプションが必要です。

Oracle Data Guard のスイッチオーバーの仕組み

Oracle Data Guard のスイッチオーバーはフルマネージド型操作です。スタンバイデータベースのスイッチオーバーを開始するには、CLI コマンド `switchover-read-replica` を実行します。すると、Amazon RDS はレプリケーション構成のプライマリロールとスタンバイロールを変更します。

オリジナルのスタンバイとオリジナルのプライマリがスイッチオーバー前のロールです。新しいスタンバイと新しいプライマリがスイッチオーバー後のロールです。バイスタンダーのレプリカは、Oracle Data Guard 環境ではスタンバイデータベースとして機能しますが、ロールの切り替えは行わないレプリカデータベースです。

トピック

- Oracle Data Guard のスイッチオーバーのステージ

- Oracle Data Guard のスイッチオーバー操作後

Oracle Data Guard のスイッチオーバーのステージ

スイッチオーバーを実行するには、Amazon RDS が次のステップを実行する必要があります。

1. 元のプライマリデータベースの新しいトランザクションをブロックします。スイッチオーバー中、Amazon RDS は Oracle Data Guard 設定のすべてのデータベースのレプリケーションを中断します。スイッチオーバー中、元のプライマリデータベースは書き込み要求を処理できません。
2. 未適用のトランザクションを元のスタンバイデータベースに送り、適用します。
3. 新しいスタンバイデータベースを読み取り専用モードまたはマウントモードで再起動します。モードは、スイッチオーバー前の元のスタンバイデータベースのオープン状態によって異なります。
4. 新しいプライマリデータベースを読み取り/書き込みモードで開きます。

Oracle Data Guard のスイッチオーバー操作後

Amazon RDS は、プライマリデータベースとスタンバイデータベースのロールを切り替えます。アプリケーションを再接続し、その他の必要な構成を実行するのはユーザーです。

トピック

- 成功基準
- 新しいプライマリデータベースへの接続
- 新しいプライマリデータベースの構成

成功基準

Oracle Data Guard のスイッチオーバーは、元のスタンバイデータベースが次の処理を実行したときに成功したと判断できます。

- 新しいプライマリデータベースとしてのロールに移行する
- 再構成を完了する

ダウンタイムを抑えるため、新しいプライマリデータベースはできるだけ早くアクティブになります。Amazon RDS はバイスタンダーのレプリカを非同期で設定するため、これらのレプリカは元のプライマリデータベースの後でアクティブになる可能性があります。

新しいプライマリデータベースへの接続

Amazon RDS は、スイッチオーバー後、現在のデータベース接続を新しいプライマリデータベースに伝達しません。Oracle Data Guard のスイッチオーバーが完了したら、アプリケーションを新しいプライマリデータベースに再接続します。

新しいプライマリデータベースの構成

新しいプライマリデータベースへのスイッチオーバーを実行するために、Amazon RDS は元のスタンバイデータベースのモードをオープンに変更します。ロールの変更がデータベースの唯一の変更です。Amazon RDS によって Multi-AZ レプリケーションなどの機能は設定されません。

オプションが異なるクロスリージョンレプリカへのスイッチオーバーを実行しても、新しいプライマリデータベースでは独自のオプションが保持されます。Amazon RDS は元のプライマリデータベースのオプションを移行しません。元のプライマリデータベースで SSL、NNE、OEM、OEM_AGENT などのオプションが指定されている場合も、Amazon RDS はそれらを新しいプライマリデータベースに伝達しません。

> **Note**
>
> スイッチオーバーはデータベースロールのみを変更するため、新しいプライマリデータベースは、有効にするまでマルチ AZ によって保護されません。スイッチオーバーが完了したら、新しいプライマリデータベースを確認し、必要に応じてマルチ AZ を有効にして、アプリケーションに必要なオプション (SSL、NNE、OEM、OEM_AGENT など) を再設定します。

Oracle Data Guard のスイッチオーバーの要件

Oracle Data Guard のスイッチオーバーを開始する前に、レプリケーション環境が次の要件を満たしていることを確認します。

- 元のスタンバイデータベースがマウントされているか、読み取り専用で開かれている。
- 元のスタンバイデータベースで自動バックアップが有効になっている。
- 元のプライマリデータベースと元のスタンバイデータベースが `available` 状態である。
- 元のプライマリデータベースと元のスタンバイデータベースに、`required`、`next window`、または `in progress` 状態の保留中のメンテナンスアクションがない。これらの状態のアクションはスイッチオーバーをブロックします。保留中のメンテナンス更新のステータスを確認する方法については、「保留中のメンテナンス更新の表示」を参照してください。

保留中のメンテナンスアクションが `available` 状態の場合は、スイッチオーバーをブロックしません。RDS for Oracle は、オペレーティングシステム (OS) の更新を `available` 状態で頻繁にリリースします。これらの保留中の OS 更新は、次回のメンテナンスウィンドウにスケジュールして `next window` 状態にしない限り、スイッチオーバーをブロックしません。

> **Note**
>
> スケジュールされたメンテナンスアクションを延期してスイッチオーバーを実行できるようにする場合は、RDS コンソールで [アクション]、[アップグレードを延期] の順に選択します。スイッチオーバーがブロックされないようにするには、保留中のメンテナンスアクションを適用するか、メンテナンスウィンドウをスイッチオーバー前の間隔に移動することもできます。詳細については、re:Post の記事「How to remove RDS pending maintenance items」を参照してください。

- 元のスタンバイデータベースがレプリケーション状態である。
- プライマリデータベースまたはスタンバイデータベースのいずれかがスイッチオーバーライフサイクルで、現在スイッチオーバーを開始しようとしていない。スイッチオーバー後にレプリカデータベースの再構成中、Amazon RDS は別のスイッチオーバーを開始できないようにします。

> **Note**
>
> バイスタンダーのレプリカとは、スイッチオーバーの対象ではない Oracle Data Guard 構成のレプリカです。バイスタンダーのレプリカは、スイッチオーバー時、どのような状態でもかまいません。

- 元のスタンバイデータベースが、元のプライマリデータベースの構成と希望に近い構成になっている。元のプライマリデータベースと元のスタンバイデータベースでオプションが異なるシナリオを想定します。スイッチオーバーが完了しても、Amazon RDS では、新しいプライマリデータベースを元のプライマリデータベースと同じオプションで自動的には再設定しません。
- 切り替えを開始する前に、目的のマルチ AZ 配置を設定します。Amazon RDS では、切り替えの一環としてマルチ AZ を管理しません。マルチ AZ 配置はそのままになります。

  db_maz がマルチ AZ 配置のプライマリデータベースで、db_saz がシングル AZ レプリカであると仮定します。db_maz から db_saz への切り替えを開始します。その後、db_maz はマルチ AZ レプリカデータベースになり、db_saz はシングル AZ プライマリデータベースになります。これで、新しいプライマリデータベースは マルチ AZ 配置によって保護されなくなりました。

- クロスリージョンの切り替えに備え、プライマリデータベースはレプリケーション設定以外の DB インスタンスと同じオプショングループを使用しません。クロスリージョンの切り替えを正常に行うには、現在のプライマリデータベースとそのリードレプリカが、現在のプライマリデータベースのオプショングループを使用する DB インスタンスのみである必要があります。そうではない場合、Amazon RDS が切り替えを妨げます。

Oracle Data Guard のスイッチオーバーの開始

RDS for Oracle リードレプリカをプライマリロールに切り替え、以前のプライマリ DB インスタンスをレプリカロールに切り替えることができます。

スイッチオーバーを開始する前に、以下を確認してください。

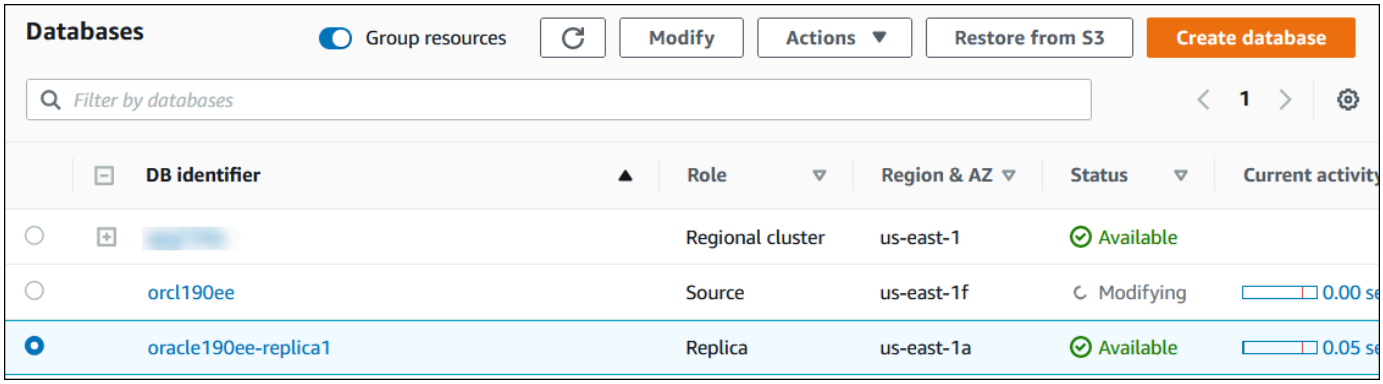
- `ReplicaLag` CloudWatch メトリクスはゼロまたはほぼゼロです。
- プライマリデータベースでは、長時間実行されるトランザクションはアクティブではありません。
- アプリケーション接続文字列は、エンドポイントの変更を処理するように設定されています。または、スイッチオーバー後に更新する計画があります。
- レプリカが正常な状態であることを確認しました (ステータスは `available`)。

コンソール

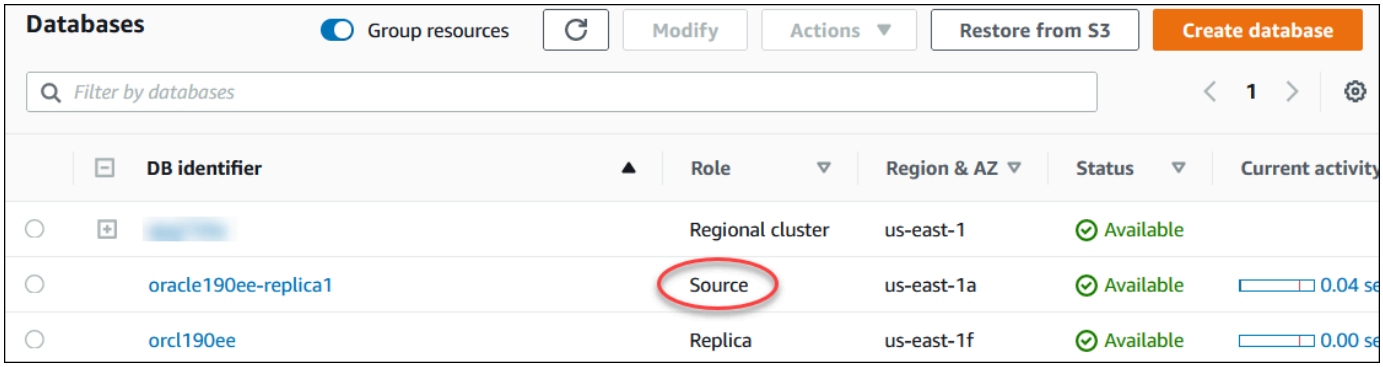
Oracle リードレプリカをプライマリ DB ロールに切り替えるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS コンソールで、[Databases (データベース)] を選択します。

   [Databases (データベース)] ペインが表示されます。各リードレプリカには、[Role (ロール)] 列に [Replica (レプリカ)] があります。
3. プライマリロールに切り替えるリードレプリカを選択します。
4. [Actions] (アクション) で、[Switch over replica] (レプリカを切り替える) を選択します。
5. [I acknowledge] (承認します) を選択します。次に、[Switch over replica] (レプリカを切り替える) を選択します。
6. [Databases] (データベース) ページで、スイッチオーバーの進行状況をモニタリングします。



スイッチオーバーが完了すると、ターゲットのロールがレプリカからソースに変わります。



AWS CLI

Oracle レプリカをプライマリ DB ロールに切り替えるには、AWS CLI [switchover-read-replica](#) コマンドを使用します。次の例では、*replica-to-be-made-primary* という名前の Oracle レプリカを新しいプライマリデータベースに切り替えます。

Example

Linux、macOS、Unix の場合:

```
aws rds switchover-read-replica \
    --db-instance-identifier replica-to-be-made-primary
```

Windows の場合:

```
aws rds switchover-read-replica ^
    --db-instance-identifier replica-to-be-made-primary
```

RDS API

Oracle レプリカをプライマリ DB ロールに切り替えるには、必須パラメータ `DBInstanceIdentifier` を使用して、Amazon RDS API `SwitchoverReadReplica` 操作を呼び出します。このパラメータは、プライマリ DB ロールを継承する Oracle レプリカの名前を指定します。

Oracle Data Guard のスイッチオーバーのモニタリング

インスタンスの状態を確認するには、AWS CLI コマンド `describe-db-instances` を使用します。次のコマンドでは、DB インスタンス *orcl2* のステータスをモニタリングします。このデータベースは、スイッチオーバー前はスタンバイデータベースでしたが、スイッチオーバー後は新しいプライマリデータベースになります。

```
aws rds describe-db-instances \
    --db-instance-identifier orcl2
```

スイッチオーバーが正常に完了したことを確認するには、`V$DATABASE.OPEN_MODE` のクエリを実行します。新しいプライマリデータベースの値が `READ WRITE` であることを確認します。

```
SELECT OPEN_MODE FROM V$DATABASE;
```

スイッチオーバー関連のイベントを検索するには、AWS CLI コマンド `describe-events` を使用します。次の例では、*orcl2* インスタンスのイベントを検索します。

```
aws rds describe-events \
    --source-identifier orcl2 \
    --source-type db-instance
```

RDS for Oracle レプリカのトラブルシューティング

このセクションでは、レプリケーションに関する潜在的な問題と解決策について説明します。

トピック

- Oracle レプリケーションラグのモニタリング
- トリガーを追加または変更した後の Oracle レプリケーションの失敗をトラブルシューティングする

Oracle レプリケーションラグのモニタリング

Amazon CloudWatch のレプリケーションラグをモニタリングするには、Amazon RDS `ReplicaLag` メトリクスを表示します。レプリケーションのラグタイムについては、「リードレプリケーションのモニタリング」および「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。

リードレプリカのラグタイムが長すぎる場合は、次のビューをクエリします。

- `V$ARCHIVED_LOG` - リードレプリカに適用されたコミットが表示されます。
- `V$DATAGUARD_STATS` - `ReplicaLag` メトリクスを占めるコンポーネントの詳細を示します。

  トランスポート遅延 (REDO がレプリカに到達するまでの時間) と適用遅延 (REDO が適用されるまでの時間) を表示するには、リードレプリカで次のクエリを実行します。

```
SELECT name, value, time_computed, datum_time FROM V$DATAGUARD_STATS WHERE name IN
 ('transport lag', 'apply lag');
```

- `V$DATAGUARD_STATUS` - Oracle の内部レプリケーションプロセスのログ出力を示します。

マウントされたレプリカの場合、ラグタイムが長すぎる場合は、`V$` ビューをクエリすることができません。代わりに、以下の手順を実行します。

- CloudWatch で `ReplicaLag` メトリクスをチェックします。
- コンソールでレプリカのアラートログファイルを確認します。回復メッセージでエラーを探します。メッセージにはログシーケンス番号が記載されていて、これをプライマリシーケンス番号と比較できます。(詳しくは、「Amazon RDS for Oracle データベースのログファイル」を参照してください。)

トリガーを追加または変更した後の Oracle レプリケーションの失敗をトラブルシューティングする

トリガーを追加または変更し、その後レプリケーションが失敗した場合は、問題はトリガーにある可能性があります。トリガーで、レプリケーションから RDS で必要な次のユーザーアカウントが除外されていることを確認します。

- 管理者権限を持つユーザーアカウント
- `SYS`
- `SYSTEM`

- RDS_DATAGUARD
- rdsdb

(詳しくは、「RDS for Oracle レプリカに関するその他の考慮事項」を参照してください。)

RDS for Oracle を使用した REDO トランスポート圧縮

RDS for Oracle REDO トランスポート圧縮を使用して、プライマリ DB インスタンスとスタンバイレプリカ間のレプリケーションパフォーマンスを向上させます。これは、ネットワーク帯域幅が限られている環境や高レイテンシーの接続で特に役立ちます。

REDO トランスポート圧縮のライセンスの取得

REDO トランスポート圧縮は、Oracle Advanced Compression オプションの一部です。REDO トランスポート圧縮を使用するには、Oracle Advanced Compression オプションの有効なライセンスが必要です。ライセンス情報については、Oracle の担当者にお問い合わせください。

REDO トランスポート圧縮の設定

REDO トランスポート圧縮を設定するには、rds.replica.redo_compression パラメータを使用できます。このパラメータは、Oracle バージョン 19c、21c、および 26ai で使用できます。

rds.replica.redo_compression パラメータには、次の値を指定できます。

- DISABLE – REDO トランスポート圧縮を無効にするデフォルト値。
- ENABLE – デフォルトのアルゴリズム ZLIB による REDO トランスポート圧縮を有効にする値。
- ZLIB – ZLIB アルゴリズムを使用して REDO トランスポート圧縮を明示的に有効にする値。これにより、圧縮率が向上します。
- LZO – 特に解凍中に圧縮速度を最適化する LZO アルゴリズムを使用して、REDO トランスポート圧縮を明示的に有効にする値。

REDO トランスポート圧縮のパフォーマンスに関する考慮事項

圧縮および解凍オペレーションは、プライマリインスタンスとスタンバイインスタンスの両方で CPU リソースを消費します。REDO トランスポート圧縮を使用する場合は、インスタンスリソースの使用状況とネットワーク条件を考慮してください。

REDO トランスポート圧縮に関連するトピック

REDO トランスポート圧縮の設定の詳細については、次のリソースを参照してください。

- Amazon RDS DB インスタンスの DB パラメータグループ
- Oracle Database 19c リリースノートの「RedoCompression」

Oracle DB インスタンスへのオプションの追加

Amazon RDS では、オプションは追加機能です。次に、OracleDB エンジンを実行している Amazon RDS インスタンスに追加できるオプションの説明を示します。

トピック

- Oracle DB オプションの概要
- 「Amazon S3 統合」
- Oracle Application Express (APEX)
- Amazon EFS の統合
- Oracle Java Virtual Machine
- Oracle Enterprise Manager
- Oracle Label Security
- Oracle Locator
- Oracle ネイティブネットワーク暗号化
- Oracle OLAP
- Oracle Secure Sockets Layer
- Oracle Spatial
- Oracle SQLT
- Oracle Statspack
- Oracle のタイムゾーン
- Oracle のタイムゾーンファイルの自動アップグレード
- Oracle Transparent Data Encryption
- Oracle UTL_MAIL
- Oracle XML DB

Oracle DB オプションの概要

Oracle Database のオプションを有効にするには、オプショングループにオプションを追加してから、そのオプショングループを DB インスタンスに関連付けます。(詳しくは、「オプショングループを使用する」を参照してください。)

トピック

- Oracle Database オプションの概要
- 異なるエディションでサポートされるオプション
- 特定のオプションのメモリ要件

Oracle Database オプションの概要

Oracle DB インスタンスには次のオプションを追加できます。

オプション	オプション ID
「Amazon S3 統合」	S3_INTEGRATION
Oracle Application Express (APEX)	APEX APEX-DEV
Oracle Enterprise Manager	OEM OEM_AGENT
Oracle Java Virtual Machine	JVM
Oracle Label Security	OLS
Oracle Locator	LOCATOR
Oracle ネイティブネットワーク暗号化	NATIVE_NETWORK_ENCRYPTION
Oracle OLAP	OLAP
Oracle Secure Sockets Layer	SSL
Oracle Spatial	SPATIAL
Oracle SQLT	SQLT
Oracle Statspack	STATSPACK
Oracle のタイムゾーン	Timezone

オプション	オプション ID
Oracle のタイムゾーンファイルの自動アップグレード	TIMEZONE_FILE_AUTOUPGRADE
Oracle Transparent Data Encryption	TDE
Oracle UTL_MAIL	UTL_MAIL
Oracle XML DB	XMLDB

異なるエディションでサポートされるオプション

RDS for Oracle では、サポートされていないエディションにオプションを追加できません。さまざまな Oracle Database エディションでサポートされている RDS オプションを確認するには、`aws rds describe-option-group-options` コマンドを使用します。次の例は、Oracle Database 19c エンタープライズエディションでサポートされているオプションを一覧表示しています。

```
aws rds describe-option-group-options \
    --engine-name oracle-ee \
    --major-engine-version 19
```

詳細については、AWS CLI コマンドリファレンスの describe-option-group-options を参照してください。

特定のオプションのメモリ要件

一部のオプションは、DB インスタンスで実行するために追加のメモリが必要です。例えば、Oracle Enterprise Manager Database Control には約 300 MB の RAM が使用されます。サイズの小さい DB インスタンスに対してこのオプションを有効にした場合は、パフォーマンスの問題やメモリ不足のエラーが発生することがあります。データベースの RAM 使用量を少なくするために、Oracle パラメータを調整できます。また、より大きな DB インスタンスにスケールアップすることもできます。

「Amazon S3 統合」

RDS for Oracle DB インスタンスと Amazon S3 バケットの間でファイルを転送することができます。Oracle Data Pump などの Oracle Database の機能と Amazon S3 統合を使用できます。例えば、Amazon S3 の Data Pump ファイルを RDS for Oracle DB インスタンスにダウンロードすることができます。詳細については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。

> **Note**
>
> DB インスタンスと Amazon S3 バケットは同じ AWS リージョン に存在する必要があります。

トピック

- Amazon S3 と RDS for Oracle を統合する IAM アクセス許可の設定
- Amazon S3 統合オプションの追加
- Amazon RDS for Oracle と Amazon S3 バケットの間でファイルを転送する
- Amazon S3 統合のトラブルシューティング
- Amazon S3 統合オプションの削除

Amazon S3 と RDS for Oracle を統合する IAM アクセス許可の設定

RDS for Oracle が Amazon S3 と統合するには、DB インスタンスが Amazon S3 バケットにアクセスできる必要があります。DB インスタンスで使用される Amazon VPC は Amazon S3 エンドポイントへのアクセスを提供する必要はありません。

RDS for Oracle は、あるアカウントの DB インスタンスと別のアカウントの Amazon S3 バケットの間のファイルの転送をサポートしています。追加のステップが必要な場合は、以下のセクションで説明します。

トピック

- ステップ 1: Amazon RDS ロール用の IAM ポリシーを作成する
- ステップ 2: (オプション) Amazon S3 バケットの IAM ポリシーを作成する
- ステップ 3: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする

- ステップ 4: IAM ロールを RDS for Oracle DB インスタンスに関連付ける

ステップ 1: Amazon RDS ロール用の IAM ポリシーを作成する

このステップでは、Amazon S3 バケットと RDS DB インスタンスの間のファイル転送に必要なアクセス許可を持つ AWS Identity and Access Management (IAM) ポリシーを作成します。このステップは、S3 バケットが既に作成されていることを前提としています。

ポリシーを作成する前に、次の情報を書き留めます。

- バケットの Amazon リソースネーム (ARN)
- お客様の AWS KMS キーの ARN、バケットが SSE-KMS または SSE-S3 暗号化を使用している場合

> **Note**
>
> RDS for Oracle DB インスタンスは、SSE-C で暗号化された Amazon S3 バケットにアクセスできません。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「サーバー側の暗号化を使用したデータの保護」を参照してください。

コンソール

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ポリシーを作成するには

1. IAM マネジメントコンソールを開きます。
2. [Access management (アクセス管理)] で、[Policies (ポリシー)] を選択します。
3. [ポリシーを作成] を選択します。
4. [Visual editor] タブで、[Choose a service] を選択し、[S3] を選択します。
5. [Actions (アクション)] で [Expand all (すべて展開)] を選択し、Amazon S3 バケットから Amazon RDS にファイルを転送するために必要なバケットのアクセス許可とオブジェクトのアクセス許可を選択します。例えば、以下を実行してください。
   - [List (リスト)] を展開し、[ListBucket] を選択します。
   - [Read (読み取り)] を展開し、[GetObject] を選択します。

- [Write (書き込み)] を展開し、[PutObject]、[DeleteObject]、[AbortMultipartUpload]、[ListMultipartUploadParts] を選択します。マルチパートアップロードのアクセス許可は、大きなファイル (100 MB 以上) を Amazon S3 にアップロードするときに必要です。
- [Permissions management] (アクセス許可の管理) を展開し、PutObjectAcl を選択します。このアクセス許可は、別のアカウントが所有するバケットにファイルをアップロードする予定で、このアカウントがバケットの内容を完全に制御する必要がある場合に必要です。

オブジェクトのアクセス許可は、Amazon S3 のオブジェクトオペレーションのアクセス許可です。バケット自体ではなくバケット内のオブジェクトに付与する必要があります。詳細については、「オブジェクトオペレーションに対するアクセス許可」を参照してください。

6. [リソース] を選択し、次の操作を行います。

   a. [特定] を選択します。

   b. [バケット] では、[ARN を追加] を選択します。バケット ARN を入力します。バケット名は自動的に入力されます。その後、[Add] (追加) を選択します。

   c. [オブジェクト] リソースが表示されている場合は、[ARN を追加] を選択してリソースを手動で追加するか、[すべて] を選択します。

   > **Note**
   >
   > Amazon S3 バケット内の特定のファイルやフォルダにのみアクセスすることを Amazon RDS に許可するには、[Amazon リソースネーム (ARN)] に、より具体的な ARN 値を設定します。Amazon S3 のアクセスポリシーの定義方法については、「Amazon S3 リソースへのアクセス許可の管理」を参照してください。

7. (オプション) ポリシーにリソースを追加するには、[Add addition permissions (アクセス許可を追加する)] を選択します。例えば、以下を実行してください。

   a. バケットがカスタム KMS キーで暗号化されている場合は、サービスの KMS を選択します。

   b. [手動アクション] では、以下を選択します。

      - 暗号化
      - [ReEncrypt from] および [ReEncrypt to]
      - Decrypt

- DescribeKey
- GenerateDataKey

c. [リソース] で [特定] を選択します。

d. [キー] では、[ARN を追加] を選択します。リソースとしてカスタムキーの ARN を入力し、[追加] を選択します。

詳細については、Amazon Simple Storage Service ユーザーガイドの「Protecting Data Using Server-Side Encryption with KMS keys Stored in AWS Key Management Service (SSE-KMS)」を参照してください。

e. Amazon RDS が他のバケットにアクセスするためにアクセスが必要な場合は、これらのバケットの ARN を追加します。オプションで、Amazon S3 内のすべてのバケットとオブジェクトへのアクセスを許可できます。

8. [次へ: タグ]、[次へ: レビュー] の順に選択します。
9. [名前] に、IAM ポリシーの名前 (例: `rds-s3-integration-policy`) を設定します。この名前は、IAM ロールを作成して DB インスタンスに関連付ける際に使用します。オプションで [Description] 値を追加することもできます。
10. [Create policy] (ポリシーの作成) を選択します。

AWS CLI

Amazon S3 バケットに Amazon RDS へのアクセス許可を付与する AWS Identity and Access Management IAM ポリシーを作成します。ポリシーを作成したら、ポリシーの ARN を書き留めます。この ARN は、後のステップで必要になります。

必要なアクセスのタイプに基づき、適切なアクションをポリシーに含めます。

- `GetObject` - Amazon S3 バケットから Amazon RDS へのファイルの転送に必要。
- `ListBucket` - Amazon S3 バケットから Amazon RDS へのファイルの転送に必要。
- `PutObject` - Amazon RDS から Amazon S3 バケットへのファイルの転送に必要。
- `AbortMultipartUpload` – マルチパートアップロードで、大きなファイル (100 MB 以上) を Amazon RDS から Amazon S3 バケットに転送するときに必要です。
- `ListMultipartUploadParts` – マルチパートアップロードで、大きなファイル (100 MB 以上) を Amazon RDS から Amazon S3 バケットに転送するときに必要です。

以下の AWS CLI コマンドでは、これらのオプションを指定して、*rds-s3-integration-policy* という名前の IAM ポリシーを作成します。このポリシーでは、*amzn-s3-demo-bucket* という名前のバケットへのアクセス権が付与されます。

Example

Linux、macOS、Unix の場合:

```
aws iam create-policy \
   --policy-name rds-s3-integration-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject",
           "s3:AbortMultipartUpload",
           "s3:ListMultipartUploadParts"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

次の例には、カスタム KMS キーのアクセス許可が含まれています。

```
aws iam create-policy \
   --policy-name rds-s3-integration-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
```

```
            "s3:PutObject",
            "kms:Decrypt",
            "kms:Encrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey",
            "kms:DescribeKey"
          ],
          "Effect": "Allow",
          "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "arn:aws:kms:::your-kms-arn"
          ]
        }
      ]
    }'
```

Windows の場合:

```
aws iam create-policy ^
    --policy-name rds-s3-integration-policy ^
    --policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "s3integration",
          "Action": [
            "s3:GetObject",
            "s3:ListBucket",
            "s3:PutObject",
            "s3:AbortMultipartUpload",
            "s3:ListMultipartUploadParts"
          ],
          "Effect": "Allow",
          "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
          ]
        }
      ]
    }'
```

次の例には、カスタム KMS キーのアクセス許可が含まれています。

```
aws iam create-policy ^
   --policy-name rds-s3-integration-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3integration",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket",
           "s3:PutObject",
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:ReEncrypt*",
           "kms:GenerateDataKey",
           "kms:DescribeKey"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*",
           "arn:aws:kms:::your-kms-arn"
         ]
       }
     ]
   }'
```

ステップ 2: (オプション) Amazon S3 バケットの IAM ポリシーを作成する

このステップは、次の条件でのみ必要です。

- あるアカウント (アカウント A) から Amazon S3 バケットにファイルをアップロードし、別のアカウント (アカウント B) からファイルにアクセスする予定です。
- アカウント B がバケットを所有しています。
- アカウント B は、バケットにロードされたオブジェクトを完全に制御する必要があります。

上記の条件が適用されない場合は、ステップ 3: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする に進みます。

バケットポリシーを作成するために、以下のものがあることを確認します。

- アカウント A のアカウント ID
- アカウント A のユーザー名
- アカウント B の Amazon S3 バケットの ARN 値

コンソール

バケットポリシーを作成または編集するには

1. AWS マネジメントコンソール にサインインし、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [Buckets (バケット)] リストで、バケットポリシーを作成するバケットの名前、またはバケットポリシーを編集するバケットの名前を選択します。
3. [Permissions (アクセス許可)] を選択します。
4. [バケットポリシー] で [編集] を選択します。バケットポリシーの編集ページが開きます。
5. [バケットポリシーを編集] ページで、「Amazon S3 ユーザーガイド」の [ポリシーの例] を確認するか、[ポリシージェネレーター] を選択してポリシーを自動的に生成するか、[ポリシー] セクションで JSON を編集します。

   ポリシージェネレーターを選択すると、AWS ポリシージェネレーターが新しいウィンドウで開きます。

   a. ポリシーの種類の選択の AWS ポリシーポリシージェネレータページで、 S3 バケットポリシーを選択します。

   b. 提供されたフィールドに情報を入力してステートメントを追加し、ステートメントの追加を選択します。追加するステートメントの数だけ繰り返します。ポリシーステートメントの詳細については、IAM ユーザーガイドの IAM JSON ポリシーのエレメントのリファレンスを参照してください。

      > **Note**
      >
      > 便宜上、バケットポリシーの編集ページでは、現在のバケットのバケット ARN (Amazonリソース名) がポリシーテキストフィールドの上に表示されます。この ARN をコピーして、AWSポリシージェネレータのステートメントで使用できます。

   c. ステートメントの追加が完了したら、ポリシーの生成を選択します。

d. 生成されたポリシーテキストをコピーし、[閉じる] を選択すると、Amazon S3 コンソールのバケットポリシーの編集ページに戻ります。

6. 左 ポリシー ボックスで、既存のポリシーを編集するか、ポリシージェネレータからバケットポリシーを貼り付けます。ポリシーを保存する前に、セキュリティ警告、エラー、一般的な警告、および提案を解決してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePermissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/account-A-user"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket",
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      ]
    }
  ]
}
```

7. 変更の保存を選択すると、[バケット許可] のページへ戻ります。

ステップ 3: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする

このステップは、IAM ポリシーを ステップ 1: Amazon RDS ロール用の IAM ポリシーを作成する で作成したことを前提としています。このステップでは、RDS for Oracle DB インスタンスのロールを作成し、ロールにポリシーをアタッチします。

コンソール

Amazon S3 バケットへのアクセスを Amazon RDS に許可する IAM ロールを作成するには

1. IAM マネジメントコンソールを開きます。
2. ナビゲーションペインで Roles (ロール) を選択してください。
3. [ロールの作成] を選択してください。
4. [AWS サービス] を選択してください。
5. [他の AWS サービスのユースケース] で、[RDS] を選択し、次に [RDS — データベースにロールを追加] を選択します。次に、[次へ] を選択します。
6. [アクセス許可ポリシー] の [検索] に、ステップ 1: Amazon RDS ロール用の IAM ポリシーを作成する で作成した IAM ポリシーの名前を入力し、リストに表示されたポリシーを選択します。次に、[次へ] を選択します。
7. [ロール名] に、`rds-s3-integration-role` などの IAM ロール名を入力します。オプションで [Description] 値を追加することもできます。
8. [ロールの作成] を選択してください。

AWS CLI

ロールを作成してポリシーをアタッチするには

1. Amazon S3 バケットにアクセスするには、お客様に代わって Amazon RDS が引き受けることのできる IAM ロールを作成します。

   リソースベースの信頼関係では `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用して、サービスに付与する特定のリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

   両方のグローバル条件コンテキストキーを使用し、`aws:SourceArn` 値にアカウント ID を含めます。この場合は、`aws:SourceAccount` 値と `aws:SourceArn` 値のアカウントは、同じステートメントで使用する場合、同じアカウント ID を使用する必要があります。

   - 単一リソースに対するクロスサービスアクセスが必要な場合は `aws:SourceArn` を使用します。
   - そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、`aws:SourceAccount`を使用します。

信頼関係では、aws:SourceArn グローバル条件コンテキストキーに、必ず、ロールにアクセスするリソースの完全な Amazon リソースネーム (ARN) を使用します。

次の AWS CLI コマンドでは、この目的で *rds-s3-integration-role* という名前のロールを作成します。

Example

Linux、macOS、Unix の場合:

```
aws iam create-role \
   --role-name rds-s3-integration-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                 "aws:SourceAccount": "my_account_ID",
                 "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
             }
         }
       }
     ]
   }'
```

Windows の場合:

```
aws iam create-role ^
   --role-name rds-s3-integration-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
```

```
            "Service": "rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "my_account_ID",
                "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
            }
        }
      }
    ]
  }'
```

詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

2. ロールが作成されたら、このロールの ARN を書き留めます。この ARN は、後のステップで必要になります。
3. 作成したポリシーを、作成したロールにアタッチします。

   以下の AWS CLI コマンドでは、*rds-s3-integration-role* という名前のロールにこのポリシーをアタッチします。

   Example

   Linux、macOS、Unix の場合:

   ```
   aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-integration-role
   ```

   Windows の場合:

   ```
   aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-integration-role
   ```

   your-policy-arn を、以前のステップで書き留めたポリシー ARN に置き換えます。

ステップ 4: IAM ロールを RDS for Oracle DB インスタンスに関連付ける

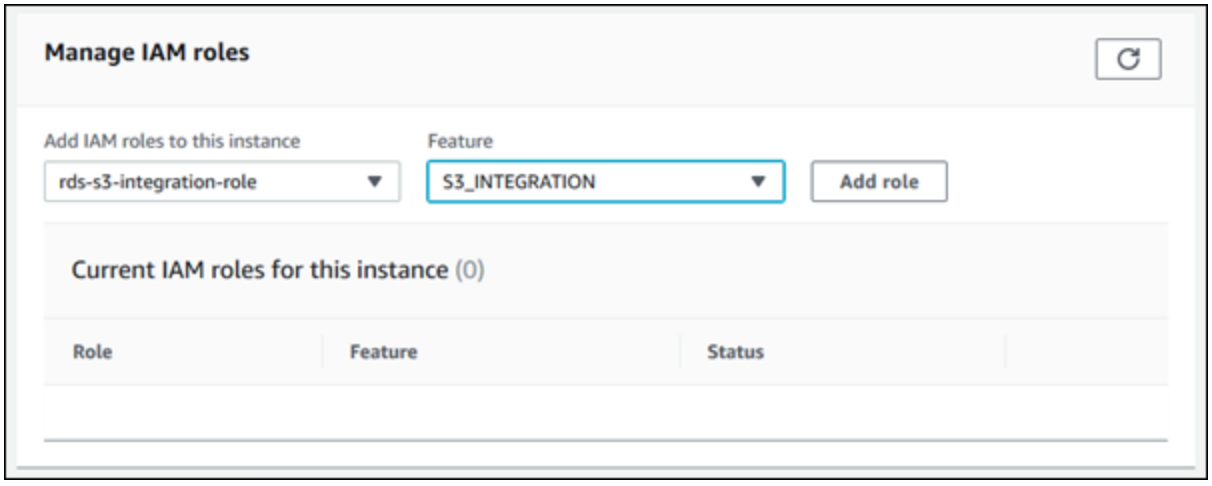
Amazon S3 統合のアクセス許可を設定する最後のステップは、IAM ロールを DB インスタンスに関連付けることです。次の要件に注意してください。

- 必須の Amazon S3 アクセス許可ポリシーがアタッチされた IAM ロールへのアクセスが許可されている必要があります。
- RDS for Oracle DB インスタンスには、一度に 1 つの IAM ロールのみを関連付けることができます。
- DB インスタンスは、[使用可能] の状態である必要があります。

コンソール

IAM ロールを RDS for Oracle DB インスタンスに関連付けるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインから [Databases (データベース)] を選択します。
3. 詳細を表示する RDS for Oracle DB インスタンスの名前を選択します。
4. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、次のいずれかを実行します。
5. [このインスタンスに IAM ロールを追加] で、ステップ 3: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする で作成したロールを選択します。
6. [機能] で、[S3_INTEGRATION] を選択します。



7. [Add role] を選択します。

AWS CLI

以下の AWS CLI コマンドでは、*mydbinstance* という名前の Oracle DB インスタンスにこのロールを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier mydbinstance \
   --feature-name S3_INTEGRATION \
   --role-arn your-role-arn
```

Windows の場合:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier mydbinstance ^
   --feature-name S3_INTEGRATION ^
   --role-arn your-role-arn
```

your-role-arn を、以前のステップで書き留めたロール ARN に置き換えます。`S3_INTEGRATION` オプションには `--feature-name` が指定されている必要があります。

Amazon S3 統合オプションの追加

Amazon RDS for Oracle と Amazon S3 を統合するには、DB インスタンスが `S3_INTEGRATION` オプションを含むオプショングループに関連付けられている必要があります。

コンソール

Amazon S3 統合用のオプショングループを設定するには

1. 新しいオプショングループを作成するか、`S3_INTEGRATION` オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [S3_INTEGRATION] オプションを追加します。

   オプショングループへのオプションの追加の詳細については、「オプショングループにオプションを追加する」を参照してください。

3. 新しい RDS for Oracle DB インスタンスを作成し、オプショングループをそのインスタンスに関連付けるか、オプショングループを関連付けるように RDS for Oracle DB インスタンスを変更します。

   DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

   DB インスタンスの変更については、Amazon RDS DB インスタンスを変更するを参照してください。

AWS CLI

Amazon S3 統合用のオプショングループを設定するには

1. 新しいオプショングループを作成するか、S3_INTEGRATION オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [S3_INTEGRATION] オプションを追加します。

   例えば、以下の AWS CLI コマンドでは、S3_INTEGRATION オプションを、**myoptiongroup** という名前のオプショングループに追加します。

   Example

   Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
   --option-group-name myoptiongroup \
   --options OptionName=S3_INTEGRATION,OptionVersion=1.0
```

   Windows の場合:

```
aws rds add-option-to-option-group ^
   --option-group-name myoptiongroup ^
   --options OptionName=S3_INTEGRATION,OptionVersion=1.0
```

3. 新しい RDS for Oracle DB インスタンスを作成し、オプショングループをそのインスタンスに関連付けるか、オプショングループを関連付けるように RDS for Oracle DB インスタンスを変更します。

   DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

   RDS for Oracle DB インスタンスの変更方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS for Oracle と Amazon S3 バケットの間でファイルを転送する

RDS for Oracle DB インスタンスと Amazon S3 バケット間でファイルを転送するには、Amazon RDS パッケージ `rdsadmin_s3_tasks` を使用できます。アップロード時に GZIP でファイルを圧縮し、ダウンロード時に解凍することができます。

トピック

- ファイル転送の要件と制限
- RDS for Oracle DB インスタンスから Amazon S3 バケットにファイルをアップロードする
- Amazon S3 バケットから Oracle DB インスタンスにファイルをダウンロードする
- ファイル転送のステータスをモニタリングする

ファイル転送の要件と制限

DB インスタンスと Amazon S3 バケットの間でファイルを転送する前に、次の点に注意してください。

- `rdsadmin_s3_tasks` パッケージは、単一のディレクトリにあるファイルを転送します。転送にサブディレクトリを含めることはできません。
- Amazon S3 バケット内の最大オブジェクトサイズは 5 TB です。
- `rdsadmin_s3_tasks` によって作成されたタスクは非同期的に実行されます。
- `DATA_PUMP_DIR` などの Data Pump ディレクトリ、またはユーザーが作成したディレクトリからファイルをアップロードできます。`adump`、`bdump`、または `trace` ディレクトリなど、Oracle バックグラウンドプロセスで使用されるディレクトリからファイルをアップロードすることはできません。

- ダウンロードできるファイル数は、download_from_s3 の場合、プロシージャコールあたり 2000 ファイルに制限されています。Amazon S3 から 2000 を超えるファイルをダウンロードする必要がある場合は、ダウンロードを別々のアクションに分割し、1 回のプロシージャコールあたりのファイル数が 2000 を超えないようにしてください。
- ダウンロードフォルダにファイルが存在し、同じ名前のファイルをダウンロードしようとすると、download_from_s3 ではダウンロードをスキップします。ダウンロードディレクトリからファイルを削除するには、PL/SQL プロシージャ UTL_FILE.FREMOVE を使用します。

RDS for Oracle DB インスタンスから Amazon S3 バケットにファイルをアップロードする

DB インスタンスから Amazon S3 バケットにファイルをアップロードするには、プロシージャ rdsadmin.rdsadmin_s3_tasks.upload_to_s3 を使用します。例えば、Oracle Recovery Manager (RMAN) のバックアップファイルまたは Oracle Data Pump ファイルをアップロードすることができます。オブジェクトの操作方法の詳細については、Amazon Simple Storage Service ユーザーガイドを参照してください。RMAN バックアップの詳細については、「Oracle DB インスタンスの一般的な RMAN タスクの実行」を参照してください。

rdsadmin.rdsadmin_s3_tasks.upload_to_s3 プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_bucket_name	VARCHAR2	–	必須	ファイルをアップロードする Amazon S3 バケットの名前。
p_directory_name	VARCHAR2	–	必須	ファイルのアップロード元の Oracle ディレクトリオブジェクトの名前。ディレクトリは、ユーザーが作成した任意のディレクトリオブジェクト、または Data Pump ディレクトリ (例: DATA_PUMP_DIR) です。adump、bdump、または trace など、バック

パラメータ名	データ型	デフォルト	必須	説明
				グラウンドプロセスで使用されるディレクトリからファイルをアップロードすることはできません。 **Note** ファイルは、指定したディレクトリからのみアップロードできます。指定したディレクトリのサブディレクトリにファイルをアップロードすることはできません。

パラメータ名	データ型	デフォルト	必須	説明
p_s3_prefix	VARCHAR2	–	必須	ファイルをアップロードする Amazon S3 ファイル名のプレフィックス。プレフィックスが空の場合は、指定された Amazon S3 バケットの最上位レベルにすべてのファイルがアップロードされますが、そのファイル名にプレフィックスは付加されません。 例えば、プレフィックスが folder_1/oradb の場合、ファイルは folder_1 にアップロードされます。この場合、ファイルにはそれぞれ oradb プレフィックスが追加されます。
p_prefix	VARCHAR2	–	必須	アップロードするファイル名が一致する必要があるファイル名のプレフィックス。プレフィックスが空の場合は、指定されたディレクトリ内のファイルがすべてアップロードされます。

パラメータ名	データ型	デフォルト	必須	説明
p_compression_level	NUMBER	0	オプション	GZIP 圧縮のレベル。有効な値の範囲は 0～9 です。 • 0 – 圧縮なし • 1 – 最速圧縮 • 9 – 最高圧縮
p_bucket_owner_full_control	VARCHAR2	–	optional	バケットのアクセスコントロールリスト。有効な値は、null と FULL_CONTROL のみです。この設定は、あるアカウント (アカウント A) から別のアカウント (アカウント B) が所有するバケットにファイルをアップロードし、アカウント B がファイルを完全に制御する必要がある場合にのみ必要です。

rdsadmin.rdsadmin_s3_tasks.upload_to_s3 プロシージャの戻り値はタスク ID です。

次の例では、*DATA_PUMP_DIR* ディレクトリ内のすべてのファイルを *amzn-s3-demo-bucket* という名前の Amazon S3 バケットにアップロードします。ファイルは圧縮されません。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_prefix         =>  '',
      p_s3_prefix      =>  '',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

次の例では、*db* ディレクトリ内のプレフィックス *DATA_PUMP_DIR* のついたすべてのファイルを、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットにアップロードします。Amazon RDS は、最高レベルの GZIP 圧縮をファイルに適用します。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name       =>  'amzn-s3-demo-bucket',
      p_prefix            =>  'db',
      p_s3_prefix         =>  '',
      p_directory_name    =>  'DATA_PUMP_DIR',
      p_compression_level =>  9)
   AS TASK_ID FROM DUAL;
```

次の例では、*DATA_PUMP_DIR* ディレクトリのすべてのファイルを、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットにアップロードします。ファイルは、dbfiles フォルダにアップロードされます。この例では、GZIP 圧縮レベルは *1* です。これは最も速い圧縮レベルです。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name       =>  'amzn-s3-demo-bucket',
      p_prefix            =>  '',
      p_s3_prefix         =>  'dbfiles/',
      p_directory_name    =>  'DATA_PUMP_DIR',
      p_compression_level =>  1)
   AS TASK_ID FROM DUAL;
```

次の例では、*DATA_PUMP_DIR* ディレクトリのすべてのファイルを、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットにアップロードします。ファイルは dbfiles フォルダにアップロードされ、各ファイル名の先頭には ora が付加されます。圧縮は適用されません。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_prefix         =>  '',
      p_s3_prefix      =>  'dbfiles/ora',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

次の例では、コマンドはアカウント A で実行されるものの、アカウント B でバケットの内容を完全に制御する必要があることを前提としています。コマンド rdsadmin_s3_tasks.upload_to_s3 は、*s3bucketOwnedByAccountB* という名前のバケットに *DATA_PUMP_DIR* ディレクトリ内のすべてのファイルを転送します。アクセスコントロールは、アカウント B がバケット内のファイル

にアクセスできるように、FULL_CONTROL に設定されています。GZIP 圧縮レベルは *6* で、速度とファイルサイズのバランスが取れています。

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_to_s3(
      p_bucket_name               =>  's3bucketOwnedByAccountB',
      p_prefix                    =>  '',
      p_s3_prefix                 =>  '',
      p_directory_name            =>  'DATA_PUMP_DIR',
      p_bucket_owner_full_control =>  'FULL_CONTROL',
      p_compression_level         =>  6)
   AS TASK_ID FROM DUAL;
```

それぞれの例では、SELECT ステートメントでは、タスクの ID が VARCHAR2 データ型で返ります。

タスクの出力ファイルを表示すると、結果を確認できます。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

task-id は、この手順で返されたタスク ID に置き換えます。

> **Note**
>
> タスクは非同期的に実行されます。

Amazon S3 バケットから Oracle DB インスタンスにファイルをダウンロードする

Amazon S3 バケットから RDS for Oracle インスタンスにファイルをダウンロードするには、Amazon RDS プロシージャ (rdsadmin.rdsadmin_s3_tasks.download_from_s3) を使用します。

download_from_s3 プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_bucket_name	VARCHAI	–	必須	ファイルのダウンロード元の Amazon S3 バケットの名前。

パラメータ名	データ型	デフォルト	必須	説明
p_directory_name	VARCHAI	–	必須	ファイルをダウンロードする Oracle ディレクトリオブジェクトの名前。ディレクトリは、ユーザーが作成した任意のディレクトリオブジェクト、または Data Pump ディレクトリ (例: DATA_PUMP_DIR) です。
p_error_on_zero_downloads	VARCHAI	誤	オプションです。	Amazon S3 バケット内のどのオブジェクトもプレフィックスと一致しない場合に、タスクによってエラーを発生させるかどうかを決定するフラグ。このパラメータが設定されていないか、FALSE (デフォルト) に設定されている場合、タスクはオブジェクトが見つからなかったというメッセージを出力しますが、例外が発生したり、エラーになったりすることはありません。このパラメータが TRUE の場合、タスクでは例外が発生し、エラーになります。 一致テストでエラーになる可能性のあるプレフィックス仕様の例としては、'import/test9.log' のようにプレフィックスにスペースが入るもの、または test9.log や test9.LOG のように大文字と小文字が一致しないものがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_s3_prefix	VARCHAI	–	必須	ダウンロードするファイル名が一致する必要があるファイル名のプレフィックス。プレフィックスを空にすると、指定された Amazon S3 バケット内の最上位ファイルがすべてダウンロードされますが、バケットのフォルダ内のファイルはダウンロードされません。 この手順では、プレフィックスと一致する初期のレベルのフォルダからのみ、Amazon S3 オブジェクトをダウンロードします。指定されたプレフィックスに一致する多層のディレクトリはダウンロードされません。 例えば、Amazon S3 バケットのフォルダ構造が `folder_1/folder_2/folder_3` とします。`'folder_1/folder_2/'` プレフィックスを指定します。この場合、`folder_2` のファイルのみダウンロードされ、`folder_1` または `folder_3` のファイルはダウンロードされません。 その代わりに、`'folder_1/folder_2'` を指定する場合、`folder_1` プレフィックスに一致する、`'folder_2'` のファイルはすべてダウンロードされますが、`folder_2` のファイルはダウンロードされません。

パラメータ名	データ型	デフォルト	必須	説明
p_decompression_format	VARCHAR	–	オプションです。	解凍形式。有効な値は、解凍しない場合は NONE、解凍する場合は GZIP です。

rdsadmin.rdsadmin_s3_tasks.download_from_s3 プロシージャの戻り値はタスク ID です。

次の例では、*amzn-s3-demo-bucket* という名前の Amazon S3 バケットのファイルをすべて、*DATA_PUMP_DIR* ディレクトリにダウンロードします。ファイルは圧縮されていないため、解凍は適用されません。

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

以下の例では、*db* という名前の Amazon S3 バケット内にある、プレフィックス *amzn-s3-demo-bucket* がついたファイルをすべて、*DATA_PUMP_DIR* ディレクトリにダウンロードします。ファイルは GZIP で圧縮されているため、解凍が適用されます。p_error_on_zero_downloads パラメータはプレフィックスエラーチェックを有効にするため、プレフィックスがバケット内のどのファイルとも一致しない場合、タスクでは例外が発生し、エラーになります。

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name               =>  'amzn-s3-demo-bucket',
      p_s3_prefix                 =>  'db',
      p_directory_name            =>  'DATA_PUMP_DIR',
      p_decompression_format      =>  'GZIP',
      p_error_on_zero_downloads   =>  'TRUE')
   AS TASK_ID FROM DUAL;
```

以下の例では、*myfolder/* という名前の Amazon S3 バケット内にある *amzn-s3-demo-bucket* フォルダ内のファイルをすべて、*DATA_PUMP_DIR* ディレクトリにダウンロードします。Amazon S3 フォルダを指定するには、p_s3_prefix パラメータを使用します。アップロードされたファイルは GZIP で圧縮されますが、ダウンロード中は解凍されません。

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
```

```
      p_bucket_name          =>  'amzn-s3-demo-bucket',
      p_s3_prefix            =>  'myfolder/',
      p_directory_name       =>  'DATA_PUMP_DIR',
      p_decompression_format =>  'NONE')
   AS TASK_ID FROM DUAL;
```

次の例では、*DATA_PUMP_DIR* ディレクトリに *amzn-s3-demo-bucket* という名前の Amazon S3 バケット内のファイル *mydumpfile.dmp* をダウンロードします。解凍は適用されません。

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
      p_bucket_name    =>  'amzn-s3-demo-bucket',
      p_s3_prefix      =>  'mydumpfile.dmp',
      p_directory_name =>  'DATA_PUMP_DIR')
   AS TASK_ID FROM DUAL;
```

それぞれの例では、SELECT ステートメントでは、タスクの ID が VARCHAR2 データ型で返ります。

タスクの出力ファイルを表示すると、結果を確認できます。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

task-id は、この手順で返されたタスク ID に置き換えます。

> **Note**
>
> タスクは非同期的に実行されます。
> ディレクトリからファイルを削除するには、Oracle プロシージャ UTL_FILE.FREMOVE を使用できます。詳細については、Oracle ドキュメントの「FREMOVE プロシージャ」を参照してください。

ファイル転送のステータスをモニタリングする

ファイル転送タスクでは、スタート時と完了時に Amazon RDS イベントが発行されます。イベントメッセージには、ファイル転送のタスク ID が含まれています。イベントの表示の詳細については、「Amazon RDS イベントの表示」を参照してください。

実行中のタスクのステータスは bdump ファイルで確認できます。bdump ファイルは /rdsdbdata/log/trace ディレクトリにあります。bdump ファイルの名前形式は、以下のとおりです。

```
dbtask-task-id.log
```

task-id を、モニタリングするタスクの ID に置き換えます。

> **Note**
>
> タスクは非同期的に実行されます。

bdump ファイルの中身を表示するには、rdsadmin.rds_file_util.read_text_file ストアドプロシージャを使用します。例えば、次のクエリでは、bdump ファイル (*dbtask-1234567890123-1234.log*) の中身が返ります。

```
SELECT text FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1234567890123-1234.log'));
```

次のサンプルは、転送が失敗したときのログファイルを示しています。

```
TASK_ID




                                                
```

```
------------------------------------------------------------------------------------------
2023-04-17 18:21:33.993 UTC [INFO ] File #1: Uploading the file /rdsdbdata/datapump/
A123B4CDEF567890G1234567890H1234/sample.dmp to Amazon S3 with bucket name amzn-s3-demo-
bucket and key sample.dmp.
2023-04-17 18:21:34.188 UTC [ERROR] RDS doesn't have permission to write to Amazon S3
 bucket name amzn-s3-demo-bucket and key sample.dmp.
2023-04-17 18:21:34.189 UTC [INFO ] The task failed.
```

Amazon S3 統合のトラブルシューティング

トラブルシューティングのヒントについては、AWS re:Post の記事「Amazon RDS for Oracle と Amazon S3 を統合する際の問題をトラブルシューティングするにはどうすればよいですか?」を参照してください。

Amazon S3 統合オプションの削除

> **Important**
>
> S3_INTEGRATION オプションを削除すると、`rdsadmin.rdsadmin_s3_tasks` パッケージが無効になります。このオプションを削除する前に、すべての S3 転送が完了していることを確認してください。

DB インスタンスから Amazon S3 統合オプションを削除できます。

Amazon S3 統合オプションを DB インスタンスから削除するには、次のいずれかを実行します。

- 複数の DB インスタンスから Amazon S3 統合オプションを削除するには、DB インスタンスが属しているオプショングループから `S3_INTEGRATION` オプションを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- 1 つの DB インスタンスから Amazon S3 統合オプションを削除するには、インスタンスを変更し、`S3_INTEGRATION` オプションを含まない別のオプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Application Express (APEX)

Amazon RDS では、APEX および APEX-DEV オプションを使用した Oracle Application Express (APEX) がサポートされています。Oracle APEX は、ランタイム環境として、あるいはウェブベースのアプリケーション用の完全開発環境としてデプロイできます。Oracle APEX を使用して、ウェブブラウザ内でアプリケーション全体を構築できます。詳細については、Oracle ドキュメントの Oracle Application Express を参照してください。

トピック

- Oracle APEX コンポーネント
- 要件と制限
- Oracle APEX と Oracle Rest Data Services (ORDS) のセットアップ
- Oracle Rest Data Services (ORDS) の設定
- Oracle APEX のアップグレードと削除

Oracle APEX コンポーネント

Oracle APEX は、次の主要コンポーネントで構成されています。

- Oracle APEX アプリケーションおよびコンポーネントのメタデータを格納するリポジトリ。リポジトリは、テーブル、インデックス、および Amazon RDS DB インスタンスにインストールされている他のオブジェクトで構成されます。
- Oracle APEX クライアントとの HTTP 通信を管理するリスナー。リスナーは別個のホスト (Amazon EC2 インスタンス、社内のオンプレミスサーバー、またはデスクトップコンピュータなど) に配置します。リスナーは、ウェブブラウザからの受信接続を受け入れ、処理するためにそれらを Amazon RDS DB インスタンスに転送した後、リポジトリからの結果をブラウザに戻します。

  Amazon RDS for Oracle は、次のタイプのリスナーをサポートしています。

  - Oracle APEX バージョン 5.0 以降では、Oracle REST Data Services (ORDS) バージョン 19.1 以上を使用します。サポートされている最新バージョンの Oracle APEX および ORDS を使用することをお勧めします。このドキュメントでは、下位互換性のためだけに古いバージョンについて説明しています。
  - Oracle APEX バージョン 4.1.1 では、Oracle APEX リスナーのバージョン 1.1.4 を使用できます。
  - Oracle HTTP Server や mod_plsql リスナーを使用できます。

> **Note**
>
> Amazon RDS では、埋込み PL/SQL ゲートウェイの Oracle XML DB HTTP サーバーを Oracle APEX のリスナーとしてサポートしません。一般的に、Oracle では、インターネット上で稼働するアプリケーションで埋め込みの PL/SQL ゲートウェイを使用しないことが推奨されています。

詳細については、Oracle ドキュメントの「Web リスナーの選択について」を参照してください。

RDS for Oracle DB インスタンスに APEX オプションと APEX-DEV オプションを追加すると、Amazon RDS は Oracle APEX リポジトリのみをインストールします。リスナーを別のホストにインストールします。

要件と制限

このトピックでは、Oracle APEX と ORDS の要件と制限を一覧表示します。

Oracle APEX バージョンの要件

APEX オプションは、DB インスタンスの DB インスタンスクラスでストレージを使用します。以下に、Oracle APEX でサポートされているバージョンおよびおよそのストレージ要件を示します。

> **Note**
>
> Oracle Database 26ai では、Amazon RDS は Oracle APEX バージョン 24.1.v1 以降のみをサポートします。Oracle APEX の以前のバージョンは、Oracle Database 26ai ではサポートされていません。

Oracle APEX バージョン	ストレージの要件	サポートされている Oracle Database のバージョン	注意事項
Oracle APEX バージョン 26.1.v1	122 MiB	26ai、21c、および 19c	このバージョンには、パッチ 39179920: PSE BUNDLE FOR APEX 26.1 (PSES ON TOP OF

Oracle APEX バージョン	ストレージの要件	サポートされている Oracle Database のバージョン	注意事項
		(リリース更新 19.0.0.0.ru-2023-01.rur-2023-01.r1 以降)	26.1.0), PATCH_VERSION 2 が含まれています。EC2 インスタンスにインストールするのにまったく同じ APEX イメージバージョンが必要な場合は、パッチ 39743459: 26.1.2 PSE BUNDLE FOR APEX 26.1 (PSES ON TOP OF 26.1.0) をダウンロードします。
Oracle APEX バージョン 24.2.v1	114 MiB	すべて	このバージョンには、パッチ 37885097: PSE BUNDLE FOR APEX 24.2 (PSES ON TOP OF 24.2.0)、PATCH_VERSION 4 が含まれています。
Oracle APEX バージョン 24.1.v1	112 MiB	すべて	このバージョンには、パッチ 36695709: PSE BUNDLE FOR APEX 24.1 (PSES ON TOP OF 24.1.0), PATCH_VERSION 3 が含まれています。EC2 インスタンスにインストールするのにまったく同じ APEX イメージバージョンが必要な場合は、パッチ 37544819: 24.1.3 PSE BUNDLE FOR APEX 24.1 (PSES ON TOP OF 24.1.0) をダウンロードします。
Oracle APEX バージョン 23.2.v1	110 MiB	21c および 19c	このバージョンには、パッチ 35895964: PSE BUNDLE FOR APEX 23.2 (PSES ON TOP OF 23.2.0)、PATCH_VERSION 6 が含まれています。EC2 インスタンスにインストールするのにまったく同じ APEX イメージバージョンが必要な場合は、パッチ 37593125: 23.2.6 PSE BUNDLE FOR APEX 23.2 (PSES ON TOP OF 23.2.0) をダウンロードします。

Oracle APEX バージョン	ストレージの要件	サポートされている Oracle Database のバージョン	注意事項
Oracle APEX バージョン 23.1.v1	106 MiB	21c および 19c	このバージョンには、パッチ 35283657: PSE BUNDLE FOR APEX 23.1 (PSES ON TOP OF 23.1.0), PATCH_VERSION 2 が含まれています。
Oracle APEX バージョン 22.2.v1	106 MiB	21c および 19c	このバージョンには、パッチ 34628174: PSE BUNDLE FOR APEX 22.2 (PSES ON TOP OF 22.2.0)、PATCH_VERSION 4 が含まれています。
Oracle APEX バージョン 22.1.v1	124 MiB	21c および 19c	このバージョンには、パッチ 34020981: PSE BUNDLE FOR APEX 22.1 (PSES ON TOP OF 22.1.0)、PATCH_VERSION 6 が含まれています。
Oracle APEX バージョン 21.2.v1	125 MiB	21c および 19c	このバージョンには、パッチ 33420059: PSE BUNDLE FOR APEX 21.2 (PSES ON TOP OF 21.2.0)、PATCH_VERSION 8 が含まれています。
Oracle APEX バージョン 21.1.v1	125 MiB	21c および 19c	このバージョンには、パッチ 32598392 (PSE BUNDLE FOR APEX 21.1、PATCH_VERSION 3) が含まれています。

Oracle APEX バージョン	ストレージの要件	サポートされている Oracle Database のバージョン	注意事項
Oracle APEX バージョン 20.2.v1	148 MiB	19c	このバージョンには、パッチ 32006852 (PSE BUNDLE FOR APEX 20.2、PATCH_VERSION 2020.11.12) が含まれています。以下のクエリを実行すると、パッチの番号と日付を確認できます。 `SELECT PATCH_VERSION, PATCH_NUMBER FROM APEX_PATCHES;`
Oracle APEX バージョン 20.1.v1	173 MiB	19c	このバージョンには、パッチ 30990551 (PSE BUNDLE FOR APEX 20.1, PATCH_VERSION 2020.07.15) が含まれています。
Oracle APEX バージョン 19.2.v1	149 MiB	19c	
Oracle APEX バージョン 19.1.v1	148 MiB	19c	

ダウンロード可能な Oracle APEX.zip ファイルについては、Oracle のウェブサイトで「Oracle APEX Prior Release Archives」を参照してください。

Oracle APEX および ORDS の前提条件

Oracle APEX と ORDS を使用する際、以下の前提条件に注意してください。

- システムは Java ランタイム環境 (JRE) を使用する必要があります。
- Oracle クライアントのインストールには次が含まれている必要があります。
  - 管理タスク用の SQL*Plus または SQL Developer
  - RDS for Oracle DB インスタンスへの接続を設定するための Oracle Net Services

Oracle APEX の制限事項

APEX_*version* ユーザーアカウントは、Amazon RDS によって管理され、ユーザーが変更することはできません。したがって、このユーザーにデータベースプロファイルを適用したり、パスワードルールを適用したりすることはできません。APEX_*version* のプロファイルとパスワードの設定は Oracle と AWS によって事前定義されており、Amazon RDS のセキュリティ要件を満たすように設計されています。

Oracle APEX と Oracle Rest Data Services (ORDS) のセットアップ

次のトピックでは、Oracle APEX と ORDS のセットアップに必要な手順を示します。

トピック

- APEX および APEX-DEV オプションの DB インスタンスへの追加
- DB インスタンスのパブリックユーザーアカウントのロック解除
- Oracle APEX の RESTful サービスの設定
- ORDS を別のホストにインストールする準備
- Oracle APEX リスナーの設定

APEX および APEX-DEV オプションの DB インスタンスへの追加

RDS for Oracle DB インスタンスに APEX および APEX-DEV オプションを追加するには、次の手順を実行します。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに APEX と APEX-DEV オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

APEX オプションと APEX-DEV オプションを追加すると、DB インスタンスが自動的に再起動する間に短時間の停止が発生します。

> **Note**
>
> APEX_MAIL は、APEX オプションがインストールされている場合に使用できます。APEX_MAIL パッケージの実行権限が PUBLIC に付与されるため、APEX 管理者アカウントを使用する必要はありません。

APEX オプションと APEX-DEV オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] で、使用する Oracle のエディションを選択します。APEX オプションと APEX-DEV オプションは、すべてのエディションでサポートされます。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループにオプションを追加します。Oracle APEX ランタイム環境のみをデプロイする場合には、APEX オプションのみを追加します。完全なデプロイ環境をデプロイするには、APEX と APEX-DEV の両方のオプションを追加します。

   [バージョン] で、使用する Oracle APEX のバージョンを選択します。

   > **Important**
   >
   > 1 つ以上の DB インスタンスに既にアタッチされている既存のオプショングループに APEX オプションまたは APEX-DEV オプションを追加すると、短時間の停止が発生します。この停止の間、すべての DB インスタンスが自動的に再起動されます。

   オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。

3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。既存の DB インスタンスに APEX オプションまたは APEX-DEV オプションを追加すると、DB インスタンスが自動的に再起動する間に短時間の停止が発生します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

DB インスタンスのパブリックユーザーアカウントのロック解除

APEX オプションまたは APEX-DEV オプションを DB インスタンスにインストールしたら、次の操作を必ず実行してください。

1. APEX_PUBLIC_USER アカウントのパスワードを変更します。
2. アカウントのロックを解除します。

Oracle SQL*Plus コマンドラインユーティリティを使用してこれを行うことができます。DB インスタンスにマスターユーザーとして接続し、次のコマンドを発行します。new_password を任意のパスワードに置き換えます。

```
ALTER USER APEX_PUBLIC_USER IDENTIFIED BY new_password;
ALTER USER APEX_PUBLIC_USER ACCOUNT UNLOCK;
```

Oracle APEX の RESTful サービスの設定

Oracle APEX で RESTful サービスを設定するには (APEX 4.1.1.V1 では必要ありません)、SQL*Plus を使用して、マスターユーザーとして DB インスタンスに接続します。接続したら、rdsadmin.rdsadmin_run_apex_rest_config ストアドプロシージャを実行します。ストアドプロシージャを実行するときは、次のユーザーにパスワードを提供します。

- APEX_LISTENER
- APEX_REST_PUBLIC_USER

ストアドプロシージャは apex_rest_config.sql スクリプトを実行し、そのスクリプトによりそれらのユーザーの新しいデータベースアカウントが作成されます。

> **Note**
>
> Oracle APEX バージョン 4.1.1.v1 に対する設定は不要です。この Oracle APEX バージョンに限り、ストアドプロシージャの実行は不要です。

次のコマンドはストアドプロシージャを実行します。

```
EXEC rdsadmin.rdsadmin_run_apex_rest_config('apex_listener_password',
 'apex_rest_public_user_password');
```

ORDS を別のホストにインストールする準備

別個のホスト (Amazon EC2 インスタンス、社内のオンプレミスサーバー、またはデスクトップコンピュータなど) にインストールします。このセクションの例では、ホストが Linux を実行し、myapexhost.example.com という名前が付けられていることを前提としています。

ORDS をインストールする前に、権限のない OS ユーザーを作成し、Oracle APEX インストールファイルをダウンロードして解凍する必要があります。

ORDS のインストールを準備するには

1. myapexhost.example.com として root にログインします。
2. リスナーのインストールを所有する権限を持っていない OS ユーザーを作成します。以下のコマンドでは、apexuser という名前の新規ユーザーを作成します。

   ```
   useradd -d /home/apexuser apexuser
   ```

   以下のコマンドは、新規ユーザーにパスワードを割り当てます。

   ```
   passwd apexuser;
   ```

3. myapexhost.example.com に apexuser としてログインし、Oracle APEX のインストールファイルを Oracle から /home/apexuser ディレクトリにダウンロードします。
   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - Oracle Application Express Prior Release Archives
4. ファイルを /home/apexuser ディレクトリに解凍します。

   ```
   unzip apex_version.zip
   ```

   ファイルは、apex ディレクトリ内に /home/apexuser ディレクトリとして解凍されます。
5. myapexhost.example.com に apexuser としてログインしている間に Oracle から Oracle REST Data Services ファイルを /home/apexuser ディレクトリにダウンロードします。http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html

Oracle APEX リスナーの設定

> **Note**
>
> Oracle APEX リスナーは非推奨です。

Amazon RDS for Oracle は、引き続き Oracle APEX バージョン 4.1.1 および Oracle APEX リスナーバージョン 1.1.4 をサポートします。サポートされている最新バージョンの Oracle APEX および ORDS を使用することをお勧めします。

Oracle APEX Listener は別個のホストにインストールします (Amazon EC2 インスタンス、社内のオンプレミスサーバー、またはデスクトップコンピュータ)。ここでは、ホスト名が `myapexhost.example.com` であり、ホストで Linux が実行されていることが前提です。

Oracle APEX リスナーのインストールの準備

Oracle APEX リスナーをインストールする前に、権限のない OS ユーザーを作成し、Oracle APEX インストールファイルをダウンロードして解凍する必要があります。

Oracle APEX リスナーのインストールを準備するには

1. `myapexhost.example.com` として `root` にログインします。
2. リスナーのインストールを所有する権限を持っていない OS ユーザーを作成します。以下のコマンドでは、apexuser という名前の新規ユーザーを作成します。

   ```
   useradd -d /home/apexuser apexuser
   ```

   以下のコマンドは、新規ユーザーにパスワードを割り当てます。

   ```
   passwd apexuser;
   ```

3. `myapexhost.example.com` に `apexuser` としてログインし、Oracle APEX のインストールファイルを Oracle から `/home/apexuser` ディレクトリにダウンロードします。
   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - Oracle Application Express Prior Release Archives
4. ファイルを `/home/apexuser` ディレクトリに解凍します。

```
unzip apex_<version>.zip
```

ファイルは、apex ディレクトリ内に /home/apexuser ディレクトリとして解凍されます。

5. myapexhost.example.com に apexuser としてログインしている間に、Oracle APEX Listener ファイルを Oracle から /home/apexuser ディレクトリにダウンロードします。

Oracle APEX リスナーのインストールと設定

Oracle APEX を使用する前に、apex.war ファイルをダウンロードし、Java を使用して Oracle APEX リスナーをインストールしてから、リスナーを起動する必要があります。

Oracle APEX リスナーをインストールして設定するには

1. Oracle APEX リスナーに基づいて新しいディレクトリを作成し、リスナーファイルを開きます。

   以下のコードを実行します。

   ```
   mkdir /home/apexuser/apexlistener
   cd /home/apexuser/apexlistener
   unzip ../apex_listener.version.zip
   ```

2. 以下の コードを実行します。

   ```
   java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase -
   jar ./apex.war
   ```

3. 次のプログラムプロンプトの情報を入力します。

   - APEX Listener Administrator のユーザー名。デフォルト値は adminlistener です。
   - APEX Listener Administrator のパスワード。
   - APEX リスナーマネージャのユーザー名。デフォルトは managerlistener です。
   - APEX Listener Administrator のパスワード。

   プログラムは、次のように設定を完了するために必要な URL を出力します。

   ```
   INFO: Please complete configuration at: http://localhost:8080/apex/
   listenerConfigure
   ```

```
Database is not yet configured
```

4. Oracle Application Express を使用できるように、Oracle APEX リスナーを実行したままにします。この設定手順を完了したら、リスナーをバックグラウンドで実行できます。
5. ウェブブラウザから、Oracle APEX リスナープログラムが提供する URL にアクセスします。Oracle Application Express Listener の管理ウィンドウが表示されます。次の情報を入力します。

   - Username - APEX_PUBLIC_USER
   - [Password] - APEX_PUBLIC_USER のパスワード このパスワードは、以前に Oracle APEX リポジトリの設定時に指定したパスワードです。詳細については、「DB インスタンスのパブリックユーザーアカウントのロック解除」を参照してください。
   - Connection Type - Basic
   - Hostname - Amazon RDS DB インスタンスのエンドポイント。mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com など。
   - [Port] - 1521
   - SID - Amazon RDS DB インスタンスのデータベースの名前。mydb など。
6. [Apply] を選択します。Oracle APEX の管理ウィンドウが表示されます。
7. Oracle APEX の admin ユーザーのパスワードを設定します。これを行うには、SQL*Plus を使用して DB インスタンスにマスターユーザーとして接続し、次のコマンドを実行します。

   ```
   EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
   grant APEX_ADMINISTRATOR_ROLE to master;
   @/home/apexuser/apex/apxchpwd.sql
   ```

   master を自身のマスターユーザー名に置き換えます。apxchpwd.sql スクリプトによってプロンプトが表示されたら、新しい admin パスワードを入力します。
8. ブラウザで Oracle APEX の管理ウィンドウに戻り、[管理] を選択します。次に、[Application Express Internal Administration] を選択します。認証情報を求められたら、以下の情報を入力します。

   - User name - admin
   - Password - apxchpwd.sql スクリプトを使用して設定したパスワード。

   [Login] を選択し、その admin ユーザーの新しいパスワードを設定します。

これで、リスナーを使用する準備ができました。

Oracle Rest Data Services (ORDS) の設定

次のトピックでは、ORDS 21 および 22 の設定オプションを一覧表示します。

トピック

- ORDS 21 以前のインストールと設定
- ORDS 22 以降のインストールと設定

ORDS 21 以前のインストールと設定

これで、Oracle APEX で使用する Oracle Rest Data Services (ORDS) をインストールして設定する準備が整いました。Oracle APEX バージョン 5.0 以降では、ORDS バージョン 19.1 ~ 21 を使用します。ORDS 22 以降をインストールする方法については、「ORDS 22 以降のインストールと設定」を参照してください。

リスナーは別個のホスト (Amazon EC2 インスタンス、社内のオンプレミスサーバー、またはデスクトップコンピュータなど) にインストールします。このセクションの例では、ホストの名前が `myapexhost.example.com` であり、ホストが Linux を実行していると仮定します。

Oracle APEX で使用する ORDS 21 以前をインストールして設定するには

1. Oracle REST data services に移動し、Readme を確認します。必要なバージョンの Java がインストールされていることを確認します。
2. ORDS インストール用の新しいディレクトリを作成します。

   ```
   mkdir /home/apexuser/ORDS
   cd /home/apexuser/ORDS
   ```

3. Oracle REST Data Services から、ファイル `ords.version.number.zip` をダウンロードします。
4. ファイルを `/home/apexuser/ORDS` ディレクトリに解凍します。
5. マルチテナントデータベースに ORDS をインストールする場合は、ファイル `/home/apexuser/ORDS/params/ords_params.properties` に次の行を追加します。

   ```
   pdb.disable.lockdown=false
   ```

6. マスターユーザーに ORDS のインストールに必要な権限を付与します。

Oracle APEX のオプションをインストールしたら、ORDS スキーマをインストールするために必要な権限をマスターユーザーに付与します。これを行うには、データベースに接続し、次のコマンドを実行します。*MASTER_USER* をマスターユーザーの (大文字で記述した) 名前に置き換えます。

> **Important**
>
> 大文字と小文字を区別する識別子を使用してユーザーを作成した場合を除き、ユーザー名を入力する際には大文字を使用します。例えば、CREATE USER myuser または CREATE USER MYUSER を実行すると、データディクショナリに MYUSER が保存されます。ただし、CREATE USER "MyUser" で二重引用符を使用すると、データディクショナリには MyUser が保存されます。詳細については、「SYS オブジェクトへの SELECT または EXECUTE 権限の付与」を参照してください。

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_ROLE_PRIVS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONS_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONSTRAINTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_PROCEDURES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TABLES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_VIEWS', 'MASTER_USER', 'SELECT',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('WPIUTL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SESSION', 'MASTER_USER',
 'EXECUTE', true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY', 'MASTER_USER',
 'EXECUTE', true);
```

> **Note**
>
> これらのコマンドは、ORDS バージョン 19.1 以降に適用されます。

7. ダウンロードした ords.war ファイルを使用して ORDS スキーマをインストールします。

```
java -jar ords.war install advanced
```

プログラムが以下の情報のプロンプトを表示します。デフォルト値は角括弧で囲まれています。詳細については、Oracle ドキュメントの Introduction to Oracle REST Data Services を参照してください。

- 設定データを保存する場所を入力します。

  「*/home/apexuser/ORDS*」と入力します。これは ORDS 設定ファイルの場所です。
- 使用するデータベース接続タイプを指定します。[1] ベーシック、[2] TNS、[3] カスタム URL に対応する番号を入力します [1]:

  目的の接続タイプを選択します。
- データベースサーバーの名前を入力します [localhost]: *DB_instance_endpoint*

  デフォルト値を選択するか、適切な値を入力します。
- データベースリスナーポートを入力します [1521]: *DB_instance_port*

  デフォルト値を選択するか、適切な値を入力します。
- データベースサービス名を指定するには 1 を、データベース SID を指定するには 2 を入力してください [1]:

  2 を選択して、データベース SID を指定します。
- データベース SID [xe]

  デフォルト値を選択するか、適切な値を入力します。
- Oracle REST Data Services スキーマを検証/インストールする場合は 1 を、このステップをスキップする場合は 2 を入力します [1]:

[] を選択します1 このステップでは、ORDS_PUBLIC_USER という名前の Oracle REST Data Services プロキシユーザーを作成します。

- ORDS_PUBLIC_USER のデータベースパスワードを入力します。

  パスワードを入力し、確認のためにもう一度入力します。

- Oracle REST Data Services のスキーマを検証するには、管理者権限でログインする必要があります。

  管理者ユーザー名: *master_user*

  master_user: *master_user_password* のデータパスワードを入力する

  パスワードを確認する: *master_user_password*

  > **Note**
  >
  > セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

- ORDS_METADATA [SYSAUX] のデフォルトのテーブルスペースを入力します。

  ORDS_METADATA [TEMP] のテンポラリテーブルスペースを入力します。

  ORDS_PUBLIC_USER [USERS] のデフォルトのテーブルスペースを入力します。

  ORDS_PUBLIC_USER [TEMP] のテンポラリテーブルスペースを入力します。

- PL/SQL Gateway を使用する場合は 1 を入力し、このステップをスキップするには 2 を入力します。Oracle Application Express を使用している場合、または mod_plsql から移行する場合は、1 [1] を入力する必要があります。

  デフォルト値を選択します。

- PL/SQL ゲートウェイデータベースのユーザー名として [APEX_PUBLIC_USER] を入力します。

  デフォルト値を選択します。

- APEX_PUBLIC_USER のデータベースパスワードを入力します。

  パスワードを入力し、確認のためにもう一度入力します。

- 1 を入力して、Application Express RESTful Services データベースユーザー (APEX_LISTENER、APEX_REST_PUBLIC_USER) のパスワードを指定するか、2 を入力して、このステップをスキップします [1]:

  APEX 4.1.1.V1 に対して 2 を選択するか、他のすべての APEX バージョンに対して 1 を選択します。

- [APEX 4.1.1.v1 では不要] APEX_LISTENER のデータベースパスワード

  パスワードを入力し (必要な場合)、確認のためにもう一度入力します。

- [APEX 4.1.1.v1 では不要] APEX_REST_PUBLIC_USER のデータベースパスワード

  パスワードを入力し (必要な場合)、確認のためにもう一度入力します。

- 有効にする機能に対応する番号を入力します。

  SQL Developer Web、REST Enabled SQL、および Database API 機能をすべて有効にするには、1 を入力します。

- スタンドアロンモードでスタートする場合は 1 を、終了する場合は 2 を入力します [1]:

  1 と入力します。

- APEX 静的リソースの場所を入力します。

  APEX インストールファイルを /home/apexuser に解凍した場合は、「/home/apexuser/apex/images」と入力します。それ以外の場合は、*unzip_path*/apex/images を入力します。*unzip_path* は、ファイルを解凍したディレクトリです。

- HTTP を使用する場合は 1 を、HTTPS を使用する場合は 2 を入力します [1]:

  1 を入力する場合は、HTTP ポートを指定します。2 を入力する場合は、HTTPS ポートと SSL ホスト名を指定します。HTTPS オプションでは、証明書を提供する方法を指定するように求められます。

  - 自己署名証明書を使用するには、1 を入力します。
  - 独自の証明書を提供するには、2 を入力します。2 を入力する場合は、SSL 証明書のパス、およびその証明書のシークレットキーのパスを指定します。

8. APEX admin ユーザーのパスワードを設定します。これを行うには、SQL*Plus を使用して DB インスタンスにマスターユーザーとして接続し、次のコマンドを実行します。

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
```

```
@/home/apexuser/apex/apxchpwd.sql
```

master を自身のマスターユーザー名に置き換えます。apxchpwd.sql スクリプトによってプロンプトが表示されたら、新しい admin パスワードを入力します。

9. ORDS リスナーを起動します。以下の コードを実行します。

```
java -jar ords.war
```

ORDS を初めてスタートすると、APEX の静的リソースの場所を指定するように求められます。このイメージフォルダは、APEX のインストールディレクトリ内の /apex/images ディレクトリにあります。

10. ブラウザで Oracle APEX の管理ウィンドウに戻り、[管理] を選択します。次に、[Application Express Internal Administration] を選択します。認証情報を求められたら、以下の情報を入力します。

- User name - admin
- Password - apxchpwd.sql スクリプトを使用して設定したパスワード。

[Login] を選択し、その admin ユーザーの新しいパスワードを設定します。

これで、リスナーを使用する準備ができました。

ORDS 22 以降のインストールと設定

これで、Oracle APEX で使用する Oracle Rest Data Services (ORDS) をインストールして設定する準備が整いました。このセクションの例では、別個のホストの名前が myapexhost.example.com であり、ホストが Linux を実行していると仮定します。ORDS 22 の手順は、以前のリリースの手順とは異なります。

Oracle APEX で使用するための ORDS 22 以降をインストールして設定するには

1. Oracle REST data services に移動し、ダウンロードする ORDS バージョンの Readme を確認します。必要なバージョンの Java がインストールされていることを確認します。
2. ORDS インストール用の新しいディレクトリを作成します。

```
mkdir /home/apexuser/ORDS
cd /home/apexuser/ORDS
```

3. Oracle REST Data Services から、ファイル ords.*version.number*.zip または ords-latest.zip をダウンロードします。
4. ファイルを /home/apexuser/ORDS ディレクトリに解凍します。
5. マスターユーザーに ORDS のインストールに必要な権限を付与します。

   APEX オプションをインストールしたら、ORDS スキーマをインストールするために必要な権限をマスターユーザーに付与します。これを行うには、データベースにログインして、以下のコマンドを実行します。*MASTER_USER* をマスターユーザーの (大文字で記述した) 名前に置き換えます。

   > **Important**
   >
   > 大文字と小文字を区別する識別子を使用してユーザーを作成した場合を除き、ユーザー名を入力する際には大文字を使用します。例えば、CREATE USER myuser または CREATE USER MYUSER を実行すると、データディクショナリに MYUSER が保存されます。ただし、CREATE USER "MyUser" で二重引用符を使用すると、データディクショナリには MyUser が保存されます。詳細については、「SYS オブジェクトへの SELECT または EXECUTE 権限の付与」を参照してください。

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_ROLE_PRIVS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONS_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_CONSTRAINTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_OBJECTS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_PROCEDURES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TAB_COLUMNS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_TABLES', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('USER_VIEWS', 'MASTER_USER', 'SELECT',
 true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('WPIUTL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SESSION', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY', 'MASTER_USER',
 'EXECUTE', true);

exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_LOB', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_ASSERT', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_OUTPUT', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SCHEDULER', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('HTP', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('OWA', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('WPG_DOCLOAD', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_CRYPTO', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_METADATA', 'MASTER_USER',
 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SQL', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('UTL_SMTP', 'MASTER_USER', 'EXECUTE',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_NETWORK_ACL_ADMIN',
 'MASTER_USER', 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('SESSION_PRIVS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_USERS', 'MASTER_USER', 'SELECT',
 true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_NETWORK_ACL_PRIVILEGES',
 'MASTER_USER', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_NETWORK_ACLS', 'MASTER_USER',
 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_REGISTRY', 'MASTER_USER',
 'SELECT', true);
```

> **Note**
>
> 上記のコマンドは、ORDS 22 以降に適用されます。

6. ダウンロードした ords スクリプトを使用して、ORDS スキーマをインストールします。設定ファイルとログファイルを格納するディレクトリを指定します。Oracle Corporation では、これらのディレクトリを ORDS 製品ソフトウェアが含まれているディレクトリ内に配置しないことを推奨しています。

```
mkdir -p /home/apexuser/ords_config /home/apexuser/ords_logs

/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config \
  install --interactive --log-folder /home/apexuser/ords_logs
```

コンテナデータベース (CDB) アーキテクチャを実行する DB インスタンスでは、ORDS 23.3 以降を使用し、ORDS のインストール時に `--pdb-skip-disable-lockdown` 引数を渡します。

```
/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config \
  install --interactive --log-folder /home/apexuser/ords_logs --pdb-skip-disable-
lockdown
```

プログラムが以下の情報のプロンプトを表示します。デフォルト値は角括弧で囲まれています。詳細については、Oracle ドキュメントの Introduction to Oracle REST Data Services を参照してください。

- `Choose the type of installation:`

  2 を選択して ORDS スキーマをデータベースにインストールし、ローカルの ORDS 設定ファイルにデータベース接続プールを作成します。

- `Specify the database connection type to use. Enter number for [1] Basic [2] TNS [3] Custom URL:`

  目的の接続タイプを選択します。この例では、ユーザーが **1** を選択することを前提としています。

- `Enter the name of the database server [localhost]:` ***DB_instance_endpoint***

デフォルト値を選択するか、適切な値を入力します。

- `Enter the database listener port [1521]: DB_instance_port`

  デフォルト **1521** を選択するか、適切な値を入力します。

- `Enter the database service name [orcl]:`

  RDS for Oracle DB インスタンスで使用するデータベース名を入力します。

- `Provide database user name with administrator privileges`

  RDS for Oracle DB インスタンスのマスターユーザー名を入力します。

- `Enter the database password for [username]:`

  RDS for Oracle DB インスタンスのマスターパスワードを入力します。

- `Enter the default tablespace for ORDS_METADATA and ORDS_PUBLIC_USER [SYSAUX]:`
- `Enter the temporary tablespace for ORDS_METADATA [TEMP]. Enter the default tablespace for ORDS_PUBLIC_USER [USERS]. Enter the temporary tablespace for ORDS_PUBLIC_USER [TEMP].`
- `Enter a number to select additional feature(s) to enable [1]:`
- `Enter a number to configure and start ORDS in standalone mode [1]:`

  2 を選択して ORDS をスタンドアロンモードですぐに起動しないようにします。

- `Enter a number to select the protocol [1] HTTP`
- `Enter the HTTP port [8080]:`
- `Enter the APEX static resources location:`

  Oracle APEX インストールファイル (/home/apexuser/apex/images) へのパスを入力します。

7. Oracle APEX の admin ユーザーのパスワードを設定します。これを行うには、SQL*Plus を使用して DB インスタンスにマスターユーザーとして接続し、次のコマンドを実行します。

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

master を自身のマスターユーザー名に置き換えます。apxchpwd.sql スクリプトによってプロンプトが表示されたら、新しい admin パスワードを入力します。

8. ords スクリプトで serve コマンドを使用して ORDS をスタンドアロンモードで実行します。実稼働環境へのデプロイでは、Apache Tomcat や Oracle WebLogic Server など、サポートされている Java EE アプリケーションサーバーの使用を検討します。詳細については、Oracle Database ドキュメントの「Oracle REST Data Services のデプロイおよびモニター」を参照してください。

```
/home/apexuser/ORDS/bin/ords \
  --config /home/apexuser/ords_config serve \
  --port 8193 \
  --apex-images /home/apexuser/apex/images
```

ORDS が実行中でも Oracle APEX インストールにアクセスできない場合、次のエラーが (特に非 CDB インスタンスで) 表示されることがあります。

```
The procedure named apex_admin could not be accessed, it may not be declared,
 or the user executing this request may not have been granted execute privilege
 on the procedure, or a function specified by security.requestValidationFunction
 configuration property has prevented access.
```

このエラーを修正するには、ords スクリプトで config コマンドを実行して ORDS が使用するリクエスト検証機能を変更します。デフォルトでは、ORDS は CDB インスタンスでのみサポートされている ords_util.authorize_plsql_gateway プロシージャを使用します。非 CDB インスタンスでは、このプロシージャを wwv_flow_epg_include_modules.authorize パッケージに変更できます。ユースケースに応じてリクエスト検証関数を設定するためのベストプラクティスについては、Oracle Database のドキュメントと Oracle Support を参照してください。

9. ブラウザで Oracle APEX の管理ウィンドウに戻り、[管理] を選択します。次に、[Application Express Internal Administration] を選択します。認証情報を求められたら、以下の情報を入力します。

   - User name - admin
   - Password - apxchpwd.sql スクリプトを使用して設定したパスワード。

   [Login] を選択し、その admin ユーザーの新しいパスワードを設定します。

これで、リスナーを使用する準備ができました。

Oracle APEX のアップグレードと削除

Oracle APEX をアップグレードまたは削除するには、このトピックの手順に従います。

トピック

- Oracle APEX バージョンのアップグレード
- APEX オプションと APEX-DEV オプションの削除

Oracle APEX バージョンのアップグレード

> **Important**
>
> Oracle APEX をアップグレードする前に、DB インスタンスをバックアップします。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」および「Oracle DB アップグレードのテスト」を参照してください。

Oracle APEX を DB インスタンスと共にアップグレードするには、以下を実行します。

- DB インスタンスのアップグレードしたバージョン用に新規のオプショングループを作成します。
- 新規のオプショングループに、アップグレードしたバージョンの APEX オプションと APEX-DEV オプションを追加します。DB インスタンスが使用するそのほかのすべてのオプションを含めるようにします。詳細については、「オプショングループに関する考慮事項」を参照してください。
- DB インスタンスをアップグレードするときに、アップグレードした DB インスタンスに新規のオプショングループを指定します。

Oracle APEX のバージョンのアップグレード後に、以前のバージョンの Oracle APEX スキーマがデータベースに残存する場合があります。不要な場合は、アップグレード後に古い Oracle APEX スキーマをデータベースから削除できます。

Oracle APEX バージョンをアップグレードする場合、以前の Oracle APEX バージョンで RESTful サービスが設定されていないときは、RESTful サービスを設定することをお勧めします。詳細については、「Oracle APEX の RESTful サービスの設定」を参照してください。

DB インスタンスのメジャーバージョンアップグレードを行う際、ターゲットデータベースのバージョンと互換性のない Oracle APEX バージョンを使用していることに気付く場合があります。この

ような場合は、DB インスタンスをアップグレードする前に、Oracle APEX のバージョンをアップグレードできます。最初に Oracle APEX をアップグレードすると、DB インスタンスのアップグレードに要する時間を短縮できます。

> **Note**
>
> Oracle APEX をアップグレードしたら、アップグレードしたバージョンで使用するリスナーをインストールして設定します。手順については、「Oracle APEX リスナーの設定」を参照してください。

APEX オプションと APEX-DEV オプションの削除

> **Important**
>
> APEX オプションを削除すると、APEX リポジトリが削除され、DB インスタンスに保存されているすべての APEX アプリケーション、ワークスペース、ユーザーアカウントが完全に削除されます。このオプションを削除する前にアプリケーションをエクスポートします。

DB インスタンスから APEX オプションと APEX-DEV オプションを削除できます。これらのオプションを DB インスタンスから削除するには、次のいずれかを実行します。

- 複数の DB インスタンスから APEX オプションと APEX-DEV オプションを削除するには、それらが属しているオプショングループから両オプションを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。複数の DB インスタンスにアタッチされているオプショングループからオプションを削除すると、DB インスタンスが再起動される間、短時間の停止が発生します。

  詳細については、「オプショングループからオプションを削除する」を参照してください。
- 1 つの DB インスタンスから APEX オプションと APEX-DEV オプションを削除するには、DB インスタンスを変更し、これらのオプションを含まない別のオプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。オプションを削除する場合、DB インスタンスが自動的に再起動される間、短時間の停止が発生します。

  詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

DB インスタンスから APEX オプションと APEX-DEV オプションを削除すると、APEX スキーマがデータベースから削除されます。

Amazon EFS の統合

Amazon Elastic File System (Amazon EFS) は、サーバーレスで伸縮自在なファイルストレージを提供するため、ストレージ容量およびパフォーマンスのプロビジョニングや管理を行わずにファイルデータを共有できます。Amazon EFS では、ファイルシステムを作成し、NFS バージョン 4.0 および 4.1 (NFSv4) プロトコルを使用して VPC にマウントできます。そうすることで、他の POSIX 準拠のファイルシステムと同様に EFS ファイルシステムを使用できます。一般的な情報については、「Amazon Elastic ファイルシステムとは」および AWS ブログの「Amazon RDS for Oracle と Amazon EFS の統合」を参照してください。

トピック

- Amazon EFS 統合の概要
- Amazon EFS と RDS for Oracle を統合するネットワークアクセス許可の設定
- Amazon EFS と RDS for Oracle を統合する IAM アクセス許可の設定
- EFS_INTEGRATION (EFS 統合) オプションの追加
- Amazon EFS ファイルシステムのアクセス許可の設定
- RDS for Oracle と Amazon EFS ファイルシステム間のファイルの転送
- EFS_INTEGRATION (EFS 統合) オプションの削除
- Amazon EFS 統合のトラブルシューティング

Amazon EFS 統合の概要

Amazon EFS では、RDS for Oracle DB インスタンスと Amazon EFS ファイルシステムの間でファイルを転送できます。例えば、EFS を使用して次のユースケースをサポートできます。

- アプリケーションと複数のデータベースサーバー間でファイルシステムを共有します。
- トランスポータブル表領域データファイルなど、移行関連ファイル用の共有ディレクトリを作成します。詳細については、「Oracle トランスポータブル表領域を使用した移行」を参照してください。
- サーバーに追加のストレージスペースを割り当てることなく、アーカイブされた REDO ログファイルを保存および共有できます。
- UTL_FILE などの Oracle Database ユーティリティを使用してファイルの読み取りおよび書き込みを行います。

Amazon EFS 統合のメリット

他のデータ転送ソリューションではなく EFS ファイルシステムを選択すると、次のようなメリットがあります。

- Amazon EFS および RDS for Oracle DB インスタンスの間で Oracle Data Pump ファイルを転送できます。Data Pump は EFS ファイルシステムから直接インポートするため、これらのファイルをローカルにコピーする必要はありません。詳細については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。
- データ移行は、データベースリンクを使用するよりも高速です。
- RDS for Oracle DB インスタンスで、ファイルを保存するためのストレージ容量を割り当てることが回避できます。
- EFS ファイルシステムでは、ストレージをプロビジョニングしなくてもストレージを自動的にスケーリングできます。
- Amazon EFS 統合には最低料金やセットアップ費用はありません。お支払いいただくのは、使用分の料金だけです。
- Amazon EFS 統合は、転送時のデータの暗号化と保存時の暗号化の 2 つの形式の暗号化をサポートしています。転送時のデータの暗号化は、TLS バージョン 1.2 を使用してデフォルトで有効になっています。Amazon EFS ファイルシステムを作成する場合、保管時のデータの暗号化を有効にすることができます。詳細については、「Amazon Elastic File System ユーザーガイド」の「保管中のデータの暗号化」を参照してください。

Amazon EFS 統合の要件

以下の要件を満たしていることを確認してください。

- データベースは 19.0.0.0.ru-2022-07.rur-2022-07.r1 以降のデータベースを実行している必要があります。
- DB インスタンスおよび EFS ファイルシステムは、同じ AWS リージョン、VPC、AWS アカウントである必要があります。RDS for Oracle は、EFS のクロスアカウントおよびクロスリージョンアクセスをサポートしていません。
- VPC では [DNS 解決] と [DNS ホスト名] の両方を有効にする必要があります。詳細については、Amazon Virtual Private Cloud ユーザーガイドの「DNS attributes for your VPC」(VPC の DNS 属性) を参照してください。

- `mount` コマンドで DNS 名を使用する場合は、VPC が Amazon が提供する DNS サーバーを使用するように設定されていることを確認してください。カスタム DNS サーバーはサポートされていません。
- RDS 以外のソリューションを使用して EFS ファイルシステムをバックアップする必要があります。RDS for Oracle は、EFS ファイルシステムの自動バックアップや手動 DB スナップショットをサポートしていません。詳細については、「Amazon EFS ファイルシステムのバックアップ」を参照してください。

Amazon EFS と RDS for Oracle を統合するネットワークアクセス許可の設定

RDS for Oracle を Amazon EFS と統合するには、DB インスタンスが EFS ファイルシステムにネットワークアクセスできることを確認してください。詳細については、Amazon Elastic File System ユーザーガイド の「NFS クライアントの Amazon EFS ファイルシステムへのネットワークアクセス制御」を参照してください。

トピック

- セキュリティグループによるネットワークアクセス制御
- ファイルシステムポリシーによるネットワークアクセスの制御

セキュリティグループによるネットワークアクセス制御

VPC セキュリティグループなどのネットワーク層セキュリティメカニズムを使用して、DB インスタンスから EFS ファイルシステムへのアクセスを制御できます。DB インスタンスの EFS ファイルシステムへのアクセスを許可するには、EFS ファイルシステムが次の要件を満たしていることを確認してください。

- EFS マウント ターゲットは、RDS for Oracle DB インスタンスによって使用されるすべてのアベイラビリティーゾーンにあります。

  EFS マウントターゲットは、Amazon EFS ファイルシステムをマウントできる NFSv4 エンドポイントの IP アドレスを提供します。DNS 名を使用してファイルシステムをマウントすると、EC2 インスタンスと同じアベイラビリティーゾーンの EFS マウントターゲットの IP アドレスに解決されます。

  別の AZ の DB インスタンスが同じ EFS ファイルシステムを使用するように設定できます。マルチ AZ の場合、配置内の AZ ごとにマウントポイントが必要です。別の AZ への DB への DB インスタンスの移動が必要になる場合があります。そのため、VPC 内の各 AZ で EFS マウントポイン

トを作成することをお勧めします。デフォルトでは、コンソールを使用して新しい EFS ファイルシステムを作成すると、RDS はすべての AZ のマウントターゲットを作成します。

- セキュリティグループは、マウントターゲットにアタッチされます。
- セキュリティグループには、TCP/2049 (タイプ NFS) で RDS for Oracle DB インスタンスのネットワークサブネットまたはセキュリティグループを許可するインバウンドルールがあります。

詳細については、「Amazon Elastic File System ユーザーガイド」の「Creating Amazon EFS file systems」(Amazon EFS ファイルシステムの作成) と「マウントターゲットとセキュリティグループの作成と管理」を参照してください。

ファイルシステムポリシーによるネットワークアクセスの制御

Amazon EFS と RDS for Oracle の統合は、デフォルト (空) のEFS ファイルシステムポリシーで動作します。デフォルトのポリシーでは、認証に IAM を使用しません。代わりに、マウントターゲットを使用して、ファイルシステムに接続できる匿名クライアントへのフルアクセスを許可します。デフォルトポリシーは、ファイルシステムの作成時を含め、ユーザーが設定したファイルシステムポリシーが有効ではない場合は常に有効になります。詳細については、Amazon Elastic File System ユーザーガイドの「EFS ファイルシステムポリシー」を参照してください。

RDS for Oracle を含むすべてのクライアントの EFS ファイルシステムへのアクセスを強化するために、IAM アクセス許可を設定できます。この方法で、ファイルシステムポリシーを作成します。詳細については、Amazon Elastic File System ユーザーガイドの「ファイルシステムポリシーの作成」を参照してください。

Amazon EFS と RDS for Oracle を統合する IAM アクセス許可の設定

デフォルトでは、Amazon EFS 統合機能は IAM ロールを使用しません。`USE_IAM_ROLE` オプション設定は `FALSE` です。RDS for Oracle と Amazon EFS および IAM ロールを統合するには、DB インスタンスが Amazon EFS ファイルシステムにアクセスするために IAM アクセス許可を持っている必要があります。

トピック

- ステップ 1: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする
- ステップ 2: Amazon EFS ファイルシステムのファイルシステムポリシーを作成します。
- ステップ 3: IAM ロールを RDS for Oracle DB インスタンスに関連付ける

ステップ 1: DB インスタンスの IAM ロールを作成し、ポリシーをアタッチする

このステップでは、RDS for Oracle DB インスタンスのロールを作成し、Amazon RDS が EFS ファイルシステムにアクセスできるようにします。

コンソール

Amazon RDS が EFS ファイルシステムへのアクセスを許可する IAM ロールを作成するには

1. IAM マネジメントコンソールを開きます。
2. ナビゲーションペインで Roles (ロール) を選択してください。
3. [Create role] を選択します。
4. [AWS のサービス] で、[RDS] を選択します。
5. [ユースケースの選択] で、[RDS - Add Role to Database (ロールをデータベースに追加する)] を選択します。
6. [次へ] を選択します。
7. アクセス許可ポリシーを追加しないでください。[次へ] を選択します。
8. [ロール名] を IAM ロールの名前 (例: `rds-efs-integration-role`) に設定します。オプションで [Description] 値を追加することもできます。
9. [ロールの作成] を選択してください。

AWS CLI

サービスのアクセス許可を特定のリソースに限定するには、リソースベースの信頼関係で `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

両方のグローバル条件コンテキストキーを使用し、`aws:SourceArn` 値にアカウント ID を含めます。この場合は、`aws:SourceAccount` 値と `aws:SourceArn` 値のアカウントは、同じステートメントで使用する場合、同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は `aws:SourceArn` を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、`aws:SourceAccount`を使用します。

信頼関係では、`aws:SourceArn` グローバル条件コンテキストキーに、必ず、ロールにアクセスするリソースの完全な Amazon リソースネーム (ARN) を使用します。

次の AWS CLI コマンドでは、この目的で *rds-efs-integration-role* という名前のロールを作成します。

Example

Linux、macOS、Unix の場合:

```
aws iam create-role \
   --role-name rds-efs-integration-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                 "aws:SourceAccount": "my_account_ID",
                 "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
             }
         }
       }
     ]
   }'
```

Windows の場合:

```
aws iam create-role ^
   --role-name rds-efs-integration-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
```

```
                "aws:SourceAccount": "my_account_ID",
                "aws:SourceArn": "arn:aws:rds:Region:my_account_ID:db:dbname"
            }
        }
      }
    ]
  }'
```

詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

ステップ 2: Amazon EFS ファイルシステムのファイルシステムポリシーを作成します。

このステップでは、EFS ファイルシステムのファイルシステムポリシーを作成します。

EFS ファイルシステムポリシーを作成または編集するには

1. EFS 管理コンソールを開きます。
2. [File Systems (ファイルシステム)] を選択します。
3. [File systems (ファイルシステム)] ページで、ファイルシステムポリシーを編集または作成する対象のファイルシステムを選択します。そのファイルシステムの詳細ページが表示されます。
4. [File system policy] (ファイルシステムポリシー) タブを選択します。

   ポリシーが空の場合は、デフォルトの EFS ファイルシステムポリシーが使用されます。詳細については、Amazon Elastic File System ユーザーガイドの「EFS ファイルシステムポリシー」を参照してください。
5. [編集] を選択します。[File system policy (ファイルシステムポリシー)] ページが表示されます。
6. [Policy editor] (ポリシーエディタ) で次のようなポリシーを入力し、[Save] (保存) を選択します。

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/rds-efs-integration-
role"
            },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-
east-1:123456789012:file-system/fs-1234567890abcdef0"
        }
    ]
}
```

ステップ 3: IAM ロールを RDS for Oracle DB インスタンスに関連付ける

このステップでは、IAM ロールを DB インスタンスに関連付けます。以下の要件に注意してください。

- 必須の Amazon EFS アクセス許可ポリシーがアタッチされた IAM ロールへのアクセスが許可されている必要があります。
- RDS for Oracle DB インスタンスには、一度に 1 つの IAM ロールのみを関連付けることができます。
- インスタンスのステータスは [Available] (使用可能) である必要があります。

詳細については、Amazon Elastic File System ユーザーガイドの「Amazon EFS の ID とアクセス管理」を参照してください。

コンソール

IAM ロールを RDS for Oracle DB インスタンスに関連付けるには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [データベース] をクリックします。
3. データベースインスタンスが使用できない場合は、[Actions (アクション)]、[Start (スタート)] の順に選択します。インスタンスのステータスに [Started (スタート済み)] と表示されたら、次のステップに進みます。

4. 詳細を表示する Oracle DB インスタンスの名前を選択します。
5. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、次のいずれかを実行します。
6. IAM ロールをこのインスタンスに追加するセクションに追加するロールを選択します。
7. [Feature] (機能) で、[EFS_INTEGRATION] (EFS 統合) を選択します。
8. [Add role] を選択します。

AWS CLI

以下の AWS CLI コマンドでは、*mydbinstance* という名前の Oracle DB インスタンスにこのロールを追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-role-to-db-instance \
   --db-instance-identifier mydbinstance \
   --feature-name EFS_INTEGRATION \
   --role-arn your-role-arn
```

Windows の場合:

```
aws rds add-role-to-db-instance ^
   --db-instance-identifier mydbinstance ^
   --feature-name EFS_INTEGRATION ^
   --role-arn your-role-arn
```

your-role-arn を、以前のステップで書き留めたロール ARN に置き換えます。`EFS_INTEGRATION` オプションには `--feature-name` が指定されている必要があります。

EFS_INTEGRATION (EFS 統合) オプションの追加

Amazon RDS for Oracle と Amazon EFS を統合するには、DB インスタンスが `EFS_INTEGRATION` オプションを含むオプショングループに関連付けられている必要があります。

同じオプショングループに属する複数の Oracle DB インスタンスは、同じ EFS ファイルシステムを共有します。異なる DB インスタンスは同じデータにアクセスできますが、異なる Oracle ディレクトリを使用することでアクセスを分割できます。詳細については、「RDS for Oracle と Amazon EFS ファイルシステム間のファイルの転送」を参照してください。

コンソール

Amazon EFS 統合用のオプショングループを設定するには

1. 新しいオプショングループを作成するか、EFS_INTEGRATION オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [EFS_INTEGRATION] オプションを追加します。EFS_ID ファイルシステム ID を指定し、USE_IAM_ROLE フラグを設定する必要があります。

   詳細については、「オプショングループにオプションを追加する」を参照してください。

3. 次のいずれかの方法を使用して、オプショングループを DB インスタンスに関連付けます。

   - 新しい Oracle DB インスタンスを作成して、オプショングループを関連付けます。DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - Oracle DB インスタンスを変更し、オプショングループを関連付けます。Oracle DB インスタンスの変更方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

EFS 統合用のオプショングループを設定するには

1. 新しいオプショングループを作成するか、EFS_INTEGRATION オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [EFS_INTEGRATION] オプションを追加します。

   例えば、以下の AWS CLI コマンドでは、EFS_INTEGRATION オプションを、**myoptiongroup** という名前のオプショングループに追加します。

   Example

   Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
   --option-group-name myoptiongroup \
   --options "OptionName=EFS_INTEGRATION,OptionSettings=\
   [{Name=EFS_ID,Value=fs-1234567890abcdef0},{Name=USE_IAM_ROLE,Value=TRUE}]"
```

Windows の場合:

```
aws rds add-option-to-option-group ^
   --option-group-name myoptiongroup ^
   --options "OptionName=EFS_INTEGRATION,OptionSettings=^
   [{Name=EFS_ID,Value=fs-1234567890abcdef0},{Name=USE_IAM_ROLE,Value=TRUE}]"
```

3. 次のいずれかの方法を使用して、オプショングループを DB インスタンスに関連付けます。

   - 新しい Oracle DB インスタンスを作成して、オプショングループを関連付けます。DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - Oracle DB インスタンスを変更し、オプショングループを関連付けます。Oracle DB インスタンスの変更方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon EFS ファイルシステムのアクセス許可の設定

デフォルトでは、ルートユーザー (UID 0) のみが読み取り、書き込み、実行のアクセス許可を持ちます。他のユーザーがファイルシステムを変更できるようにするには、ルートユーザーは、明示的にアクセス許可を付与する必要があります。RDS for Oracle DB インスタンスのユーザーは `others` カテゴリに含まれます。詳細については、Amazon Elastic File System ユーザーガイドの「ネットワークファイルシステム (NFS) レベルでのユーザー、グループ、アクセス許可の操作」を参照してください。

RDS for Oracle DB インスタンスによって EFS ファイルシステムのファイルを読み書きできるようにするには、次の手順を実行します。

- Amazon EC2 またはオンプレミスインスタンスに、EFS ファイルシステムをローカルでマウントします。
- きめ細かいアクセス許可を設定します。

例えば、other ユーザーに EFS ファイルシステムのルートへの書き込み許可を付与するには、このディレクトリで chmod 777 を実行します。詳細については、Amazon Elastic File System ユーザーガイドの「Amazon EFS ファイルシステムのユースケースとアクセス許可の例」を参照してください。

RDS for Oracle と Amazon EFS ファイルシステム間のファイルの転送

RDS for Oracle インスタンスと Amazon EFS ファイルシステム間でファイルを転送するには、Oracle ディレクトリを 1 つまたは複数作成し、DB インスタンスのアクセスを制御する EFS ファイルシステム権限を設定します。

トピック

- Oracle ディレクトリの作成
- EFS ファイルシステム間のデータ転送: 例

Oracle ディレクトリの作成

ディレクトリを作成するには、rdsadmin.rdsadmin_util.create_directory_efs のプロシージャを使用します。 プロシージャには以下のパラメータがあります。

パラメータ名	データ型	デフォルト	必須	説明
p_directory_name	VARCHAR	–	はい	Oracle ディレクトリの名前。
p_path_on_efs	VARCHAR	–	はい	EFS ファイルシステムのパス。パス名のプレフィックスには /rdsefs-*fsid*/ パターンが使用されています。*fsid* は EFS ファイルシステム ID のプレースホルダーです。 例えば、fs-1234567890abcdef0 という名前の EFS ファイルシステムがあり、この mydir という名前のファイルシステムにサブディレクトリを作成する場合、次の値を指定できます。

パラメータ名	データ型	デフォルト	必須	説明
				/rdsefs-fs-1234567890abcdef0/mydir

fs-1234567890abcdef0 という EFS ファイルシステムに、/datapump1 という名前のサブディレクトリを作成したとします。次の例では、EFS ファイルシステムの /datapump1 ディレクトリを指す Oracle ディレクトリ DATA_PUMP_DIR_EFS を作成しています。p_path_on_efs パラメータのファイルシステムパス値の先頭には文字列 /rdsefs- が付きます。

```
BEGIN
  rdsadmin.rdsadmin_util.create_directory_efs(
    p_directory_name => 'DATA_PUMP_DIR_EFS',
    p_path_on_efs    => '/rdsefs-fs-1234567890abcdef0/datapump1');
END;
/
```

EFS ファイルシステム間のデータ転送: 例

次の例では、Oracle Data Pump を使用して、MY_TABLE という名前のテーブルを datapump.dmp ファイルにエクスポートしています。このファイルは EFS ファイルシステムにあります。

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'EXPORT', job_mode => 'TABLE',
 job_name=>null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump.dmp',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_dump_file);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump-exp.log',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'NAME_EXPR','IN (''MY_TABLE'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
```

```
END;
/
```

次の例では、Oracle Data Pump を使用して、MY_TABLE という名前のテーブルを datapump.dmp ファイルからインポートしています。このファイルは EFS ファイルシステムにあります。

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'TABLE',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump.dmp',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'datapump-imp.log',
    directory => 'DATA_PUMP_DIR_EFS',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'NAME_EXPR','IN (''MY_TABLE'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

詳細については、「Amazon RDS の Oracle にデータをインポートする」を参照してください。

EFS_INTEGRATION (EFS 統合) オプションの削除

EFS_INTEGRATION オプションを削除する手順は、複数の DB インスタンスからオプションを削除するか、単一のインスタンスから削除するかによって異なります。

DB インスタンス数	アクション	関連情報
複数	DB インスタンスが所属するオプショングループから EFS_INTEGRATION オプションを削除します。この変更は、そのオプショングル	オプショングループからオプションを削除する

DB インスタンス数	アクション	関連情報
	ープを使用するすべてのインスタンスに影響します。	
単一	DB インスタンスを修正して、EFS_INTEGRATION オプションが含まれない別オプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。	Amazon RDS DB インスタンスを変更する

EFS_INTEGRATION オプションを削除した後、オプションで、DB インスタンスに接続された EFS ファイルシステムを削除できます。

Amazon EFS 統合のトラブルシューティング

RDS for Oracle DB インスタンスは、Amazon EFS ファイルシステムとの接続をモニタリングしています。モニタリングによって問題が検出されると、その問題を修正して RDS コンソールにイベントを公開しようとする場合があります。詳細については、「Amazon RDS イベントの表示」を参照してください。

このセクションの情報を使用して、Amazon EFS 統合を使用する際の一般的な問題の診断と修正を行います。

Notification	説明	Action
`The EFS for RDS Oracle instance instance_name isn't available on the primary host. NFS port 2049 of your EFS isn't reachable.`	DB インスタンスが、EFS ファイルシステムと通信できない。	以下を確認してください。 • EFS ファイルシステムが存在している。 • EFS マウントターゲットにアタッチされているセキュリティグループには、TCP/2049 (タイプ NFS) で RDS for Oracle DB インスタンスのセキュリティグループ、またはネットワークサブ

Notification	説明	Action
		ネットを許可するインバウンドルールがある。
The EFS isn't reachable.	EFS_INTEGRATION オプションのインストール中にエラーが発生した。	以下を確認してください。 • EFS ファイルシステムが存在している。 • EFS マウントターゲットにアタッチされているセキュリティグループには、TCP/2049 (タイプ NFS) で RDS for Oracle DB インスタンスのセキュリティグループ、またはネットワークサブネットを許可するインバウンドルールがある。 • VPC の enableDnsSupport 属性が有効になっている。 • お客様の VPC では Amazon が提供する DNS サーバーを使用しています。Amazon EFS 統合はカスタム DHCP DNS では機能しません。
The associated role with your DB instance wasn't found.	EFS_INTEGRATION オプションのインストール中にエラーが発生した。	IAM ロールが RDS for Oracle DB インスタンスに関連付けられていることを確認してください。

Notification	説明	Action
`The associated role with your DB instance wasn't found.`	`EFS_INTEGRATION` オプションのインストール中にエラーが発生した。RDS for Oracle は、`USE_IAM_ROLE` オプション設定が `TRUE` の DB スナップショットから復元されました。	IAM ロールが RDS for Oracle DB インスタンスに関連付けられていることを確認してください。
`The associated role with your DB instance wasn't found.`	`EFS_INTEGRATION` オプションのインストール中にエラーが発生した。RDS for Oracle は、`USE_IAM_ROLE` オプション設定が `TRUE` のオールインワン CloudFormation テンプレートから作成されました。	回避策として、以下のステップを実行します。 1. IAM ロールとデフォルトのオプショングループを使用して DB インスタンスを作成します。 2. 後続のスタックの更新では、`EFS_INTEGRATION` オプションを使用してカスタムオプショングループを追加します。
`PLS-00302: component 'CREATE_DIRECTORY_EFS' must be declared`	このエラーは、Amazon EFS をサポートしていないバージョンの RDS for Oracle を使用している場合に発生する可能性があります。	RDS for Oracle DB インスタンスのバージョン 19.0.0.0.ru-2022-07.rur-2022-07.r1 以降を使用していることを確認してください。
`Read access of your EFS is denied. Check your file system policy.`	DB インスタンスは EFS ファイルシステムを読み取ることができません。	EFS ファイルシステムが IAM ロールまたは EFS ファイルシステムレベルでの読み取りアクセスを許可していることを確認してください。

Notification	説明	Action
該当なし	DB インスタンスは EFS ファイルシステムに書き込むことができません。	次のステップを実行します。 1. EFS ファイルシステムが Amazon EC2 インスタンスにマウントされていることを確認してください。 2. `others` グループに RDS ユーザーへの書き込み権限を付与します。EFS ファイルシステムのトップディレクトリで `chmod 777` コマンドを実行するのが一番簡単です。

Oracle Java Virtual Machine

Amazon RDS は、JVM オプションを使用することで Oracle Java Virtual Machine (JVM) をサポートします。Oracle Java では、SQL スキーマと関数が提供され、Oracle データベース内で Oracle Java の機能を活用できます。詳細については、Oracle ドキュメントの「Oracle Database における Java の概要」を参照してください。Oracle JVM は、Oracle Database 26ai (26.0.0)、21c (21.0.0) および 19c (19.0.0) のすべてのバージョンで使用できます。

Oracle JVM に関する考慮事項

Amazon RDS の Java 実装のアクセス許可セットは制限されています。マスターユーザーには RDS_JAVA_ADMIN ロールが付与されます。このロールでは、JAVA_ADMIN ロールによって付与される特権のサブセットを付与します。RDS_JAVA_ADMIN ロールに付与される特権を一覧表示するには、DB インスタンスで次のクエリを実行します。

```
SELECT * FROM dba_java_policy
   WHERE grantee IN ('RDS_JAVA_ADMIN', 'PUBLIC')
   AND enabled = 'ENABLED'
   ORDER BY type_name, name, grantee;
```

Oracle JVM の前提条件

Oracle Java を使用するための前提条件は次のとおりです。

- DB インスタンスが十分な大きさのクラスである必要があります。Oracle Java は、db.t3.small DB インスタンスクラスではサポートされていません。詳細については、「 DB インスタンスクラス」を参照してください。
- DB インスタンスで [マイナーバージョン自動アップグレード] が有効になっている必要があります。このオプションにより、リリースされた DB エンジンのマイナーバージョンアップグレードが、自動的に DB インスタンスに適用されるようになります。Amazon RDS では、このオプションを使用して、DB インスタンスに対し最新の Oracle パッチセット更新 (PSU)、またはリリース更新 (RU) を行います。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle JVM のベストプラクティス

Oracle Java を使用するためのベストプラクティスは次のとおりです。

- セキュリティを最大にするためには、JVM オプションを Secure Sockets Layer (SSL) で使用します。詳細については、「Oracle Secure Sockets Layer」を参照してください。
- ネットワークアクセスを制限するように、DB インスタンスを設定します。詳細については、「VPC の DB インスタンスにアクセスするシナリオ」および「VPC 内の DB インスタンスの使用」を参照してください。
- 次の条件を満たす場合は、TLSv1.2 をサポートするように HTTPS エンドポイントの設定を更新します。
  - Oracle Java Virtual Machine (JVM) を使用して、TLSv1 または TLSv1.1 プロトコルを介して HTTPS エンドポイントを接続します。
  - エンドポイントは TLSv1.2 プロトコルをサポートしていません。
  - 2021 年 4 月リリースの更新を Oracle DB に適用していません。

  エンドポイント設定を更新することで、HTTPS エンドポイントへの JVM の接続が確実に引き続き機能するようにします。Oracle JRE および JDK の TLS の変更の詳細については、Oracle JRE and JDK Cryptographic Roadmap を参照してください。

Oracle JVM オプションの追加

DB インスタンスに JVM オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

JVM オプションが追加されるあいだ、短い停止が発生します。オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに Oracle Java が使用可能となります。

> **Note**
>
> この停止中、パスワード検証機能は一時的に無効になります。また、停止中にパスワード検証機能に関連するイベントを確認することもできます。Oracle DB インスタンスが使用可能になる前に、パスワード検証機能が再び有効になります。

JVM オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   - [エンジン] に、DB インスタンスによって使用される DB エンジン ([oracle-ee] または [oracle-se2]) を選択します。
   - [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [JVM] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
4. ユーザーに必要なアクセス権限を付与します。

   Amazon RDS マスターユーザーには、JVM オプションを使用するアクセス許可がデフォルトで付与されています。他のユーザーがこれらのアクセス許可を必要とする場合は、SQL クライアントのマスターユーザーとして DB インスタンスに接続し、そのユーザーにアクセス許可を付与します。

   次の例では、JVM オプションを使用するアクセス許可を test_proc ユーザーに付与しています。

```
create user test_proc identified by password;
CALL dbms_java.grant_permission('TEST_PROC',
 'oracle.aurora.security.JServerPermission', 'LoadClassInPackage.*', '');
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

ユーザーにアクセス許可が付与されると、次のクエリで出力が返されます。

```
select * from dba_java_policy where grantee='TEST_PROC';
```

> **Note**
>
> Oracle ユーザー名では大文字と小文字が区別され、通常すべて大文字が使用されます。

Oracle JVM オプションの削除

DB インスタンスから JVM オプションを削除できます。オプションが削除されるあいだ、短い停止が発生します。JVM オプションを削除した後に DB インスタンスを再起動する必要はありません。

> **Warning**
>
> JVM オプションを削除すると、DB インスタンスがオプションの一部として有効であったデータ型を使用している場合、データ損失が発生する可能性があります。続行する前にデータをバックアップしてください。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

JVM オプションを DB インスタンスから削除するには、次のいずれかを実行します。

- JVM オプションを所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、JVM オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Enterprise Manager

Amazon RDS は Oracle Enterprise Manager (OEM) をサポートしています。OEM は、エンタープライズ情報技術の統合管理用の Oracle 製品ラインです。

Amazon RDS は、Oracle Database 19c の非 CDB または CDB、21c CDB、26ai CDB で OEM をサポートしています。次の表で、サポートされている OEM オプションについて説明します。

オプション	オプション ID	サポートされている OEM のリリース
OEM Database Express	`OEM`	OEM Database Express 19c
OEM Management Agent	`OEM_AGENT`	OEM Cloud Control for 13c

> **Note**
>
> OEM Database または OEM Management Agent を使用できますが、両方を使用することはできません。OEM Database Express を使用すると、追加のインフラストラクチャなしで単一のデータベースを軽量かつ組み込みでモニタリングできます。OEM Management Agent を使用して、既存の Oracle Enterprise Manager Cloud Control インストールを通じて DB インスタンスを一元管理します。

Oracle Enterprise Manager Database Express

Amazon RDS は OEM オプションの使用を通じて、Oracle Enterprise Manager Database Express (EM Express) をサポートします。Amazon RDS は、CDB または非 CDB アーキテクチャを使用する EM Express for Oracle Database 19c をサポートします。

EM Express は、データベースに含まれるウェブベースのデータベース管理ツールで、開いている場合のみ使用できます。主要なパフォーマンス管理機能と基本的なデータベース管理機能をサポートしています。詳細については、Oracle Database ドキュメントの「Introduction to Oracle Enterprise Manager Database Express」を参照してください。

> **Note**
>
> EM Express は、db.t3.small DB インスタンスクラスではサポートされていません。DB インスタンスクラスの詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。

> **Note**
>
> Oracle Database 26ai では、Oracle Enterprise Manager Database Express (EM Express) はサポートされなくなりました。詳細については、Oracle Database ドキュメントの「Oracle Database 26ai データベース管理者ガイド」を参照してください。

OEM オプション設定

Amazon RDS は、OEM オプションの次の設定をサポートします。

オプション設定	有効な値	説明
ポート –	整数値	EM Express をリッスンする RDS for Oracle DB インスタンスのポート。デフォルトは 5500 です。
セキュリティグループ	—	[Port] へのアクセス権限を持つセキュリティグループ。

ステップ 1: OEM オプションの追加

DB インスタンスに OEM オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループにオプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

OEM オプションを追加する場合、DB インスタンスが自動的に再起動する間に短い停止が発生します。

OEM オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] で DB インスタンスの Oracle エディションを選択します。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに OEM オプションを追加し、オプションを設定します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。各設定の詳細については、「OEM オプション設定」を参照してください。

   > **Note**
   >
   > すでに 1 つ以上の DB インスタンスにアタッチされている既存のオプショングループに OEM オプションを追加すると、すべての DB インスタンスが自動的に再起動する間に短い停止が発生します。

3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

- 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。OEM オプションを追加する場合、DB インスタンスが自動的に再起動する間に短い停止が発生します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

> **Note**
>
> AWS CLI を使用して OEM オプションを追加することもできます。例については、「オプショングループにオプションを追加する」を参照してください。

ステップ 2: (CDB のみ) DBSNMP ユーザーアカウントのロック解除

DB インスタンスが CDB アーキテクチャを使用している場合は、EM Express に DBSNMP として口グインする必要があります。CDB では、DBSNMP は共通ユーザーです。デフォルトでは、このアカウントはロックされています。DB インスタンスが CDB アーキテクチャを使用しない場合は、このステップをスキップします。

CDB インスタンスで DBSNMP ユーザーアカウントをロック解除するには

1. SQL*Plus または別の Oracle SQL アプリケーションで、マスターユーザーとして DB インスタンスにログインします。
2. 次のストアドプロシージャを実行して、DBSNMP アカウントのロックを解除します。

```
EXEC rdsadmin.rdsadmin_util.reset_oem_agent_password('new_password');
```

プロシージャが存在しないことを示すエラーが表示された場合は、CDB インスタンスを再起動して自動的にインストールします。詳細については、「 DB インスタンスの再起動」を参照してください。

ステップ 3: ブラウザから EM Express にアクセスする

ウェブブラウザから EM Express にアクセスすると、ログインウィンドウが表示され、ユーザー名とパスワードを求められます。

ブラウザから EM Express にアクセスするには

1. Amazon RDS DB インスタンスのエンドポイントと EM Express ポートを特定します。Amazon RDS DB インスタンスのエンドポイントを見つける方法については、「RDS for Oracle DB インスタンスのエンドポイントを見つける」を参照してください。
2. ブラウザのロケーターバーに URL を次の形式で入力します。

   ```
   https://endpoint.rds.amazonaws.com:port/em
   ```

   例えば、Amazon RDS DB インスタンスのエンドポイントが `mydb.a1bcde234fgh.us-east-1.rds.amazonaws.com` で、EM Express ポートが `5500` の場合、EM Express にアクセスする URL は次のようになります。

   ```
   https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:5500/em
   ```

3. ログインの詳細の入力を求められたら、データベースアーキテクチャに応じて、次のいずれかのアクションを実行します。

   データベースが非 CDB の場合

   DB インスタンスのマスターユーザー名とマスターパスワードを入力します。

   データベースが CDB の場合

   ユーザーに `DBSNMP` と入力し、`DBSNMP` のパスワードを入力します。`Container` フィールドは空白のままにします。

OEM データベース設定の変更

OEM データベースを有効にすると、オプションのセキュリティグループ設定を変更できます。

オプショングループを DB インスタンスに関連付けた後に OEM ポート番号を変更することはできません。DB インスタンスの OEM ポート番号を変更するには、以下の作業を行います。

1. 新しいオプショングループを作成します。
2. 新しいポート番号の OEM オプションを新しいオプショングループに追加します。
3. DB インスタンスから既存のオプショングループを削除します。
4. 新しいオプショングループを DB インスタンスに追加します。

オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。各設定の詳細については、「OEM オプション設定」を参照してください。

OEM Database Express タスクの実行

Amazon RDS プロシージャを使用して、特定の OEM Database Express のタスクを実行できます。これらの手順を実行すると、以下のタスクを実行できます。

> **Note**
>
> OEM Database Express のタスクは非同期で実行されます。

タスク

- OEM Database Express のウェブサイトフロントエンドを Adobe Flash に切り替える
- OEM Database Express のウェブサイトフロントエンドを Oracle JET に切り替える

OEM Database Express のウェブサイトフロントエンドを Adobe Flash に切り替える

> **Note**
>
> このタスクは、Oracle Database 19c 非 CDB でのみ使用できます。

Oracle Database 19c 以降では、以前の OEM Database Express ユーザーインターフェイスは非推奨になりました。このユーザーインターフェイスは Adobe Flash に基づいていました。代わりに、OEM Database Express は Oracle JET で構築されたインターフェイスを使用するようになりました。新しいインターフェイスで問題が発生した場合は、非推奨の Flash ベースのインターフェイスに戻すことができます。新しいインターフェイスで発生する可能性のある問題として、OEM Database Express にログインした後に `Loading` 画面が停止することがあります。また、Flash ベースのバージョンの OEM Database Express に存在していた特定の機能を見逃す可能性もあります。

OEM Database Express のウェブサイトフロントエンドを Adobe Flash に切り替えるには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash` を実行します 。このプロシージャは、`execemx emx` SQL コマンドに相当します。

セキュリティのベストプラクティスでは、Adobe Flash の使用は推奨されていません。Flash ベースの OEM Database Express に戻すことはできますが、可能であれ

ば、JET ベースの OEM Database Express のウェブサイトを使用することをお勧めします。Adobe Flash を使用するように戻した後で、Oracle JET の使用に切り替える場合は、`rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet` プロシージャを使用します。Oracle データベースにアップグレードすると、新しいバージョンの Oracle JET では、OEM Database Express の JET 関連の問題が解決される場合があります。Oracle JET への切り替えの詳細については、「OEM Database Express のウェブサイトフロントエンドを Oracle JET に切り替える」を参照してください。

> **Note**
>
> リードレプリカの出典 DB インスタンスからこのタスクを実行すると、リードレプリカは OEM Database Express ウェブサイトフロントエンドを Adobe Flash に切り替えます。

次のプロシージャの呼び出しでは、OEM Database Express ウェブサイトを Adobe Flash に切り替えるタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示すると、結果を確認できます。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

task-id は、この手順で返されたタスク ID に置き換えます。Amazon RDS プロシージャ `rdsadmin.rds_file_util.read_text_file` の詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

AWS マネジメントコンソール の [ログとイベント] セクションでログエントリを検索して、タスクの出力ファイルの内容を `task-id` に表示することもできます。

OEM Database Express のウェブサイトフロントエンドを Oracle JET に切り替える

> **Note**
>
> このタスクは、Oracle Database 19c 非 CDB でのみ使用できます。

OEM Database Express のウェブサイトフロントエンドを Oracle JET に切り替えるには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet` を実行します。このプロシージャは、`execemx omx` SQL コマンドに相当します。

デフォルトでは、19c 以降を実行している Oracle DB インスタンス用の OEM Database Express ウェブサイトは Oracle JET を使用します。`rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash` プロシージャを使用して OEM Database Express のウェブサイトフロントエンドを Adobe Flash に切り替えた場合は、Oracle JET に戻すことができます。これを行うには、`rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet` プロシージャを使用します。Adobe Flash への切り替えの詳細については、「OEM Database Express のウェブサイトフロントエンドを Adobe Flash に切り替える」を参照してください。

> **Note**
>
> リードレプリカの出典 DB インスタンスからこのタスクを実行すると、リードレプリカによって OEM Database Express ウェブサイトフロントエンドが Oracle JET に切り替わります。

次のプロシージャの呼び出しでは、OEM Database Express ウェブサイトを Oracle JET に切り替えるタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示すると、結果を確認できます。

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

task-id は、この手順で返されたタスク ID に置き換えます。Amazon RDS プロシージャ `rdsadmin.rds_file_util.read_text_file` の詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

AWS マネジメントコンソール の [ログとイベント] セクションでログエントリを検索して、タスクの出力ファイルの内容を `task-id` に表示することもできます。

OEM データベースオプションの削除

DB インスタンスから OEM オプションを削除できます。OEM オプションを削除する場合、インスタンスが自動的に再起動する間に短い停止が発生します。そのため、OEM オプションを削除した後に DB インスタンスを再起動する必要はありません。

OEM オプションを DB インスタンスから削除するには、次のいずれかを実行します。

- OEM オプションを、所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、OEM オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Enterprise Manager Cloud Control 向け Oracle Management Agent

Oracle Enterprise Manager (OEM) Management Agent は、ホスト上で実行中のターゲットをモニタリングし、その情報を中間層 Oracle Management Service (OMS) に送信するソフトウェアコンポーネントです。Amazon RDS は `OEM_AGENT` オプションを使用して Management Agent をサポートします。

詳細については、Oracle ドキュメントの「Oracle Enterprise Manager Cloud Control の概要」を参照してください。

トピック

- Management Agent の要件
- OMS ホスト通信の前提条件
- Management Agent の制限
- Management Agent のオプション設定
- Management Agent オプションを DB インスタンス用に有効にする
- Management Agent オプションの削除
- Management Agent を使用したデータベースタスクの実行

Management Agent の要件

以下は、Management Agent を使用するための一般的な要件です。

- CDB アーキテクチャまたは非 CDB アーキテクチャのいずれかを使用できます。
- DB インスタンスに接続するように設定されている Oracle Management Service (OMS) が必要です。次の OMS 要件に注意してください。
  - Management Agent のバージョン 24.1.0.0.v1 には、OMS バージョン 24.1 が必要です。
  - Management Agent のバージョン 13.5.0.0.v2 および 13.5.0.0.v3 には、OMS バージョン 13.5.0.23 または 24.1 が必要です。
  - Management Agent のバージョン 13.5.0.0.v1 には、OMS バージョン 13.5.0.0 または 24.1 が必要です。
  - Management Agent のバージョン 13.4.0.9.v1 および 13.4.0.9.v2 には、OMS バージョン 13.4.0.9 以降と 32198287 のパッチが必要です。
- ほとんどの場合、OMS から DB インスタンスへの接続を許可するように VPC を設定する必要があります。Amazon Virtual Private Cloud (Amazon VPC) についてあまり詳しくない場合は、

「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」のステップを完了してから続行することをお勧めします。

- Management Agent は、Oracle Enterprise Manager Cloud Control for 12c および 13c で使用できます。OEM リリース用の十分なストレージ領域があることを確認します。
  - OEM 24.1 リリース 1 では 8.5 GiB 以上
  - OEM 13c リリース 5 では 8.5 GiB 以上
  - OEM 13c リリース 4 では 8.5 GiB 以上
  - OEM 13c リリース 3 では 8.5 GiB 以上
  - OEM 13c リリース 2 では 5.5 GiB 以上
  - OEM 13c リリース 1 では 4.5 GiB 以上
  - OEM 12c では 2.5 GiB 以上
- Management Agent バージョン `OEM_AGENT 13.2.0.0.v3` および `13.3.0.0.v2` を使用している場合、また、TCPS 接続を使用する場合は、Oracle ドキュメントの「ターゲットデータベースと通信するためのサードパーティー CA 証明書の設定」の手順に従ってください。また、Oracle Doc ID 2241358.1 の Oracle のドキュメントの指示に従って、OMS の JDK を更新します。このステップにより、データベースがサポートするすべての暗号スイートが OMS でサポートされるようになります。

> **Note**
>
> Management Agent と DB インスタンス間の TCPS 接続は、Management Agent の `OEM_AGENT 13.2.0.0.v3`、`13.3.0.0.v2`、`13.4.0.9.v1` 以上のバージョンでサポートされます。

OMS ホスト通信の前提条件

OMS ホストと Amazon RDS DB インスタンスが通信できることを確認してください。以下の操作を実行します。

- OMS ホストがファイアウォールの内側にある場合、Management Agent から OMS ホストに接続するには、ファイアウォール に DB インスタンスの IP アドレスを追加します。OMS のファイアウォールで次のネットワークトラフィックが許可されていることを確認します。

OMS ホストから DB インスタンスへ

OMS ホストからデータベースリスナーポート (デフォルトは 1521) および OEM エージェントポート (デフォルトは 3872) へのトラフィックを許可する一方向ファイアウォールルールを設定します。

DB インスタンスから OMS ホストへ

DB インスタンスから OMS HTTP ポート (デフォルトは 4903) へのトラフィックを許可する一方向ファイアウォールルールを設定します。

- OMS にパブリックに解決可能なホスト名がある場合、OMS から Management Agent に接続するには、セキュリティグループに OMS アドレスを追加します。セキュリティグループには、DB リスナーポートと Management Agent ポートへのアクセスを許可するインバウンドルールが必要です。セキュリティの作成とインバウンドルールの追加の例については、「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。
- OMS にパブリックに解決可能なホスト名がない場合、OMS から Management Agent に接続するには、以下のいずれかを使用します。
  - OMS がプライベート VPC の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでホストされている場合、VPC ペアリングを設定して OMS から Management Agent に接続できます。詳細については、「VPC 内の DB インスタンスに別の VPC 内の EC2 インスタンスからアクセスする」を参照してください。
  - OMS がオンプレミスでホストされている場合、VPN 接続を設定して OMS から Management Agent へのアクセスを許可できます。VPN 接続の詳細については、「インターネット経由でクライアントアプリケーションから VPC 内の DB インスタンスにアクセスする」または「VPN 接続」を参照してください。
- OEM Management Agent バージョン 13.5.0.0 (v1-v3) または 24.1.0.0.v1 を 24.1 OMS ホストに接続するには、設定オプション `TLSv1.2` で TLS 1.2 プロトコルを使用するように `MINIMUM_TLS_VERSION` を設定します。

Management Agent の制限

Management Agent の使用にあたってのいくつかの制限を以下に挙げます。

- カスタム Oracle Management Agent イメージを指定することはできません。
- ジョブの実行やデータベースのパッチなど、ホスト認証情報を必要とする管理タスクはサポートされません。

- ホストメトリクスおよびプロセスリストが実際のシステムの状態を反映しているかは、保証されていません。したがって、OEM を使用してルートファイルシステムまたはマウントポイントファイルシステムをモニタリングしないでください。オペレーティングシステムのモニタリングの詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。
- 自動検出はサポートされていません。手動でデータベースターゲットを追加する必要があります。
- OMS モジュールの可用性はデータベースのエディションによって異なります。例えば、データベースのパフォーマンス診断およびモジュール調整は、Oracle Database Enterprise Edition でのみ使用できます。
- Management Agent は増設メモリとコンピューティングリソースを消費します。OEM_AGENT オプションを有効にしてパフォーマンスの問題が発生する場合、DB インスタンスのクラスをスケールアップすることをお勧めします。詳細については、「DB インスタンスクラス」および「Amazon RDS DB インスタンスを変更する」を参照してください。
- Amazon RDS ホストで OEM_AGENT を実行しているユーザーには、アラートログへのオペレーティングシステムアクセス権がありません。したがって、OEM の DB Alert Log と DB Alert Log Error Status のメトリクスを収集することはできません。

Management Agent のオプション設定

Amazon RDS は Management Agent オプションの次の設定をサポートします。

> **Note**
>
> Oracle Database 26ai の場合、Management Agent オプションには次の要件があります。
>
> - サポートされている Management Agent のバージョンは 24.1.0.0.v1 のみです。
> - 以下の TLS 暗号化スイートのみがサポートされています。
>   - TLS_RSA_WITH_AES_256_CBC_SHA256
>   - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
>   - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
>   - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

オプション設定	必須	有効な値	説明
バージョン (AGENT_VERSION)	はい	24.1.0.0.v1 13.5.0.0.v3 13.5.0.0.v2 13.5.0.0.v1 13.4.0.9.v2 13.4.0.9.v1 13.3.0.0.v2 13.3.0.0.v1 13.2.0.0.v3 13.2.0.0.v2 13.2.0.0.v1 13.1.0.0.v1	Management Agent ソフトウェアのバージョン。サポートされる最小のバージョンは、13.1.0.0.v1 です。 AWS CLI オプション名は OptionVersion です。 **Note** AWS GovCloud (US) リージョンでは、13.1 バージョンは利用できません。 **Note** Oracle Database 26ai では、サポートされているエージェントバージョンは 24.1.0.0.v1 のみです。
ポート (AGENT_PORT)	はい	整数値	OMS ホストをリッスンする DB インスタンスのポート。デフォルトは 3872 で

オプション設定	必須	有効な値	説明
			す。OMS ホストは、このポートへのアクセス権限があるセキュリティグループに属さなければなりません。 AWS CLI オプション名は `Port` です。
セキュリティグループ	はい	既存のセキュリティグループ	[Port] へのアクセス権限を持つセキュリティグループ。OMS ホストは、このセキュリティグループに属さなければなりません。 AWS CLI オプション名は `VpcSecurityGroupMemberships` または `DBSecurityGroupMemberships` です。
OMS_HOST	はい	文字列値、例えば *`my.example.oms`*	OMS のパブリックに解決可能なホスト名または IP アドレスです。 AWS CLI オプション名は `OMS_HOST` です。
OMS_PORT	はい	整数値	Management Agent をリッスンする OMS ホストの HTTPS アップロードポート。 HTTPS アップロードポートを確認するには、OMS ホストに接続し、次のコマンドを実行します (SYSMAN のパスワードが必要)。 `emctl status oms -details` AWS CLI オプション名は `OMS_PORT` です。

オプション設定	必須	有効な値	説明
AGENT_REGISTRATION_PASSWORD	はい	文字列値	Management Agent が OMS に対して自身を認証するために使用するパスワード。OEM_AGENT オプションを有効化する前に、OMS に永続的なパスワードを作成することをお勧めします。永続的なパスワードを使用すると、複数の Amazon RDS データベース間で、Management Agent オプショングループを共有できます。 AWS CLI オプション名は AGENT_REGISTRATION_PASSWORD です。
ALLOW_TLS_ONLY	いいえ	true、false (デフォルト値)	エージェントがサーバーとしてリッスンしている間 TLSv1 プロトコルのみをサポートするように、OEM エージェントを構成する値。この設定はサポートされなくなりました。Management Agent バージョン 13.1.0.0.v1 以降では、デフォルトで Transport Layer Security (TLS) がサポートされています。
MINIMUM_TLS_VERSION	いいえ	TLSv1 (デフォルト)TLSv1.2	エージェントがサーバーとしてリッスンしている間 OEM エージェントがサポートする、最小 TLS バージョンを指定する値。サポートされなくなったエージェントバージョンでは、TLSv1 設定のみがサポートされています。 13.5.0.0 (v1-v3) または 24.1.0.0.v1 を 24.1 OMS ホストに接続するには、これを TLSv1.2 に設定します。

オプション設定	必須	有効な値	説明
TLS_CIPHER_SUITE	いいえ	「Management Agent のオプション設定」を参照してください。	エージェントがサーバーとしてリッスンしている間 OEM エージェントによって使用される、TLS 暗号スイートを指定する値。

次の表は、Management Agent オプションがサポートする TLS 暗号スイートの一覧です。

暗号スイート	サポートされる Agent のバージョン	FedRAMP 準拠	サポートされている Oracle バージョン
TLS_RSA_WITH_AES_128_CBC_SHA	すべて	いいえ	19c、21c
TLS_RSA_WITH_AES_128_CBC_SHA256	13.1.0.0.v1 以降	いいえ	19c、21c
TLS_RSA_WITH_AES_256_CBC_SHA	13.2.0.0.v3 以降	いいえ	19c、21c
TLS_RSA_WITH_AES_256_CBC_SHA256	13.2.0.0.v3 以降	いいえ	19c、21c、26ai
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	13.2.0.0.v3 以降	はい	19c、21c
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	13.2.0.0.v3 以降	はい	19c、21c
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	13.2.0.0.v3 以降	はい	19c、21c
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	13.2.0.0.v3 以降	はい	19c、21c、26ai

暗号スイート	サポートされる Agent のバージョン	FedRAMP 準拠	サポートされている Oracle バージョン
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	13.4.0.9.v1 以降	はい	19c、21c、26ai
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	13.4.0.9.v1 以降	はい	19c、21c、26ai

暗号スイートと証明書の互換性

RDS for Oracle は、RSA 証明書と楕円曲線デジタル署名アルゴリズム (ECDSA) 証明書の両方をサポートしています。DB インスタンスの OEM エージェントオプションを設定するときは、`TLS_CIPHER_SUITE` オプション設定で指定した暗号スイートが DB インスタンスで使用される証明書タイプと互換性があることを確認する必要があります。

次の表は、証明書タイプと暗号スイート間の互換性を示しています。

[証明書タイプ]	互換性のある暗号スイート	互換性のない暗号スイート
RSA 証明書 (rds-ca-2019、rds-ca-rsa2048-g1、rds-ca-rsa4096-g1)	TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

[証明書タイプ]	互換性のある暗号スイート	互換性のない暗号スイート
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	
ECDSA 証明書 (rds-ca-ecc384-g1)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

TLS_CIPHER_SUITE オプション設定で暗号スイートを指定するときは、DB インスタンスで使用される証明書タイプと互換性があることを確認してください。DB インスタンスの証明書タイプと互換性のない暗号スイートを含む OEM エージェントオプションにオプショングループを関連付けようとすると、オペレーションは失敗し、非互換性を示すエラーメッセージが表示されます。

Management Agent オプションを DB インスタンス用に有効にする

Management Agent オプションを有効にするには、次の手順を実行します。

トピック

- ステップ 1: DB インスタンスに Management Agent オプションを追加する
- ステップ 2: DBSNMP ユーザーアカウントのロックを解除する
- ステップ 3: Management Agent コンソールにターゲットを追加する

ステップ 1: DB インスタンスに Management Agent オプションを追加する

Management Agent オプションを DB インスタンスに追加するには、次の手順を実行します。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

エラーが発生した場合は、特定の問題の解決に関する情報について、My Oracle Support のドキュメントを確認します。

Management Agent オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに OEM Agent がアクティブになります。

OMS ホストで信頼できないサードパーティーの証明書が使用されている場合は、Amazon RDS より次のエラーが返ります。

```
You successfully installed the OEM_AGENT option. Your OMS host is using an untrusted
 third party certificate.
Configure your OMS host with the trusted certificates from your third party.
```

このエラーが返った場合、Management Agent オプションは問題が解決するまで有効になりません。問題の修正については、My Oracle Support ドキュメント「2202569.1」を参照してください。

コンソール

Management Agent オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

a. [Engine] で DB インスタンスの Oracle エディションを選択します。

b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [OEM_AGENT] オプションを追加し、オプションを設定します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。各設定の詳細については、「Management Agent のオプション設定」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

- 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

次の例では、AWS CLI の add-option-to-option-group コマンドを使用して、OEM_AGENT オプションを myoptiongroup オプショングループに追加しています。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options
 OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-123456
{Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options
 OptionName=OEM_AGENT,OptionVersion=13.1.0.0.v1,Port=3872,VpcSecurityGroupMemberships=sg-123456
{Name=OMS_PORT,Value=4903},{Name=AGENT_REGISTRATION_PASSWORD,Value=password}] ^
```

```
    --apply-immediately
```

ステップ 2: DBSNMP ユーザーアカウントのロックを解除する

Management Agent は、DBSNMP ユーザーアカウントを使用してデータベースに接続し、問題を Oracle Enterprise Manager に報告します。CDB では、DBSNMP は共通ユーザーです。このユーザーアカウントは、Management Agent と OEM Database Express の両方に必要です。デフォルトでは、このアカウントはロックされています。このアカウントのロック解除手順は、データベースが非 CDB アーキテクチャを使用しているか CDB アーキテクチャを使用しているかによって異なります。

DBSNMP ユーザーアカウントのロックを解除するには

1. SQL*Plus または別の Oracle SQL アプリケーションで、マスターユーザーとして DB インスタンスにログインします。
2. データベースアーキテクチャに応じて、次のいずれかのアクションを実行します。

   データベースが非 CDB の場合

   次の SQL ステートメントを実行します。

   ```
   ALTER USER dbsnmp IDENTIFIED BY new_password ACCOUNT UNLOCK;
   ```

   データベースが CDB の場合

   次のストアドプロシージャを実行して、DBSNMP アカウントのロックを解除します。

   ```
   EXEC rdsadmin.rdsadmin_util.reset_oem_agent_password('new_password');
   ```

   プロシージャが存在しないことを示すエラーが表示された場合は、CDB インスタンスを再起動して自動的にインストールします。詳細については、「DB インスタンスの再起動」を参照してください。

ステップ 3: Management Agent コンソールにターゲットを追加する

DB インスタンスをターゲットとして追加するには、エンドポイントとポートがわかっていることを確認してください。Amazon RDS DB インスタンスのエンドポイントを見つける方法については、「RDS for Oracle DB インスタンスのエンドポイントを見つける」を参照してください。データベー

スが CDB アーキテクチャを使用している場合は、CDB$ROOT コンテナをターゲットとして個別に追加します。

Management Agent コンソールにターゲットを追加するには

1. OMS コンソールで、[Setup]、[Add Target]、[Add Targets Manually] の順に選択します。
2. [Add Targets Declaratively by Specifying Target Monitoring Properties] を選択します。
3. [Target Type] で、[Database Instance] を選択します。
4. [Monitoring Agent (モニタリングエージェント)] で、RDS DB インスタンス識別子と同じ識別子のエージェントを選択します。
5. [Add Manually] を選択します。
6. Amazon RDS DB インスタンスのエンドポイントを入力するか、ホスト名リストから選択します。指定されたホスト名が Amazon RDS DB インスタンスのエンドポイントと一致することを確認します。
7. 次のデータベースのプロパティを指定します。
   - [ターゲット名] に、名前を入力します。
   - [Database system name (データベースシステム名)] に名前を入力します。
   - [Monitor username (モニターユーザーネーム)] に **dbsnmp** と入力します。
   - [モニターパスワード] に ステップ 2: DBSNMP ユーザーアカウントのロックを解除する のパスワードを入力します。
   - [ロール] に normal と入力します。
   - [Oracle home path (Oracle ホームパス)] に **/oracle** と入力します。
   - [Listener Machine name] には、エージェント識別子があらかじめ表示されます。
   - [Port (ポート)] にデータベースポートを入力します。RDS のデフォルトのポート番号は 1521 です。
   - [Database name (データベース名)] には、データベースの名前を入力します。データベースが CDB の場合、この名前は RDSCDB です。
8. [Test Connection] を選択します。
9. [次へ] を選択します。ターゲットデータベースは、モニタリング対象リソースのリストに表示されます。

Management Agent オプションの削除

DB インスタンスから OEM Agent を削除できます。OEM Agent を削除した後、DB インスタンスを再起動する必要はありません。

OEM Agent を DB インスタンスから削除するには、次のいずれかを実行します。

- OEM Agent オプションを、所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、OEM Agent オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Management Agent を使用したデータベースタスクの実行

Amazon RDS 手順を使用して、Management Agent で特定の EMCTL コマンドを実行できます。これらの手順を実行すると、以下のタスクを実行できます。

> **Note**
>
> タスクは非同期的に実行されます。

タスク

- Management Agent を保護する
- Management Agent のステータスの取得
- Management Agent の再起動
- Management Agent でモニタリングするターゲットのリスト化
- Management Agent でモニタリングする取集スレッドのリスト化
- Management Agent のステータスの削除
- Management Agent に OMS をアップロードさせる
- OMS への ping
- 実行中のタスクのステータスの表示

Management Agent を保護する

Management Agent を保護するには、Amazon RDS 手順 `rdsadmin.rdsadmin_oem_agent_tasks.secure_oem_agent` を実行します。この手順は、`emctl secure agent` コマンドの実行に相当します。

次の手順では、管理エージェントを保護するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.secure_oem_agent as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent のステータスの取得

Management Agent のステータスを取得するには、Amazon RDS のプロシージャ `rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent` を実行します。このプロシージャは、`emctl status agent` コマンドに相当します。

次の手順では、管理エージェントのステータスを取得するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent の再起動

Management Agent を再起動するには、Amazon RDS 手順 `rdsadmin.rdsadmin_oem_agent_tasks.restart_oem_agent` を実行します。この手順は、`emctl stop agent` および `emctl start agent` コマンドの実行に相当します。

次の手順では、管理エージェントを再起動するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.restart_oem_agent as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent でモニタリングするターゲットのリスト化

Management Agent によってモニタリングされるターゲットをリスト化するには、Amazon RDS 手順 `rdsadmin.rdsadmin_oem_agent_tasks.list_targets_oem_agent` を実行します。この手順は、`emctl config agent listtargets` コマンドの実行に相当します。

次の手順では、管理エージェントによってモニタリングされるターゲットを一覧表示するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.list_targets_oem_agent as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent でモニタリングする取集スレッドのリスト化

管理エージェントによってモニタリングされる、実行中、準備完了、スケジュール済みの収集スレッドをすべて一覧表示するには、Amazon RDS プロシージャ `rdsadmin.rdsadmin_oem_agent_tasks.list_clxn_threads_oem_agent` を実行します。このプロシージャは、`emctl status agent scheduler` コマンドに相当します。

次の手順では、収集スレッドを一覧表示するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.list_clxn_threads_oem_agent() as TASK_ID from
 DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent のステータスの削除

Management Agent のステータスを削除するには、Amazon RDS の手順 `rdsadmin.rdsadmin_oem_agent_tasks.clearstate_oem_agent` を実行します。この手順は、`emctl clearstate agent` コマンドの実行に相当します。

次の手順では、管理エージェントの状態をクリアするタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.clearstate_oem_agent() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

Management Agent に OMS をアップロードさせる

Management Agent に関連付けられた Oracle Management Server (OMS) をアップロードするには、Amazon RDS のプロシージャ `rdsadmin.rdsadmin_oem_agent_tasks.upload_oem_agent` を実行します。この手順は、`emctl upload agent` コマンドの実行に相当します。

次の手順では、関連付けられている OMS を管理エージェントにアップロードさせるタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.upload_oem_agent() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

OMS への ping

Management Agent の OMS を ping するには、Amazon RDS の手順 `rdsadmin.rdsadmin_oem_agent_tasks.ping_oms_oem_agent` を実行します。この手順は、`emctl pingOMS` コマンドの実行に相当します。

次の手順では、管理エージェントの OMS に ping を送信するタスクを作成し、タスクの ID を返します。

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.ping_oms_oem_agent() as TASK_ID from DUAL;
```

タスクの出力ファイルを表示し、結果を確認するには、「実行中のタスクのステータスの表示」を参照してください。

実行中のタスクのステータスの表示

実行中のタスクのステータスは bdump ファイルで確認できます。bdump ファイルは `/rdsdbdata/log/trace` ディレクトリにあります。bdump ファイルの名前形式は、以下のとおりです。

```
dbtask-task-id.log
```

タスクをモニタリングしたい場合は、*`task-id`* を、モニタリングするタスクの ID に置き換えます。

bdump ファイルの内容を表示するには、Amazon RDS 手順 `rdsadmin.rds_file_util.read_text_file` を実行します。次のクエリは、`dbtask-1546988886389-2444.log` bdump ファイルの中身を返します。

```
SELECT text FROM
 table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1546988886389-2444.log'));
```

このAmazon RDS 手順 `rdsadmin.rds_file_util.read_text_file` の詳細については、「DB インスタンスディレクトリ内のファイルの読み取り」を参照してください。

Oracle Label Security

Amazon RDS は OLS オプションの使用を通じて Oracle Database の Oracle Enterprise Edition 用の Oracle Label Security をサポートします。

ほとんどのデータベースセキュリティでは、オブジェクトレベルでアクセスを制御します。Oracle Label Security は、個別のテーブル行へのアクセスのきめ細かい制御を提供します。例えば、Label Security を使用して、ポリシーベースの管理モデルで規制コンプライアンスを適用できます。Label Security ポリシーを使用して機密データへのアクセスを制御し、適切なクリアランスレベルを持つユーザーのみにアクセスを制限できます。詳細については、Oracle ドキュメントの「Introduction to Oracle Label Security」を参照してください。

トピック

- Oracle Label Security の要件
- Oracle Label Security を使用する際の考慮事項
- Oracle Label Security オプションの追加
- トラブルシューティング

Oracle Label Security の要件

Oracle Label Security に関する次の要件を理解してください。

- DB インスタンスでは Bring-Your-Own-License モデルを使用する必要があります。詳細については、「RDS for Oracle のライセンスオプション」を参照してください。
- Oracle Enterprise Edition の有効なライセンスと、ソフトウェア更新ライセンスおよびサポートが必要です。
- Oracle ライセンスには、Label Security オプションが必要です。

Oracle Label Security を使用する際の考慮事項

Oracle Label Security を使用するには、テーブルの特定の行へのアクセスを制御するポリシーを作成します。詳細については、Oracle ドキュメントの「Creating an Oracle Label Security Policy」を参照してください。

以下の点を考慮してください。

- Oracle Label Security は、固定かつ永続オプションです。このオプションは固定的なものであり、オプショングループから削除することはできません。Oracle Label Security をオプショングループに追加して DB インスタンスに関連付けた場合、後で DB インスタンスに別のオプショングループを関連付けることはできますが、このグループには Oracle Label Security オプションも含める必要があります。
- Label Security を操作する場合、すべてのアクションを LBAC_DBA ロールとして実行します。DB インスタンスのマスターユーザーには LBAC_DBA ロールが付与されます。他のユーザーが Label Security ポリシーを管理できるように、LBAC_DBA ロールを他のユーザーに付与できます。
- Oracle Label Security へのアクセスが必要な新規ユーザーに OLS_ENFORCEMENT パッケージへのアクセス権を付与するようにしてください。OLS_ENFORCEMENT パッケージへのアクセス権を付与するには、DB インスタンスにマスターユーザーとして接続し、次の SQL ステートメントを実行します。

```
GRANT ALL ON LBACSYS.OLS_ENFORCEMENT TO username;
```

- Oracle Enterprise Manager (OEM) Cloud Control を使用して、Label Security を設定することができます。Amazon RDS は Management Agent オプションを通じて OEM Cloud Control をサポートします。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。

Oracle Label Security オプションの追加

DB インスタンスに Oracle Label Security オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループにオプションを追加します。

> **Important**
>
> Oracle Label Security は、固定かつ永続オプションです。

3. オプショングループを DB インスタンスに関連付けます。

Label Security オプションの追加後、オプショングループがアクティブになるとすぐに、Label Security がアクティブになります。

Label Security オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] で、[oracle-ee] を選択します。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [OLS] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。

   > **Important**
   >
   > すでに 1 つ以上の DB インスタンスにアタッチされている既存のオプショングループに Label Security を追加すると、すべての DB インスタンスが再起動されます。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。既存の DB インスタンスに Label Security オプションを追加すると、DB インスタンスを自動的に再起動している間に短い停止が発生します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

トラブルシューティング

Oracle Label Security を使用するときに発生する可能性のある問題を次に示します。

問題	トラブルシューティングの推奨事項
ポリシーを作成しようとすると、次のようなエラーメッセージが表示されま	Oracle の Label Security 機能の既知の問題により、16 ~ 24 文字のユーザーネームを持つユーザー

問題	トラブルシューティングの推奨事項
す: `insufficient authorization for the SYSDBA package`。	は、Label Security のコマンドを実行することができません。文字数が異なる新しいユーザーを作成し、LBAC_DBA を新しいユーザーに付与し、新しいユーザーとしてログインして、新しいユーザーとして OLS コマンドを実行します。詳細については、Oracle サポートにお問い合わせください。

Oracle Locator

Amazon RDS は、LOCATOR オプションを使用することで Oracle Locator をサポートします。Oracle Locator は、インターネットとワイヤレスベースのアプリケーションをサポートするために一般的に必要な機能とパートナーベースの GIS ソリューションを提供します。Oracle Locator は Oracle Spatial の制限付きのサブネットです。詳細については、Oracle ドキュメントの Oracle Locator を参照してください。

> **Important**
>
> Oracle Locator を使用する際、共通脆弱性評価システム (CVSS) のスコアが 9 以上、またはそのほかのセキュリティ脆弱性の報告によりセキュリティ脆弱性がある場合に、Amazon RDS は自動的に DB インスタンスを最新の Oracle PSU にアップデートします。

サポートされている Oracle Locator のデータベースリリース

RDS for Oracle は、Oracle Database 19c について Oracle Locator をサポートしています。Oracle Locator は Oracle Database 21c 以降ではサポートされていませんが、その機能は Oracle Spatial オプションで使用できます。以前は、Spatial オプションには追加のライセンスが必要でした。Oracle Locator は Oracle Spatial 機能のサブセットの 1 つであり、追加のライセンスは必要ありませんでした。2019 年、Oracle は Oracle Spatial のすべての機能が Enterprise Edition と Standard Edition 2 のライセンスに追加費用なしで含まれたことを発表しました。その結果、Oracle Spatial オプションに追加のライセンスは必要なくなりました。詳細については、Oracle Database Insider ブログの Machine Learning, Spatial and Graph - No License Required! (機械学習、Spatial と Graph のライセンスが不要に) を参照してください。

Oracle Locator の前提条件

Oracle Locator を使用するための前提条件は次のとおりです。

- DB インスタンスが十分なクラスである必要があります。Oracle Locator は、db.t3.small DB インスタンスクラスではサポートされていません。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。
- DB インスタンスで [マイナーバージョン自動アップグレード] が有効になっている必要があります。このオプションにより、DB インスタンスは、(それが利用可能になった時点で) DB エンジンのマイナーバージョンアップグレードを自動的に受信できるようになります。これは、Oracle の Java 仮想マシン (JVM) をインストールする、すべてのオプションに必須です。Amazon RDS

では、このオプションを使用して、DB インスタンスに対し最新の Oracle パッチセット更新 (PSU)、またはリリース更新 (RU) を行います。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Locator のベストプラクティス

Oracle Locator を使用するためのベストプラクティスは次のとおりです。

- セキュリティを最大にするためには、LOCATOR オプションを Secure Sockets Layer (SSL) で使用します。詳細については、「Oracle Secure Sockets Layer」を参照してください。
- DB インスタンスへのアクセスを制限するように、DB インスタンスを設定します。詳細については、「VPC の DB インスタンスにアクセスするシナリオ」および「VPC 内の DB インスタンスの使用」を参照してください。

Oracle Locator オプションの追加

DB インスタンスに LOCATOR オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

DB インスタンスに Oracle Java Virtual Machine (JVM) がインストールされていない場合、LOCATOR オプションが追加されている間は短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに Oracle Locator が使用可能となります。

> **Note**
>
> この停止中、パスワード検証機能は一時的に無効になります。また、停止中にパスワード検証機能に関連するイベントを確認することもできます。Oracle DB インスタンスが使用可能になる前に、パスワード検証機能が再び有効になります。

LOCATOR オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [エンジン] で DB インスタンスの Oracle エディションを選択します。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [LOCATOR] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Locator を使用する

Oracle Locator オプションを有効にしたら、使用をスタートできます。Oracle Locator の機能のみ使用する必要があります。Oracle Spatial のライセンスがない場合には、Oracle Spatial の機能は使用しないでください。

Oracle Locator でサポートされている機能の一覧については、Oracle ドキュメントで Features Included with Locator を参照してください。

Oracle Locator でサポートされていない機能の一覧については、Oracle ドキュメントで Features Not Included with Locator を参照してください。

Oracle Locator オプションの削除

LOCATOR オプションが提供するデータ型を使用するすべてのオブジェクトを削除したら、そのオプションを DB インスタンスから削除できます。DB インスタンスに Oracle Java Virtual Machine

(JVM) がインストールされていない場合、LOCATOR オプションの削除中にサービスが短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。LOCATOR オプションを削除した後に DB インスタンスを再起動する必要はありません。

LOCATOR オプションを削除するには

1. データをバックアップします。

   > **Warning**
   >
   > オプションの一部として有効化されたデータ型がインスタンスで使用されている場合、LOCATOR オプションを削除すると、データが失われる可能性があります。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

2. 既存のオブジェクトが、LOCATOR オプションのデータ型や機能を参照しているかどうかを確認します。

   LOCATOR オプションが存在する場合、LOCATOR オプションを持たない新しいオプショングループを適用すると、インスタンスが停止することがあります。次のクエリを使用して、オブジェクトを識別できます。

   ```
   SELECT OWNER, SEGMENT_NAME, TABLESPACE_NAME, BYTES/1024/1024 mbytes
   FROM   DBA_SEGMENTS
   WHERE  SEGMENT_TYPE LIKE '%TABLE%'
   AND    (OWNER, SEGMENT_NAME) IN
          (SELECT DISTINCT OWNER, TABLE_NAME
           FROM   DBA_TAB_COLUMNS
           WHERE  DATA_TYPE='SDO_GEOMETRY'
           AND    OWNER <> 'MDSYS')
   ORDER BY 1,2,3,4;

   SELECT OWNER, TABLE_NAME, COLUMN_NAME
   FROM   DBA_TAB_COLUMNS
   WHERE  DATA_TYPE = 'SDO_GEOMETRY'
   AND    OWNER <> 'MDSYS'
   ORDER BY 1,2,3;
   ```

3. LOCATOR オプションのデータ型や機能を参照するすべてのオブジェクトを削除します。
4. 次のいずれかを行ってください。

- LOCATOR オプションを所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- DB インスタンスを修正して、LOCATOR オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle ネイティブネットワーク暗号化

Amazon RDS は、Oracle ネイティブネットワーク暗号化 (NNE) をサポートしています。NATIVE_NETWORK_ENCRYPTION オプションを使用すると、DB インスタンスとの間でデータを移動するときにデータを暗号化できます。Amazon RDS では、Oracle Database のすべてのエディションの NNE がサポートされています。

Oracle ネイティブネットワーク暗号化の詳細な説明はこのガイドでは取り上げませんが、配置で使用するソリューションを決定する前に各アルゴリズムおよびキーの長所と短所を理解する必要があります。Oracle ネイティブネットワークの暗号化で使用できるアルゴリズムとキーについては、Oracle ドキュメントの「ネットワークデータ暗号化と整合性の設定」を参照してください。AWS セキュリティの詳細については、AWS セキュリティセンターを参照してください。

> **Note**
>
> ネイティブネットワーク暗号化または Secure Sockets Layer を使用できますが、両方を使用することはできません。詳細については、「Oracle Secure Sockets Layer」を参照してください。

トピック

- NATIVE_NETWORK_ENCRYPTION オプション設定
- NATIVE_NETWORK_ENCRYPTION オプションの追加
- NNE がアクティブであることを確認する
- sqlnet.ora で NNE 値を設定する
- NATIVE_NETWORK_ENCRYPTION オプション設定
- NATIVE_NETWORK_ENCRYPTION オプションの削除

NATIVE_NETWORK_ENCRYPTION オプション設定

暗号化要件は、サーバーとクライアントの両方で指定できます。DB インスタンスは、データベースリンクを使用して別のデータベースに接続する場合などに、クライアントとして機能します。サーバー側で暗号化の強制を回避できます。例えば、サーバーで必要としているからといって、すべてのクライアント通信で暗号化の使用を強制することはありません。この場合、SQLNET.*CLIENT オプションを使用してクライアント側で暗号化を強制できます。

Amazon RDS は、NATIVE_NETWORK_ENCRYPTION オプションの次の設定をサポートします。

> **Note**
>
> オプション設定の値をコンマで区切る場合は、コンマの後にスペースを挿入しないでください。

> **Note**
>
> Oracle Database 26ai の場合、Amazon RDS は次のオプション設定に対して限定された一連のアルゴリズムをサポートします。
>
> - SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT および SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER の場合、SHA256、SHA384、および SHA512 のみがサポートされます。
> - SQLNET.ENCRYPTION_TYPES_CLIENT および SQLNET.ENCRYPTION_TYPES_SERVER の場合、AES256 および AES192 のみがサポートされます。

オプション設定	有効な値	デフォルト値	説明
SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS	TRUE, FALSE	TRUE	非セキュア暗号を使用しているクライアントでデータベースへの接続を試行したときのサーバーの動作。TRUE の場合、2021 年 7 月 PSU パッチが適用されていなくても、クライアントでの接続が可能です。 設定を FALSE にすると、クライアントで 2021 年 7 月 PSU パッチが適用された場合にだけ、データベースへの接続が可能です。SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS を FALSE に設定する前に、次の条件が満たされていることを確認してください。

オプション設定	有効な値	デフォルト値	説明
			• SQLNET.ENCRYPTION_TYPES_SERVER と SQLNET.ENCRYPTION_TYPES_CLIENT には、DES、3DES、または RC4 (すべてのキーの長さ) とは異なる、一致する暗号化方式が 1 つあります。 • SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER と SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT には、MD5 とは異なる、一致するセキュアなチェックサム方式が 1 つあります。 • クライアントで 2021 年 7 月 PSU パッチが適用されます。クライアントにパッチが適用されていない場合、クライアントでの接続が失われ、ORA-12269 エラーを受け取ります。

オプション設定	有効な値	デフォルト値	説明
SQLNET.ALLOW_WEAK_CRYPTO	TRUE, FALSE	TRUE	非セキュア暗号を使用しているクライアントでデータベースへの接続を試行したときのサーバーの動作。次の暗号はセキュアではないと見なされます。 • DES 暗号化方式 (すべてのキーの長さ) • 3DES 暗号化方式 (すべてのキーの長さ) • RC4 暗号化方式 (すべてのキーの長さ) • MD5 チェックサム方式 設定を TRUE にすると、クライアントで前述の非セキュア暗号を使用した場合に接続できます。 設定を FALSE にすると、クライアントで前述の非セキュア暗号を使用した場合にデータベースが接続を阻止します。SQLNET.ALLOW_WEAK_CRYPTO を FALSE に設定する前に、次の条件が満たされていることを確認してください。 • SQLNET.ENCRYPTION_TYPES_SERVER と SQLNET.ENCRYPTION_TYPES_CLIENT には、DES、3DES、または RC4 (すべてのキーの長さ) とは異なる、一致する暗号化方式が 1 つあります。 • SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER

オプション設定	有効な値	デフォルト値	説明
			と SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT には、MD5 とは異なる、一致するセキュアなチェックサム方式が 1 つあります。 • クライアントで 2021 年 7 月 PSU パッチが適用されます。クライアントにパッチが適用されていない場合、クライアントでの接続が失われ、ORA-12269 エラーを受け取ります。
SQLNET.CRYPTO_CHECKSUM_CLIENT	Accepted Rejected Requested , Required	Requested	クライアント、またはクライアントとして機能しているサーバーが DB インスタンスに接続する場合のデータ整合性動作。DB インスタンスでデータベースリンクが使用されている場合、そのインスタンスはクライアントとして機能します。 Requested は、DB インスタンスによるチェックサムの実行をクライアントで必要としないことを示しています。

オプション設定	有効な値	デフォルト値	説明
SQLNET.CRYPTO_CHECKSUM_SERVER	Accepted, Rejected, Requested, Required	Requested	クライアント、またはクライアントとして機能しているサーバーが DB インスタンスに接続する場合のデータ整合性動作。DB インスタンスでデータベースリンクが使用されている場合、そのインスタンスはクライアントとして機能します。 Requested は、クライアントによるチェックサムの実行を DB インスタンスで必要としないことを示しています。
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	SHA256, SHA384, SHA512, SHA1, MD5	SHA256, SHA384, SHA512	チェックサムアルゴリズムのリスト。 1 つの値を指定することも、カンマで区切られた値リストを指定することもできます。コンマを使用する場合は、コンマの後にスペースを挿入しないでください。スペースを挿入すると、InvalidParameterValue エラーが発生します。 このパラメータと SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER には、共通の暗号が必要です。 **Note** Oracle Database 26ai では、SHA256、SHA384、および SHA512 のみがサポートされています。

オプション設定	有効な値	デフォルト値	説明
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA256, SHA384, SHA512, SHA1, MD5	SHA256, SHA384, SHA512, SHA1, MD5	チェックサムアルゴリズムのリスト。 1 つの値を指定することも、カンマで区切られた値リストを指定することもできます。コンマを使用する場合は、コンマの後にスペースを挿入しないでください。スペースを挿入すると、InvalidParameterValue エラーが発生します。 このパラメータと SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT には、共通の暗号が必要です。 **Note** Oracle Database 26ai では、SHA256、SHA384、および SHA512 のみがサポートされています。
SQLNET.ENCRYPTION_CLIENT	Accepted Rejected Requested , Required	Requested	クライアント、またはクライアントとして機能しているサーバーが DB インスタンスに接続する場合のクライアントの暗号化動作。DB インスタンスでデータベースリンクが使用されている場合、そのインスタンスはクライアントとして機能します。 Requested は、クライアントでサーバーからのトラフィックを暗号化する必要がないことを示しています。

オプション設定	有効な値	デフォルト値	説明
SQLNET.ENCRYPTION_SERVER	Accepted Rejected Requested , Required	Requested	クライアント、またはクライアントとして機能しているサーバーが DB インスタンスに接続する場合のサーバーの暗号化動作。DB インスタンスでデータベースリンクが使用されている場合、そのインスタンスはクライアントとして機能します。 Requested は、DB インスタンスでクライアントからのトラフィックを暗号化する必要がないことを示します。 **Important** デフォルト値の Requested では、クライアントが暗号化をサポートしていない場合、接続は暗号化されないままになる可能性があります。すべての接続に暗号化を適用するには、SQLNET.ENCRYPTION_SERVER を Required に設定します。

オプション設定	有効な値	デフォルト値	説明
SQLNET.ENCRYPTION_TYPES_CLIENT	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	クライアントによって使用される暗号化アルゴリズムのリスト。クライアントは、各アルゴリズムが順序どおりに使用され、アルゴリズムが成功するか、リストの末尾に到達するまでサーバー入力の復号を試みます。 Amazon RDS は、Oracle による次のデフォルトリストを使用します。RDS は RC4_256 で始まり、リストの下の方へ順番に進みます。順序を変更したり、DB インスタンスで受け入れられるアルゴリズムを制限したりすることができます。 1. RC4_256: RSA RC4 (256 ビットのキーサイズ) 2. AES256: AES (256 ビットのキーサイズ) 3. AES192: AES (192 ビットのキーサイズ) 4. 3DES168: 3-key Triple-DES (112 ビットの有効キーサイズ) 5. RC4_128: RSA RC4 (128 ビットのキーサイズ) 6. AES128: AES (128 ビットのキーサイズ) 7. 3DES112: 2-key Triple-DES (80 ビットの有効キーサイズ) 8. RC4_56: RSA RC4 (56 ビットのキーサイズ)

オプション設定	有効な値	デフォルト値	説明
			9. DES: Standard DES (56 ビットのキーサイズ) 10RC4_40: RSA RC4 (40 ビットのキーサイズ) 11DES40: DES40 (40 ビットのキーサイズ) 1 つの値を指定することも、カンマで区切られた値リストを指定することもできます。コンマを使用する場合は、コンマの後にスペースを挿入しないでください。スペースを挿入すると、InvalidParameterValue エラーが発生します。 このパラメータと SQLNET.ENCRYPTION_TYPES_SERVER には、共通の暗号が必要です。 **Note** Oracle Database 26ai では、AES256 および AES192 のみがサポートされています。

オプション設定	有効な値	デフォルト値	説明
SQLNET.ENCRYPTION_TYPES_SERVER	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	RC4_256, AES256, AES192, 3DES168, RC4_128, AES128, 3DES112, RC4_56, DES, RC4_40, DES40	DB インスタンスによって使用された暗号化アルゴリズムのリスト。DB インスタンスは、各アルゴリズムを順序どおりに使用し、アルゴリズムが成功するか、リストの末尾に到達するまでクライアントの復号を試みます。 Amazon RDS は、Oracle による次のデフォルトリストを使用します。順序を変更したり、クライアントで受け入れられるアルゴリズムを制限したりすることができます。 1. RC4_256: RSA RC4 (256 ビットのキーサイズ) 2. AES256: AES (256 ビットのキーサイズ) 3. AES192: AES (192 ビットのキーサイズ) 4. 3DES168: 3-key Triple-DES (112 ビットの有効キーサイズ) 5. RC4_128: RSA RC4 (128 ビットのキーサイズ) 6. AES128: AES (128 ビットのキーサイズ) 7. 3DES112: 2-key Triple-DES (80 ビットの有効キーサイズ) 8. RC4_56: RSA RC4 (56 ビットのキーサイズ) 9. DES: Standard DES (56 ビットのキーサイズ)

オプション設定	有効な値	デフォルト値	説明
			10RC4_40: RSA RC4 (40 ビットのキーサイズ) 11DES40: DES40 (40 ビットのキーサイズ) 1 つの値を指定することも、カンマで区切られた値リストを指定することもできます。コンマを使用する場合は、コンマの後にスペースを挿入しないでください。スペースを挿入すると、InvalidParameterValue エラーが発生します。 このパラメータと SQLNET.ENCRYPTION_TYPES_CLIENT には、共通の暗号が必要です。 **Note** Oracle Database 26ai では、AES256 および AES192 のみがサポートされています。

NATIVE_NETWORK_ENCRYPTION オプションの追加

DB インスタンスに NATIVE_NETWORK_ENCRYPTION オプションを追加する一般的なプロセスは以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

オプショングループがアクティブになると、NNE がアクティブになります。

AWS マネジメントコンソール を使用して DB インスタンスに NATIVE_NETWORK_ENCRYPTION オプションを追加するには

1. [Engine] で、使用する Oracle のエディションを選択します。NNE はすべてのエディションでサポートされます。
2. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
3. NATIVE_NETWORK_ENCRYPTION オプションをオプショングループに追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。

   > **Note**
   >
   > NATIVE_NETWORK_ENCRYPTION オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに NNE がアクティブになります。
4. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。NATIVE_NETWORK_ENCRYPTION オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに NNE がアクティブになります。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

NNE がアクティブであることを確認する

DB インスタンスに接続したら、セッションの V$SESSION_CONNECT_INFO をクエリして、ネイティブネットワーク暗号化がアクティブであることを確認します。

```
SELECT NETWORK_SERVICE_BANNER
FROM V$SESSION_CONNECT_INFO
WHERE SID = SYS_CONTEXT('USERENV', 'SID');
```

NNE がアクティブな場合、結果には encryption service adapter を含むバナーが含まれます。バナーテキストには、AES256 などのネゴシエートされたアルゴリズムが表示されます。チェックサムも有効になっている場合、結果には crypto-checksumming service adapter バナーが含まれます。クエリの結果に暗号化サービスアダプタバナーが含まれていない場合、その接続ではネイティブネットワーク暗号化が使用されていません。

sqlnet.ora で NNE 値を設定する

Oracle ネイティブネットワーク暗号化を使用すると、サーバー側とクライアント側でネットワーク暗号化を設定できます。クライアントとは DB インスタンスへの接続に使用されるコンピュータです。slqnet.ora では、次のクライアント設定を指定できます。

- SQLNET.ALLOW_WEAK_CRYPTO
- SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS
- SQLNET.CRYPTO_CHECKSUM_CLIENT
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT
- SQLNET.ENCRYPTION_CLIENT
- SQLNET.ENCRYPTION_TYPES_CLIENT

詳細については、Oracle ドキュメントの「Oracle サーバーおよびクライアントのネットワークデータ暗号化と整合性の設定」を参照してください。

DB インスタンスでアプリケーションからの接続リクエストが拒否される場合があります。例えば、クライアント側の暗号化アルゴリズムとサーバー側の暗号化アルゴリズムが一致しない場合、拒否される可能性があります。Oracle ネイティブネットワーク暗号化をテストするには、クライアントの sqlnet.ora ファイルに次の行を追加します。

```
DIAG_ADR_ENABLED=off
TRACE_DIRECTORY_CLIENT=/tmp
TRACE_FILE_CLIENT=nettrace
TRACE_LEVEL_CLIENT=16
```

接続が試行されると、前の行により /tmp/nettrace* というトレースファイルがクライアントに生成されます。トレースファイルには、接続に関する情報が含まれています。Oracle ネイティブネットワーク暗号化を使用した場合の接続に関連する問題の詳細については、「Oracle Database ドキュメント」の「暗号化と整合性のネゴシエーションについて」を参照してください。

NATIVE_NETWORK_ENCRYPTION オプション設定

NATIVE_NETWORK_ENCRYPTION オプションを有効にすると、その設定を変更できます。現在、NATIVE_NETWORK_ENCRYPTION オプション設定は、AWS CLI または RDS API でのみ変更できます。 コンソールは使用できません。次の例では、オプションで 2 つの設定を変更します。

```
aws rds add-option-to-option-group \
    --option-group-name my-option-group \
    --options
 "OptionName=NATIVE_NETWORK_ENCRYPTION,OptionSettings=[{Name=SQLNET.CRYPTO_CHECKSUM_TYPES_SERVE
{Name=SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT,Value=SHA256}]" \
    --apply-immediately
```

CLI を使用したオプション設定の変更方法の詳細については、「AWS CLI」を参照してください。各設定の詳細については、「NATIVE_NETWORK_ENCRYPTION オプション設定」を参照してください。

トピック

- CRYPTO_CHECKSUM_* 値の変更
- ALLOW_WEAK_CRYPTO* 設定の変更

CRYPTO_CHECKSUM_* 値の変更

NATIVE_NETWORK_ENCRYPTION オプション設定を変更する場合は、次のオプション設定に共通の暗号が少なくとも 1 つあることを確認してください。

- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT

次の例は、SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER を変更するシナリオを示しています。CRYPTO_CHECKSUM_TYPES_CLIENT、CRYPTO_CHECKSUM_TYPES_SERVER の両方が SHA256 を使用するため、設定は有効です。

オプション設定	変更前の値	変更後の値
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	**SHA256**, SHA384, SHA512	変更なし

オプション設定	変更前の値	変更後の値
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	**SHA256**, SHA384, SHA512, SHA1, MD5	SHA1,MD5, **SHA256**

別の例では、SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER をデフォルト設定から SHA1,MD5 に変更するとします。この場合、SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT を 必ず SHA1 または MD5 に設定してください。これらのアルゴリズムは SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT のデフォルト値に含まれていません。

ALLOW_WEAK_CRYPTO* 設定の変更

SQLNET.ALLOW_WEAK_CRYPTO* オプションをデフォルト値から FALSE に設定する場合は、次の条件が満たされていることを確認してください。

- SQLNET.ENCRYPTION_TYPES_SERVER と SQLNET.ENCRYPTION_TYPES_CLIENT には、一致するセキュアな暗号化方式が 1 つあります。方式が DES、3DES、または RC4 (すべてのキーの長さ) ではない場合、セキュアと見なされます。
- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER と SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT には、一致するセキュアなチェックサム方式が 1 つあります。方式が MD5 ではない場合、セキュアと見なされます。
- クライアントで 2021 年 7 月 PSU パッチが適用されます。クライアントにパッチが適用されていない場合、クライアントでの接続が失われ、ORA-12269 エラーを受け取ります。

次の例は サンプル NNE 設定を示しています。SQLNET.ENCRYPTION_TYPES_SERVER と SQLNET.ENCRYPTION_TYPES_CLIENT をFALSE に設定して、非セキュア接続をブロックすると仮定します。チェックサムオプションの設定は、どちらにも SHA256 が含まれているため、前提条件を満たしています。ただし、SQLNET.ENCRYPTION_TYPES_CLIENT と SQLNET.ENCRYPTION_TYPES_SERVER では、非セキュアな暗号化方式である DES、3DES、および RC4 が使用されます。したがって、SQLNET.ALLOW_WEAK_CRYPTO* オプションを FALSE に設定する場合は、まず SQLNET.ENCRYPTION_TYPES_SERVER と SQLNET.ENCRYPTION_TYPES_CLIENT をセキュアな暗号化方式 (AES256 など) に設定します。

オプション設定	値
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	SHA256, SHA384, SHA512
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA1,MD5,SHA256
SQLNET.ENCRYPTION_TYPES_CLIENT	RC4_256, 3DES168, DES40
SQLNET.ENCRYPTION_TYPES_SERVER	RC4_256, 3DES168, DES40

NATIVE_NETWORK_ENCRYPTION オプションの削除

DB インスタンスから NNE を削除できます。

NATIVE_NETWORK_ENCRYPTION オプションを DB インスタンスから削除するには、次のいずれかを実行します。

- 複数の DB インスタンスからオプションを削除するには、それらが属しているオプショングループから NATIVE_NETWORK_ENCRYPTION オプションを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。NATIVE_NETWORK_ENCRYPTION オプションを削除した後に DB インスタンスを再起動する必要はありません。詳細については、「オプショングループからオプションを削除する」を参照してください。
- 1 つの DB インスタンスからオプションを削除するには、DB インスタンスを変更し、NATIVE_NETWORK_ENCRYPTION オプションを含まない別のオプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。NATIVE_NETWORK_ENCRYPTION オプションを削除した後に DB インスタンスを再起動する必要はありません。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle OLAP

Amazon RDS は、0LAP オプションを使用することで Oracle OLAP をサポートします。このオプションは、Oracle DB インスタンスのオンライン分析処理 (OLAP) を提供します。Oracle OLAP を使用すると、OLAP スタンダードに従って次元オブジェクトやキューブを作成することにより、大量のデータを分析できます。詳細については、Oracle のドキュメントを参照してください。

> **Important**
>
> Oracle OLAP を使用する際、共通脆弱性評価システム (CVSS) のスコアが 9 以上、またはそのほかのセキュリティ脆弱性の報告によりセキュリティ脆弱性がある場合に、Amazon RDS は自動的に DB インスタンスを最新の Oracle PSU にアップデートします。

Amazon RDS は、Oracle Database 19c および Oracle Database 21c の Enterprise Edition の Oracle OLAP をサポートしています。

> **Note**
>
> RDS for Oracle は、Oracle Database 26ai の OLAP をサポートしていません。これは、Oracle がそのリリースで OLAP を廃止したためです。ユースケースによっては、代わりに Oracle 分析ビューを使用できます。詳細については、Oracle ドキュメントの「Oracle 分析ビュー」を参照してください。

Oracle OLAP の前提条件

Oracle OLAP を使用するための前提条件は次のとおりです。

- Oracle から Oracle OLAP のライセンスを入手する必要があります。詳細については、Oracle のドキュメントの Licensing Information を参照してください。
- DB インスタンスは、十分な性能のインスタンスクラスのものである必要があります。Oracle OLAP は、db.t3.small DB インスタンスクラスではサポートされていません。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。
- DB インスタンスで [マイナーバージョン自動アップグレード] が有効になっている必要があります。このオプションにより、DB インスタンスは、(それが利用可能になった時点で) DB エンジンのマイナーバージョンアップグレードを自動的に受信できるようになります。これは、Oracle の Java 仮想マシン (JVM) をインストールする、すべてのオプションに必須です。Amazon RDS

では、このオプションを使用して、DB インスタンスに対し最新の Oracle パッチセット更新 (PSU)、またはリリース更新 (RU) を行います。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

- DB インスタンスに 0LAPSYS というユーザーが存在していないことが必要です。存在している場合、OLAP オプションのインストールは失敗します。

Oracle OLAP のベストプラクティス

Oracle OLAP を使用するためのベストプラクティスは次のとおりです。

- セキュリティを最大にするためには、0LAP オプションを Secure Sockets Layer (SSL) で使用します。詳細については、「Oracle Secure Sockets Layer」を参照してください。
- DB インスタンスへのアクセスを制限するように、DB インスタンスを設定します。詳細については、「VPC の DB インスタンスにアクセスするシナリオ」および「VPC 内の DB インスタンスの使用」を参照してください。

Oracle OLAP オプションの追加

DB インスタンスに 0LAP オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

DB インスタンスに Oracle Java Virtual Machine (JVM) がインストールされていない場合、0LAP オプションが追加されている間は短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに Oracle OLAP が使用可能となります。

OLAP オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

- [エンジン] で DB インスタンスの Oracle エディションを選択します。
- [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [OLAP] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

- 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle OLAP の使用

Oracle OLAP オプションを有効にしたら、使用をスタートできます。Oracle OLAP でサポートされている機能のリストについては、Oracle のドキュメントを参照してください。

Oracle OLAP オプションの削除

OLAP オプションが提供するデータ型を使用するすべてのオブジェクトを削除したら、そのオプションを DB インスタンスから削除できます。DB インスタンスに Oracle Java Virtual Machine (JVM) がインストールされていない場合、OLAP オプションの削除中にサービスが短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。OLAP オプションを削除した後に DB インスタンスを再起動する必要はありません。

OLAP オプションを削除するには

1. データをバックアップします。

> ⚠ Warning
>
> オプションの一部として有効化されたデータ型がインスタンスで使用されている場合、OLAP オプションを削除すると、データが失われる可能性があります。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

2. 既存のオブジェクトが、OLAP オプションのデータ型や機能を参照しているかどうかを確認します。
3. OLAP オプションのデータ型や機能を参照するすべてのオブジェクトを削除します。
4. 次のいずれかを行ってください。

   - OLAP オプションを所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
   - DB インスタンスを修正して、OLAP オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Secure Sockets Layer

RDS for Oracle DB インスタンスの SSL 暗号化を有効するには、その DB インスタンスに関連付けられているオプショングループに、Oracle SSL オプションを追加します。Oracle からの要求があるため、Amazon RDS では SSL 接続のために 2 番目のポートを使用しています。この方法では、クリアテキストと SSL 暗号化の両方の通信を、DB インスタンスと SQL*Plus 間で同時に実行できます。例えば、このポートで SSL 暗号化通信を使用して VPC 外部のリソースと通信する一方で、このポートでクリアテキスト通信を使用して VPC 内の他のリソースと通信できます。

> **Note**
>
> 同じ RDS for Oracle DB インスタンスで SSL またはネイティブネットワークの暗号化 (NNE) のいずれかを使用できますが、両方使用することはできません。SSL 暗号化を使用する場合は、他のすべての接続の暗号化を無効にする必要があります。詳細については、「Oracle ネイティブネットワーク暗号化」を参照してください。

SSL/TLS と NNE は、Oracle Advanced Security の一部ではなくなりました。RDS for Oracle では、以下のデータベースバージョンのすべてのライセンスされたエディションで、SSL 暗号化を使用できます。

- Oracle Database 26ai (26.0.0.0)
- Oracle Database 21c (21.0.0.0)
- Oracle Database 19c (19.0.0.0)

> **Note**
>
> Oracle Database 26ai の場合、Oracle SSL オプションは TLS 1.2 のみをサポートしており、デフォルトの暗号スイートは `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` です。Oracle Database 26ai では、次の暗号スイートがサポートされています。
>
> - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` (デフォルト)
> - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
> - `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
> - `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384`

トピック

- Oracle SSL オプションの TLS バージョン
- Oracle SSL オプションの暗号スイート
- FIPS のサポート
- 暗号スイートと証明書の互換性
- SSL オプションの追加
- RDS for Oracle DB インスタンスで SSL を使用するように SQL*Plus を設定する
- SSL を使用した RDS for Oracle DB インスタンスへの接続
- JDBC を介した SSL 接続のセットアップ
- SSL 接続で DN を一致させる
- SSL 接続のトラブルシューティング

Oracle SSL オプションの TLS バージョン

Amazon RDS for Oracle は、Transport Layer Security (TLS) バージョン 1.0 および 1.2 をサポートしています。新しい Oracle SSL オプションを追加するときは、SQLNET.SSL_VERSION を有効な値に明示的に設定します。次の表は、オプション設定に使用できる値を示しています。

値	サポートされている Oracle バージョン	説明
"1.0"	19c、21c	クライアントは、TLS バージョン 1.0 のみを使用して DB インスタンスに接続できます。既存の Oracle SSL オプションで、SQLNET.SSL_VERSION は自動的に "1.0" に設定されます。必要に応じて設定を変更できます。
"1.2"	すべて	クライアントは、TLS 1.2 のみを使用して DB インスタンスに接続できます。
"1.2 or 1.0"	19c、21c	クライアントは、TLS 1.2 または 1.0 のいずれかを使用して DB インスタンスに接続できます。

Oracle SSL オプションの暗号スイート

Amazon RDS for Oracle は、複数の SSL 暗号スイートをサポートしています。デフォルトでは、Oracle SSL オプションは Oracle Database 19c および Oracle Database 21c に対して `SSL_RSA_WITH_AES_256_CBC_SHA` 暗号スイートを使用するように設定されています。Oracle Database 26ai の場合、デフォルトの暗号スイートは `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` です。SSL 接続で使用する別の暗号スイートを指定するには、`SQLNET.CIPHER_SUITE` オプション設定を使用します。

`SQLNET.CIPHER_SUITE` には複数の値を指定することができます。この手法は、DB インスタンス間にデータベースリンクがあり、暗号スイートを更新する場合に便利です。

次の表は、Oracle Database 19c、21c および 26ai のすべての工ディションでの RDS for Oracle の SSL サポートをまとめたものです。

暗号スイート (SQLNET.CIPHER_SUITE)	TLS バージョン (SQLNET.SSL_VERSION)	FIPS のサポート	FedRAMP 準拠	サポートされている Oracle バージョン
SSL_RSA_WITH_AES_256_CBC_SHA (デフォルトは 19c および 21c)	1.0 および 1.2	はい	いいえ	19c、21c
SSL_RSA_WITH_AES_256_CBC_SHA256	1.2	はい	いいえ	19c、21c
SSL_RSA_WITH_AES_256_GCM_SHA384	1.2	はい	いいえ	19c、21c
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (デフォルトは 26ai)	1.2	可能	はい	19c、21c、26ai
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	1.2	可能	はい	19c、21c

暗号スイート (SQLNET.CIPHER_SUITE)	TLS バージョン (SQLNET.SSL_VERSION)	FIPS のサポート	FedRAMP 準拠	サポートされている Oracle バージョン
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	1.2	可能	はい	19c、21c、26ai
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	1.2	可能	はい	19c、21c
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	1.2	可能	はい	19c、21c
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	1.2	可能	はい	19c、21c
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	1.2	可能	はい	19c、21c、26ai
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	1.2	可能	はい	19c、21c、26ai

FIPS のサポート

RDS for Oracle では、連邦情報処理規格 (FIPS) 140-2 標準を使用できます。FIPS 140-2 は、暗号化モジュールのセキュリティ要件を規定する米国政府のスタンダード規格です。FIPS 標準は、Oracle SSL オプションの `FIPS.SSLFIPS_140` を `TRUE` に設定することでオンにします。SSL に FIPS 140-2 を設定した場合、暗号化ライブラリはクライアントと RDS for Oracle DB インスタンスの間のデータを暗号化します。

クライアントは、FIPS 準拠の暗号スイートを使用する必要があります。接続を確立する際、クライアントと RDS for Oracle DB インスタンスはメッセージの送受信に使用する暗号スイートをネゴシエートします。Oracle SSL オプションの暗号スイート の表に、各 TLS バージョンについて FIPS 準拠の SSL 暗号スイートを示します。詳細については、Oracle データベースドキュメントの「Oracle データベース FIPS 140-2 設定」を参照してください。

暗号スイートと証明書の互換性

RDS for Oracle は、RSA 証明書と楕円曲線デジタル署名アルゴリズム (ECDSA) 証明書の両方をサポートしています。DB インスタンスの SSL を設定するときは、`SQLNET.CIPHER_SUITE` オプション設定で指定した暗号スイートが DB インスタンスで使用される証明書タイプと互換性があることを確認する必要があります。

次の表は、証明書タイプと暗号スイート間の互換性を示しています。

[証明書タイプ]	互換性のある暗号スイート	互換性のない暗号スイート
RSA 証明書 (rds-ca-2019、rds-ca-rsa2048-g1、rds-ca-rsa4096-g1)	SSL_RSA_WITH_AES_256_CBC_SHA SSL_RSA_WITH_AES_256_CBC_SHA256 SSL_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDSA 証明書 (rds-ca-ecc384-g1)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_CBC_SHA

[証明書タイプ]	互換性のある暗号スイート	互換性のない暗号スイート
	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	SSL_RSA_WITH_AES_256_CBC_SHA256 SSL_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

SQLNET.CIPHER_SUITE オプション設定で複数の暗号スイートを指定する場合は、DB インスタンスで使用される証明書タイプと互換性のある暗号スイートを少なくとも 1 つ含めてください。証明書タイプが異なる複数の DB インスタンスでオプショングループを使用している場合は、証明書タイプごとに少なくとも 1 つの暗号スイートを含めます。

DB インスタンスの証明書タイプと互換性のない暗号スイートのみを含む SSL オプションにオプショングループを関連付けようとすると、オペレーションは失敗し、非互換性を示すエラーメッセージが表示されます。

SSL オプションの追加

SSL を使用するには、RDS for Oracle DB インスタンスが、SSL オプションを含むオプショングループに関連付けられている必要があります。

コンソール

SSL オプションをオプショングループに追加するには

1. 新しいオプショングループを作成するか、SSL オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [SSL] オプションを追加します。

   SSL 接続に FIPS で検証された暗号スイートのみを使用する場合は、FIPS.SSLFIPS_140 オプションを TRUE に設定します。FIPS スタンダードの詳細については、「FIPS のサポート」を参照してください。

   オプショングループへのオプションの追加の詳細については、「オプショングループにオプションを追加する」を参照してください。

3. 新しい RDS for Oracle DB インスタンスを作成し、オプショングループをそのインスタンスに関連付けるか、オプショングループを関連付けるように RDS for Oracle DB インスタンスを変更します。

   DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

   DB インスタンスの変更については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

SSL オプションをオプショングループに追加するには

1. 新しいオプショングループを作成するか、SSL オプションを追加する既存のオプショングループを識別します。

   オプショングループの作成の詳細については、「オプショングループを作成する」を参照してください。

2. オプショングループに [SSL] オプションを追加します。

   以下のオプション設定を指定します。

- `Port` - SSL ポート番号
- `VpcSecurityGroupMemberships` - オプションが有効な VPC セキュリティグループ
- `SQLNET.SSL_VERSION` - クライアントから DB インスタンスへの接続に使用できる TLS バージョン

例えば、以下の AWS CLI コマンドでは、`SSL` オプションを、`ora-option-group` という名前のオプショングループに追加します。

Example

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group --option-group-name ora-option-group \
  --options
 'OptionName=SSL,Port=2484,VpcSecurityGroupMemberships="sg-68184619",OptionSettings=[{Name=
```

Windows の場合:

```
aws rds add-option-to-option-group --option-group-name ora-option-group ^
  --options
 'OptionName=SSL,Port=2484,VpcSecurityGroupMemberships="sg-68184619",OptionSettings=[{Name=
```

3. 新しい RDS for Oracle DB インスタンスを作成し、オプショングループをそのインスタンスに関連付けるか、オプショングループを関連付けるように RDS for Oracle DB インスタンスを変更します。

   DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

   DB インスタンスの変更については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS for Oracle DB インスタンスで SSL を使用するように SQL*Plus を設定する

Oracle SSL オプションを使用する RDS for Oracle DB インスタンスに接続する前に、SQL*Plus を設定する必要があります。

> **Note**
>
> 適切なクライアントからの DB インスタンスへのアクセスを許可するには、セキュリティグループが適切に設定されていることを確認します。詳細については、「セキュリティグループによるアクセス制御」を参照してください。さらに、この手順は、SQL*Plus および Oracle ホームを直接使用するその他のクライアント向けです。JDBC 接続については、「JDBC を介した SSL 接続のセットアップ」を参照してください。

SSL を使用して RDS for Oracle DB インスタンスに接続するように SQL*Plus を設定するには

1. ORACLE_HOME 環境変数を Oracle ホームディレクトリの場所に設定します。

   Oracle ホームディレクトリへのパスは、インストールによって異なります。次の例では、ORACLE_HOME 環境変数を設定します。

   ```
   prompt>export ORACLE_HOME=/home/user/app/user/product/19.0.0/dbhome_1
   ```

   Oracle 環境変数の設定については、Oracle ドキュメントの「SQL*Plus 環境変数」を参照してください。オペレーティングシステムについては、Oracle インストールガイドを参照してください。

2. $ORACLE_HOME/lib を LD_LIBRARY_PATH 環境変数に追加します。

   LD_LIBRARY_PATH 環境変数を設定する例を以下に示します。

   ```
   prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
   ```

3. $ORACLE_HOME/ssl_wallet に Oracle Wallet のディレクトリを作成します。

   以下は、Oracle Wallet のディレクトリを作成する例です。

   ```
   prompt>mkdir $ORACLE_HOME/ssl_wallet
   ```

4. すべての AWS リージョン で動作する証明書バンドルをダウンロードし、そのファイルを ssl_wallet ディレクトリに置きます。詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

5. $ORACLE_HOME/network/admin ディレクトリで、tnsnames.ora ファイルを変更または作成して以下のエントリを含めます。

```
net_service_name =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCPS)
        (HOST = endpoint)
        (PORT = ssl_port_number)
      )
    )
    (CONNECT_DATA =
      (SID = database_name)
    )
    (SECURITY =
      (SSL_SERVER_CERT_DN =
 "C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=endpoint")
    )
  )
```

6. 同じディレクトリで、sqlnet.ora ファイルを変更または作成して以下のパラメータを含めます。

> **Note**
>
> TLS でセキュリティ保護された接続を介してエンティティと通信するために、Oracle は認証用の必要な証明書を持つウォレットを必要とします。ステップ 7 に示すように、Oracle の ORAPKI ユーティリティを使用して Oracle ウォレットを作成して管理できます。詳細については、Oracle ドキュメントで「ORAPKI を使用した Oracle ウォレットの設定」を参照してください。

```
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
 $ORACLE_HOME/ssl_wallet)))
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 1.2
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA256)
SSL_SERVER_DN_MATCH = ON
```

> **Note**
>
> SSL_VERSION は別の値に設定できます (DB インスタンスでサポートされている場合)。

7. 以下のコマンドを実行して Oracle Wallet を作成します。

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only
```

8. OS ユーティリティを使用して、.pem バンドルファイル内の各証明書を個別の .pem ファイルに抽出します。
9. 個別の orapki コマンドを使用して各証明書をウォレットに追加し、*certificate-pem-file* を.pem ファイルの絶対ファイル名に置き換えます。

```
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
      certificate-pem-file -auto_login_only
```

詳細については、「SSL/TLS 証明書のローテーション」を参照してください。

SSL を使用した RDS for Oracle DB インスタンスへの接続

前述のように SQL*Plus で SSL を使用するように設定すると、SSL オプションを使用して RDS for Oracle DB インスタンスに接続できるようになります。オプションとして、初期に、tnsnames.ora ファイルと sqlnet.ora ファイルが含まれているディレクトリを指す TNS_ADMIN 値をエクスポートできます。これにより、これらのファイルを SQL*Plus は確実に見つけることができます。次の例では、TNS_ADMIN の値をエクスポートしています。

```
export TNS_ADMIN = ${ORACLE_HOME}/network/admin
```

DB インスタンスに接続します。例えば、SQL *Plus と、tnsnames.ora ファイルの *<net_service_name>* を使用して接続できます。

```
sqlplus mydbuser@net_service_name
```

以下のコマンドを使用すると、tnsnames.ora ファイルを使用せずに SQL *Plus のみを使用して、DB インスタンスに接続することもできます。

```
sqlplus 'mydbuser@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST = endpoint) (PORT
 = ssl_port_number))(CONNECT_DATA = (SID = database_name)))'
```

SSL を使用せずに RDS for Oracle DB インスタンスに接続することもできます。例えば、以下のコマンドは SSL 暗号化を使用せずにクリアテキストポートを使用して DB インスタンスに接続します。

```
sqlplus 'mydbuser@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = endpoint) (PORT
 = port_number))(CONNECT_DATA = (SID = database_name)))'
```

Transmission Control Protocol (TCP) ポートアクセスを閉じる場合は、IP アドレスの進入を許可しないセキュリティグループを作成して、インスタンスに追加します。これを追加することで、TCP ポートを使用した接続を閉じながら、SSL オプションのセキュリティグループで許可された範囲内の IP アドレスから指定された SSL ポートを使用して接続できます。

JDBC を介した SSL 接続のセットアップ

JDBC を介した SSL 接続を使用するには、キーストアを作成して、Amazon RDS のルート CA 証明書を信頼し、以下で指定されるコードスニペットを使用する必要があります。

キーストアの作成の詳細については、Oracle のドキュメントの「キーストアの作成」を参照してください。リファレンス情報については、「Java プラットフォーム、標準エディションツールリファレンス」の「keytool」を参照してください。

```
keytool -genkey -alias client -validity 365 -keyalg RSA -keystore clientkeystore
```

Amazon RDS のルート CA 証明書を信頼するよう、以下のステップを実行します。

Amazon RDS ルート CA 証明書を信頼するには

1. すべての AWS リージョン で動作する証明書バンドルをダウンロードし、そのファイルを ssl_wallet ディレクトリに置きます。

   証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。
2. OS ユーティリティを使用して、.pem ファイル内の各証明書を個別のファイルに抽出します。
3. *certificate-pem-file* を証明書 .pem ファイルの名前 (.pem 拡張子なし) に置き換えて、個別の openssl コマンドを使用して各証明書を .der 形式に変換します。

   ```
   openssl x509 -outform der -in certificate-pem-file.pem -out certificate-pem-
   file.der
   ```
4. 次のコマンドを使用して、各証明書をキーストアにインポートします。

```
keytool -import -alias rds-root -keystore clientkeystore.jks -file certificate-pem-file.der
```

詳細については、「SSL/TLS 証明書のローテーション」を参照してください。

5. キーストアが正常に作成されたことを確認します。

```
keytool -list -v -keystore clientkeystore.jks
```

キーストアのパスワードを求められたら、これを入力します。

次のコード例は、JDBC を使用する SSL 接続のセットアップ方法を示します。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "dns-name-provided-by-amazon-rds";
    private static final String SSL_PORT = "ssl-option-port-configured-in-option-group";
    private static final String DB_SID = "oracle-sid";
    private static final String DB_USER = "user-name";
    private static final String DB_PASSWORD = "password";
    // This key store has only the prod root ca.
    private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
    private static final String KEY_STORE_PASS = "keystore-password";

    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
                "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%s))(CONNECT_DATA=(SID=%s)))",
                DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
```

```
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
 properties);
        // If no exception, that means handshake has passed, and an SSL connection can
 be opened
    }
}
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

SSL 接続で DN を一致させる

データベースサーバーの識別子名 (DN) をそのサービス名と一致させるには、Oracle パラメータ SSL_SERVER_DN_MATCH を使用できます。マッチングの検証を実施する場合は、SSL により、証明書がサーバーから取得されたものであることが確認されます。マッチングの検証を実施しない場合、SSL によって確認が行われますが、一致しているかどうかにかかわらず接続が許可されます。マッチングを実施しない場合、サーバーで識別をモデリング偽造することを許可できます。

DN マッチングを実施するには、DN マッチングのプロパティを追加し、以下に示されている接続文字列を使用します。

DN マッチングを実施するには、このプロパティをクライアント接続に追加します。

```
properties.put("oracle.net.ssl_server_dn_match", "TRUE");
```

SSL の使用時は、以下の接続文字列を使用して DN マッチングを実施します。

```
final String connectionString = String.format(
    "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%s))" +
    "(CONNECT_DATA=(SID=%s))" +
    "(SECURITY = (SSL_SERVER_CERT_DN =
\"C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=%s\")))",
    DB_SERVER_NAME, SSL_PORT, DB_SID, DB_SERVER_NAME);
```

SSL 接続のトラブルシューティング

データベースにクエリを実行すると、ORA-28860 エラーが表示されることがあります。

```
ORA-28860: Fatal SSL error
28860. 00000 - "Fatal SSL error"
*Cause: An error occurred during the SSL connection to the peer. It is likely that this
 side sent data which the peer rejected.
*Action: Enable tracing to determine the exact cause of this error.
```

このエラーは、サーバーがサポートしていないバージョンの TLS を使用してクライアントが接続しようとしたときに発生します。このエラーを回避するには、sqlnet.ora を編集して、SSL_VERSION を正しい TLS バージョンに設定します。詳細については、My Oracle Support の「Oracle Support ドキュメント 2748438.1」を参照してください。

Oracle Spatial

Amazon RDS は、SPATIAL オプションを使用することで Oracle Spatial をサポートします。Oracle Spatial は、Oracle データベースのストレージ、取得、更新、および spatial データコレクションのクエリを可能にする SQL スキーマと機能を提供します。詳細については、Oracle ドキュメントの Spatial Concepts を参照してください。Amazon RDS は、サポートされているすべてのリリースのすべてのエディションで Oracle Spatial をサポートしています。

Supplemental パッチバンドル (SPB) の仕組み

RDS for Oracle は、四半期ごとに、サポートされているすべてのメジャーエンジンに対して新しいマイナーエンジンバージョンをリリースします。リリースアップデート (RU) エンジンバージョンには、指定された四半期の RU パッチを含めることで Oracle からのバグ修正が組み込まれています。Supplemental パッチバンドル (SPB) エンジンバージョンには、すべての RU パッチと追加のサプリメンタルパッチが含まれています。DB インスタンスが Oracle Spatial を使用していて、Spatial オプションに関連する追加のパッチが必要な場合は、SPB リリースを使用します。例えば、19.0.0.0.ru-2026-04.spb-1.r1 は、エンジンバージョン 19.0.0.0.ru-2026-04.rur-2026-04.r1 の RU パッチと Supplemental パッチを含むマイナーエンジンバージョンです。SPB は Oracle Database 19c でのみサポートされています。

> **Note**
>
> Spatial パッチバンドルの名前が Supplemental パッチバンドル (SPB) に変更されました。SPB には、Data Pump や GoldenGate など、Oracle Spatial 以外のバンドルパッチが追加されました。省略形「SPB」は変更されません。既存の SPB エンジンバージョンはすべて、以前と同じように動作し続けます。

SPB は名前が異なりますが、RU と同じように機能します。RU は、19.0.0.0.ru-2025-01.rur-2025-01.r1 という命名形式を使用します。SPB 名には、19.0.0.0.ru-2026-04.spb-1.r1 のように「spb」というテキストが含まれます。通常、SPB は対応する四半期 RU の 2～3 週間後にリリースされます。例えば、19.0.0.0.ru-2026-04.spb-1.r1 は 19.0.0.0.ru-2026-04.rur-2026-04.r1 以降にリリースされています。

RDS for Oracle には、RU と SPB の自動マイナーバージョンアップグレード用の個別のパスがあります。DB インスタンスが RU を使用している場合、RDS は自動的にインスタンスを RU にアップグレードします。DB インスタンスが SPB を使用している場合、RDS はインスタンスを SPB にアップグレードします。

RU および SPB の詳細については、「Oracle のマイナーバージョンのアップグレード」を参照してください。Oracle Database 19c でサポートされている RU と SPB のリストについては、「Amazon RDS for Oracle リリースノート」の「Amazon RDS for Oracle Database 19c (19.0.0.0)」を参照してください。

Oracle Spatial の前提条件

Oracle Spatial を使用するための前提条件は次のとおりです。

- DB インスタンスが十分なインスタンスクラスであることを確認します。Oracle Spatial は、db.t3.small DB インスタンスクラスではサポートされていません。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。
- DB インスタンスで [自動マイナーバージョンアップグレード] が有効になっていることを確認します。このオプションにより、DB インスタンスは、(それが利用可能になった時点で) DB エンジンのマイナーバージョンアップグレードを自動的に受信できるようになります。これは、Oracle の Java 仮想マシン (JVM) をインストールする、すべてのオプションに必須です。Amazon RDS では、このオプションを使用して、DB インスタンスに対し最新の Oracle パッチセット更新 (PSU)、またはリリース更新 (RU) を行います。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Spatial のベストプラクティス

Oracle Spatial を使用するためのベストプラクティスは次のとおりです。

- セキュリティを最大にするためには、SPATIAL オプションを Secure Sockets Layer (SSL) で使用します。詳細については、「Oracle Secure Sockets Layer」を参照してください。
- DB インスタンスへのアクセスを制限するように、DB インスタンスを設定します。詳細については、「VPC の DB インスタンスにアクセスするシナリオ」および「VPC 内の DB インスタンスの使用」を参照してください。

Oracle Spatial オプションの追加

DB インスタンスに SPATIAL オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。

3. オプショングループを DB インスタンスに関連付けます。

DB インスタンスに Oracle Java Virtual Machine (JVM) がインストールされていない場合、SPATIAL オプションが追加されている間は短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。オプションを追加した後に DB インスタンスを再起動する必要はありません。オプショングループがアクティブになると、すぐに Oracle Spatial が使用可能となります。

> **Note**
>
> この停止中、パスワード検証機能は一時的に無効になります。また、停止中にパスワード検証機能に関連するイベントを確認することもできます。Oracle DB インスタンスが使用可能になる前に、パスワード検証機能が再び有効になります。

SPATIAL オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [エンジン] で DB インスタンスの Oracle エディションを選択します。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [SPATIAL] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Spatial オプションの削除

SPATIAL オプションで提供されるデータ型を使用するすべてのオブジェクトを削除したら、DB インスタンスからそのオプションを削除できます。DB インスタンスに Oracle Java Virtual Machine (JVM) がインストールされていない場合、SPATIAL オプションの削除中にサービスが短時間停止します。Oracle Java Virtual Machine (JVM) がすでに DB インスタンスにインストールされている場合、停止は発生しません。SPATIAL オプションを削除した後に DB インスタンスを再起動する必要はありません。

SPATIAL オプションを削除するには

1. データをバックアップします。

   > **Warning**
   >
   > オプションの一部として有効化されたデータ型がインスタンスで使用されている場合、SPATIAL オプションを削除すると、データが失われる可能性があります。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

2. 既存のオブジェクトが、SPATIAL オプションのデータ型や機能を参照しているかどうかを確認します。

   SPATIAL オプションが存在する場合、SPATIAL オプションを持たない新しいオプショングループを適用すると、インスタンスが停止することがあります。次のクエリを使用して、オブジェクトを識別できます。

```
SELECT OWNER, SEGMENT_NAME, TABLESPACE_NAME, BYTES/1024/1024 mbytes
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE '%TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
       (SELECT DISTINCT OWNER, TABLE_NAME
        FROM   DBA_TAB_COLUMNS
        WHERE  DATA_TYPE='SDO_GEOMETRY'
        AND    OWNER <> 'MDSYS')
ORDER BY 1,2,3,4;

SELECT OWNER, TABLE_NAME, COLUMN_NAME
FROM   DBA_TAB_COLUMNS
WHERE  DATA_TYPE = 'SDO_GEOMETRY'
AND    OWNER <> 'MDSYS'
```

```
ORDER BY 1,2,3;
```

3. SPATIAL オプションのデータ型や機能を参照するすべてのオブジェクトを削除します。
4. 次のいずれかを行ってください。

   - SPATIAL オプションを所属するオプショングループから削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
   - DB インスタンスを修正して、SPATIAL オプションが含まれない別オプショングループを指定します。この変更は、単一の DB インスタンスに影響します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle SQLT

Amazon RDS は、SQLT オプションの使用を通じて SQLTPLAIN (SQLT) をサポートします。SQLT は Oracle Database 19c 以降の任意のエディションで使用できます。

Oracle `EXPLAIN PLAN` ステートメントでは、SQL ステートメントの実行計画を決定できます。Oracle オプティマイザが、ネステッドループされたループ結合などの特定の実行計画を選択しているかどうかを検証できます。また、ハッシュ結合を介してネステッドループ結合を選択した理由など、オプティマイザの決定を理解するのに役立ちます。そのため、`EXPLAIN PLAN` はステートメントのパフォーマンスの理解に役立ちます。

SQLT は、レポートを作成する Oracle ユーティリティです。このレポートには、オブジェクト統計、オブジェクトメタデータ、オプティマイザ関連の初期化パラメータ、およびデータベース管理者が最適なパフォーマンスを得るために SQL ステートメントを調節するために使用できるその他の情報が含まれています。SQLT は、レポート内のすべてのセクションへのハイパーリンクを含む HTML レポートを生成します。

SQLT は、自動ワークロードリポジトリまたは Statspack レポートとは異なり、個々の SQL ステートメントに対して機能します。SQLT は、パフォーマンスデータを収集、保存、表示する SQL、PL/SQL、および SQL*Plus ファイルのコレクションです。

各 SQLT バージョンでは次の Oracle バージョンがサポートされています。

SQLT バージョン	Oracle Database 26ai	Oracle Database 21c	Oracle Database 19c
2018-07-25.v1	サポート対象	サポート対象	サポート
2018-03-31.v1	サポートされません	サポートされません	サポートされません
2016-04-29.v1	サポートされません	サポートされません	サポートされません

SQLT および使用のためのアクセス手順をダウンロードします。

- My Oracle Support アカウントにログインして、以下のドキュメントを開きます。
- SQLT のダウンロード: ドキュメント 215187.1
- SQLT のご利用方法: ドキュメント 1614107.1
- SQLT についてのよくある質問: ドキュメント 1454160.1

- 出力された SQLT 読み取りについて: ドキュメント 1456176.1
- メインレポートの解釈について: ドキュメント 1922234.1

Amazon RDS は、以下の SQLT メソッドをサポートしていません。

- XPLORE
- XHUME

&SQLT の前提条件

SQLT を使用するための前提条件は次のとおりです。

- 存在する場合は、SQLT で必要とされるロールとユーザーを削除する必要があります。

  SQLT オプションは、DB インスタンスの次のユーザーおよびロールを作成します。

  - SQLTXPLAIN ユーザー
  - SQLTXADMIN ユーザー
  - SQLT_USER_ROLE ロール

  DB インスタンスにこれらのユーザーまたはロールがある場合は、SQL クライアントを使用して DB インスタンスにログインし、次のステートメントを使用して削除します。

```
DROP USER SQLTXPLAIN CASCADE;
DROP USER SQLTXADMIN CASCADE;
DROP ROLE SQLT_USER_ROLE;
```

- 存在する場合は、SQLT で必要とされるテーブルスペースを削除する必要があります。

  SQLT オプションは、DB インスタンスの次のテーブルスペースを作成します。

  - RDS_SQLT_TS
  - RDS_TEMP_SQLT_TS

  DB インスタンスにこれらのテーブルスペースがある場合は、SQL クライアントを使用して DB インスタンスにログインし、削除します。

SQLT オプション設定

SQLT は、Oracle Tuning Pack および Oracle Diagnostics Pack によって提供されるライセンス機能を使用して動作します。Oracle Tuning Pack には SQL チューニングアドバイザが、Oracle Diagnostics Pack には自動ワークロードリポジトリが含まれています。SQLT 設定は、SQLT からこれらの機能へのアクセスを有効または無効にします。

Amazon RDS は、SQLT オプションの次の設定をサポートします。

オプション設定	有効な値	デフォルト値	説明
LICENSE_PACK	T, D, N	N	SQLT を使用してアクセスする Oracle Management Pack。以下のいずれかの値にエラーがあります。 • T は、Oracle Tuning Pack および Oracle Diagnostics Pack のライセンスがあり、SQL チューニングアドバイザおよび自動ワークロードリポジトリに SQLT からアクセスすることを示します。 • D は、Oracle Diagnostics Pack のライセンスがあり、自動ワークロードリポジトリに SQLT からアクセスすることを示します。 • N は、Oracle Tuning Pack および Oracle Diagnostics Pack のライセンスを所有していないこと、または一方または両方のライセンスを所有していますが、SQLT にアクセスを希望しないことを示しています。 **Note** Amazon RDS はこれらの Oracle Management Pack

オプション設定	有効な値	デフォルト値	説明
			のライセンスを提供しません。DB インスタンスに含まれていないパックを使用することを指定した場合は、DB インスタンスで SQLT を使用できます。ただし、SQLT はパックにアクセスできず、SQLT レポートにはパックのデータは含まれません。例えば、T を指定したが、DB インスタンスに Oracle Tuning Pack が含まれていない場合、SQLT は DB インスタンス上で動作しますが、作成するレポートに Oracle Tuning Pack に関係するデータは含まれません。
VERSION	2016-04-29.v1 2018-03-31.v1 2018-07-25.v1	2016-04-29.v1	インストールする SQLT のバージョン。 **Note** Oracle Database 19c と 21c では、サポートされているバージョンは 2018-07-25.v1 のみです。このバージョンは、これらのリリースのデフォルトです。

SQLT オプションの追加

DB インスタンスに SQLT オプションを追加する一般的なプロセスを次に示します。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに SQLT オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

SQLT オプションの追加後、オプショングループがアクティブになるとすぐに、SQLT がアクティブになります。

SQLT オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] で、使用する Oracle のエディションを選択します。SQLT オプションは、すべてのエディションでサポートされます。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [SQLT] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
4. (オプション) SQLT オプションを伴う各 DB インスタンスの SQLT のインストールを検証します。

   a. マスターユーザーとして DB インスタンスに接続している SQL クライアントを使用します。

Oracle DB インスタンスで SQL クライアントに接続する詳細については、「Oracle DB インスタンスへの接続」を参照してください。

b. 次のクエリを実行します。

```
SELECT sqltxplain.sqlt$a.get_param('tool_version') sqlt_version FROM DUAL;
```

クエリは Amazon RDS の SQLT オプションの最新バージョンを返します。12.1.160429 は Amazon RDS で使用可能である SQLT のバージョンの例です。

5. SQLT オプションで作成されたユーザーのパスワードを変更します。

a. マスターユーザーとして DB インスタンスに接続している SQL クライアントを使用します。

b. SQLTXADMIN ユーザーのパスワードを変更するには、以下の SQL ステートメントを実行します。

```
ALTER USER SQLTXADMIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

c. SQLTXPLAIN ユーザーのパスワードを変更するには、以下の SQL ステートメントを実行します。

```
ALTER USER SQLTXPLAIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

> **Note**
>
> SQLT をアップグレードするには、SQLT の旧バージョンをアンインストールしてから、新しいバージョンをインストールする必要があります。そのため、すべての SQLT メタデータが SQLT をアップグレードすると失われる可能性があります。データベースのメジャーバージョンのアップグレードでも、SQLT がアンインストールされ、再インストールされます。メジャーバージョンのアップグレードの例は、Oracle Database 18c から Oracle Database 19c へのアップグレードです。

SQLT の使用

SQLT は Oracle SQL*Plus ユーティリティで動作します。

使用する SQLT

1. My Oracle Support サイトの ドキュメント 215187.1 から SQLT .zip ファイルをダウンロードします。

   > **Note**
   >
   > My Oracle Support サイトから SQLT 12.1.160429 をダウンロードすることはできません。Oracle はこの古いバージョンを廃止しました。

2. SQLT .zip ファイルを解凍します。
3. コマンドプロンプトから、ファイルシステムの `sqlt/run` をディレクトリに変更します。
4. コマンドプロンプトで、SQL*Plus を開き、マスターユーザーとして DB インスタンスに接続します。

   SQL*Plus を使用する DB インスタンスへの接続の詳細については、「Oracle DB インスタンスへの接続」を参照してください。

5. SQL ステートメントの SQL ID を取得します。

```
SELECT SQL_ID FROM V$SQL WHERE SQL_TEXT='sql_statement';
```

以下のような出力が生成されます。

```
SQL_ID
-------------
chvsmttqjzjkn
```

6. SQLT を含む SQL ステートメントの分析:

```
START sqltxtract.sql sql_id sqltxplain_user_password
```

例えば、SQL IDchvsmttqjzjkn に対して、以下を入力してください。

```
START sqltxtract.sql chvsmttqjzjkn sqltxplain_user_password
```

SQLT は、HTML レポートと関連リソースを SQLT コマンドが実行されたディレクトリに .zip ファイルとして生成します。

7. (オプション) アプリケーションユーザーが SQLT を使用して SQL ステートメントを診断できるようにするには、次のステートメントを使用して SQLT_USER_ROLE を各アプリケーションユーザーに付与します。

```
GRANT SQLT_USER_ROLE TO application_user_name;
```

> **Note**
>
> Oracle は、SYS ユーザー、または DBA ロールを持つユーザーで SQLT を実行することを推奨していません。SQLT_USER_ROLE をアプリケーションユーザーに付与することにより、アプリケーションユーザーのアカウントを使用して SQLT 診断を実行することをお勧めします。

SQLT オプションのアップグレード

Amazon RDS for Oracle では、SQLT オプションをバージョンを既存のバージョンから上位のバージョンにアップグレードできます。SQLT オプションをアップグレードするには、SQLT の新しいバージョンの SQLT の使用 のステップ 1-3 を完了します。また、そのセクションのステップ 7 で以前のバージョンの SQLT の特権を付与した場合は、新しい SQLT バージョンの特権を再度付与してください。

SQLT オプションをアップグレードすると、古い SQLT バージョンのメタデータが失われます。古い SQLT バージョンのスキーマおよび関連オブジェクトは削除され、新しいバージョンの SQLT がインストールされます。最新の SQLT バージョン における変更の詳細については、My Oracle Support サイトの「ドキュメント 1614201.1」を参照してください。

> **Note**
>
> バージョンのダウングレードはサポートされていません。

SQLT 設定の変更

SQLT を有効にした後、オプションで LICENSE_PACK および VERSION 設定を変更できます。

オプション設定の変更方法の詳細については、「オプションの設定を変更する」を参照してください。各設定の詳細については、「SQLT オプション設定」を参照してください。

SQLT オプションの削除

DB インスタンスから SQLT を削除できます。

DB インスタンスから SQLT を削除するには、次のいずれかを実行します。

- 複数の DB インスタンスから SQLT を削除するには、DB インスタンスが属しているオプショングループから SQLT オプションを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。

- 単一の DB インスタンスから SQLT を削除するには、DB インスタンスを変更し、NNE オプションが含まれていない別のオプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle Statspack

Oracle Statspack オプションは、Oracle Statspack パフォーマンス統計機能をインストールして有効にします。Oracle Statspack は、パフォーマンスデータを収集、保存、表示する SQL、PL/SQL、および SQL*Plus スクリプトのコレクションです。Statspack スクリプトとそのドキュメントは、`$ORACLE_HOME/rdbms/admin` のデータベースとともにインストールされています (例: `spdoc.txt`)。

> **Note**
>
> Oracle Statspack は、Oracle によるサポートが停止されたため、より高度な自動ワークロードリポジトリ (AWR) に置き換えられました。AWR は、Diagnostics Pack を購入した Oracle Enterprise Edition のお客様だけが使用できます。Oracle Statspack は、Amazon RDS 上の任意の Oracle DB エンジンで使用できます。Amazon RDS リードレプリカでは Oracle Statspack を実行できません。

Oracle Statspack のセットアップ

Statspack スクリプトを実行するには、Statspack オプションを追加する必要があります。

Oracle Statspack をセットアップするには

1. SQL クライアントで、管理アカウントを使用して Oracle DB に口グインします。
2. Statspack がインストールされているかどうかに応じて、次のいずれかの操作を行います。
   - Statspack がインストールされていて、`PERFSTAT` アカウントが Statspack に関連付けられている場合は、ステップ 4 に進みます。
   - Statspack がインストールされておらず、`PERFSTAT` アカウントが存在する場合は、次のようにアカウントを削除します。

     ```
     DROP USER PERFSTAT CASCADE;
     ```

     それ以外の場合は、Statspack オプションを追加しようとすると、エラーと `RDS-Event-0058` が生成されます。
3. オプショングループに Statspack オプションを追加します。「オプショングループにオプションを追加する」を参照してください。

Amazon RDS は、DB インスタンスに Statspack スクリプトを自動的にインストールしてから、PERFSTAT アカウントを設定します。

4. 次の SQL 文を使用してパスワードをリセットし、pwd を新しいパスワードに置き換えます。

```
ALTER USER PERFSTAT IDENTIFIED BY pwd ACCOUNT UNLOCK;
```

PERFSTAT ユーザーアカウントを使用してログインし、Statspack スクリプトを実行できます。

5. 次のステートメントを使用して、PERFSTAT アカウントに CREATE JOB 権限を付与します。

```
GRANT CREATE JOB TO PERFSTAT;
```

6. PERFSTAT.STATS$IDLE_EVENT テーブル内のアイドル待機イベントが設定されていることを確認します。

Oracle バグ 28523746 のため、PERFSTAT.STATS$IDLE_EVENT のアイドル待機イベントが設定されていないことがあります。すべてのアイドルイベントが利用可能であることを確認するには、次のステートメントを実行します。

```
INSERT INTO PERFSTAT.STATS$IDLE_EVENT (EVENT)
SELECT NAME FROM V$EVENT_NAME WHERE WAIT_CLASS='Idle'
MINUS
SELECT EVENT FROM PERFSTAT.STATS$IDLE_EVENT;
COMMIT;
```

Statspack レポートの生成

Statspack レポートでは、2 つのスナップショットを比較します。

Statspack レポートを生成するには

1. SQL クライアントで、PERFSTAT アカウントを使用して Oracle DB にログインします。
2. 次のいずれかの方法を使用して、スナップショットを作成します。

- Statspack スナップショットを手動で作成します。
- 特定の時間間隔の後に Statspack スナップショットを取得するジョブを作成します。例えば、次のジョブは Statspack スナップショットを 1 時間おきに作成します。

```
VARIABLE jn NUMBER;
exec dbms_job.submit(:jn, 'statspack.snap;',SYSDATE,'TRUNC(SYSDATE
+1/24,''HH24'')');
COMMIT;
```

3. 次のクエリを使用してスナップショットを表示します。

```
SELECT SNAP_ID, SNAP_TIME FROM STATS$SNAPSHOT ORDER BY 1;
```

4. Amazon RDS プロシージャ `rdsadmin.rds_run_spreport` を実行し、begin_snap と end_snap をスナップショット ID に置き換えます。

```
exec rdsadmin.rds_run_spreport(begin_snap,end_snap);
```

例えば、次のコマンドでは、Statspack スナップショット 1 と 2 の間の間隔に基づいてレポートが作成されます。

```
exec rdsadmin.rds_run_spreport(1,2);
```

Statspack レポートのファイル名には、2 つのスナップショットの番号が含まれます。例えば、Statspack スナップショット 1 および 2 を使用して作成されたレポートファイルの名前は `ORCL_spreport_1_2.lst` になります。

5. 出力にエラーがないかモニタリングします。

Oracle Statspack は、レポートを実行する前にチェックを実行します。したがって、コマンド出力にエラーメッセージが表示される可能性もあります。例えば、スタートの Statspack スナップショット値が終了値よりも大きい、無効な範囲に基づいてレポートを生成しようとするとします。この場合、出力にはエラーメッセージが表示されますが、DB エンジンはエラーファイルを生成しません。

```
exec rdsadmin.rds_run_spreport(2,1);
*
ERROR at line 1:
ORA-20000: Invalid snapshot IDs. Find valid ones in perfstat.stats$snapshot.
```

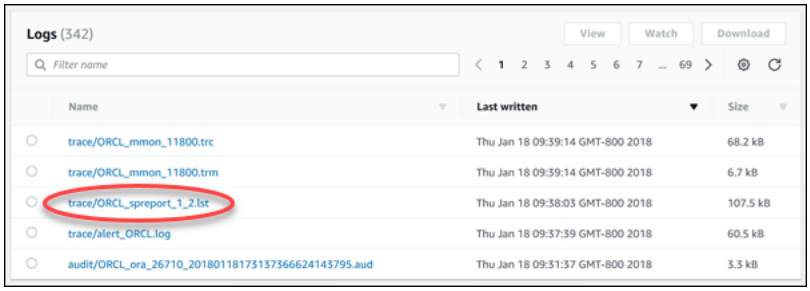
無効な番号の Statspack スナップショットを使用すると、出力にエラーが表示されます。例えば、スナップショット 1 と 50 のレポートを生成しようとして、スナップショット 50 が存在しない場合、出力にはエラーが表示されます。

```
exec rdsadmin.rds_run_spreport(1,50);
*
ERROR at line 1:
ORA-20000: Could not find both snapshot IDs
```

6. (オプション)

   レポートを取得するには、Oracle トレースファイルを使用する で説明されているように、トレースファイルプロシージャを呼び出します。

   または、RDS コンソールから Statspack レポートをダウンロードします。DB インスタンスの詳細の [ログ] セクションに移動し、[ダウンロード] を選択します。次の例は、`trace/ORCL_spreport_1_2.lst` を示します。



   レポートの生成中にエラーが発生した場合、DB エンジンはレポートと同じ命名規則を使用しますが、拡張子は `.err` です。例えば、Statspack スナップショット 1 および 7 を使用してレポートを作成するときにエラーが発生した場合、レポートファイルの名前は `ORCL_spreport_1_7.err` になります。エラーレポートは、スタンダードのスナップショットレポートと同じ方法でダウンロードできます。

Statspack スナップショットの削除

一定の範囲の Oracle Statspack スナップショットを削除するには、次のコマンドを使用します。

```
exec statspack.purge(begin snap, end snap);
```

Oracle のタイムゾーン

タイムゾーンオプションを使用して、Oracle DB インスタンスで使用するシステムのタイムゾーンを変更することができます。例えば、オンプレミス環境またはレガシーアプリケーションとの互換性があるように、DB インスタンスのタイムゾーンで変更が必要になることがあります。タイムゾーンオプションでは、ホストレベルでタイムゾーンが変更されます。タイムゾーンを変更すると、SYSDATE や SYSTIMESTAMP など、すべての日付列および値に影響を与えます。

タイムゾーンオプションは、rdsadmin_util.alter_db_time_zone コマンドと異なります。alter_db_time_zone コマンドによって変更されるタイムゾーンは、一定のデータタイプのみ対象です。タイムゾーンオプションでは、すべての日付列および値のタイムゾーンが変更されます。alter_db_time_zone の詳細については、「データベースタイムゾーンの設定」を参照してください。アップグレードに関する考慮事項の詳細については、「タイムゾーンに関する考慮事項」を参照してください。

タイムゾーンの設定に関する制限

タイムゾーンオプションは、固定かつ永続オプションです。したがって、以下の操作を行うことはできません。

- タイムゾーンオプションを追加した後、オプショングループからオプションを削除します。
- DB インスタンスに追加したオプショングループを削除する。
- オプションのタイムゾーン設定を別のタイムゾーンに変更する。

タイムゾーンの設定に関する推奨事項

タイムゾーンオプションを運用データベースに追加する前に、次の操作をお勧めします。

- インスタンスの DB スナップショットを取得します。誤ってタイムゾーンを設定した場合、DB インスタンスを以前のタイムゾーン設定に戻す必要があります。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
- タイムゾーンオプションを DB インスタンスに追加します。タイムゾーンオプションを追加すると、システムの日付を使用して日付または時刻を追加するテーブルに問題が発生することがあります。テストインスタンスでデータとアプリケーションを分析することをお勧めします。これにより、本番稼働用インスタンスのタイムゾーンを変更した場合の影響を評価できます。

DB インスタンスでデフォルトのオプショングループを使用している場合は、以下のステップに従います。

1. インスタンスの DB スナップショットを取得します。
2. タイムゾーンオプションを DB インスタンスに追加します。

DB インスタンスでデフォルト以外のオプショングループを使用している場合は、以下のステップに従います。

1. インスタンスの DB スナップショットを取得します。
2. 新しいオプショングループを作成します。
3. タイムゾーンオプションを、既存のオプショングループに現在関連付けられている他のすべてのオプションとともに追加します。

   これにより、タイムゾーンオプションを有効にしている間に既存のオプションがアンインストールされるのを防ぐことができます。
4. DB インスタンスにオプショングループを追加します。

タイムゾーンオプション設定

Amazon RDS は、タイムゾーンオプションの次の設定をサポートします。

オプション設定	有効な値	説明
TIME_ZONE	利用可能なタイムゾーンの例利用できるタイムゾーンの一覧については、「利用可能なタイムゾーン」を参照してください。	DB インスタンスの新しいタイムゾーン。

タイムゾーンオプションの追加

タイムゾーンオプションを DB インスタンスに追加するには、次のステップを完了します。

1. (推奨) DB インスタンスのスナップショットを取ります。
2. 次のいずれかのタスクを行います。
   - 新しいオプショングループを初めから作成します。詳細については、「オプショングループを作成する」を参照してください。

- AWS CLI または API を使用して既存のオプショングループをコピーします。詳細については、「オプショングループをコピーする」を参照してください。
- デフォルト以外の既存のオプショングループを再利用します。ベストプラクティスは、DB インスタンスまたはスナップショットに現在関連付けられていないオプショングループを使用することです。

3. 前のステップからオプショングループに新しいオプションを追加します。
4. 現在 DB インスタンスに関連付けられているオプショングループでオプションが有効になっている場合は、これらのオプションを新しいオプショングループに追加します。この戦略は、新しいオプションを有効にしている間に既存のオプションがアンインストールされるのを防ぐことができます。
5. DB インスタンスに新しいオプショングループを追加します。

タイムゾーンオプションを追加すると、DB インスタンスが自動的に再起動する際に短い停止が発生します。

コンソール

タイムゾーンオプションをオプショングループに追加して、DB インスタンスに関連付けるには

1. RDS コンソールで、[オプショングループ] を選択します。
2. オプションを追加するオプショングループの名前を選択します。
3. [オプションを追加] を選択します。
4. [オプション名] として [タイムゾーン] を選択し、オプション設定を行います。
5. オプショングループを新規または既存の DB インスタンスに関連付けます。

   - 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。既存の DB インスタンスに新しいオプションを追加すると、DB インスタンスが自動的に再起動している間に短い停止が発生します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

以下の例では、AWS CLI の add-option-to-option-group コマンドを使用して、Timezone オプションと TIME_ZONE オプションの設定をオプショングループ myoptiongroup に追加しています。タイムゾーンは Africa/Cairo に設定されています。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/
Cairo}]" \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/
Cairo}]" ^
    --apply-immediately
```

タイムゾーン設定の変更

タイムゾーンオプションは、固定かつ永続オプションです。オプショングループに追加したオプションを削除することはできません。DB インスタンスに追加したオプショングループを削除することはできません。オプションのタイムゾーン設定を別のタイムゾーンへと変更することはできません。タイムゾーンオプションが正しく設定されていない場合は、タイムゾーンオプション追加前の DB インスタンスのスナップショットを復元します。

タイムゾーンオプションの削除

タイムゾーンオプションは、固定かつ永続オプションです。オプショングループに追加したオプションを削除することはできません。DB インスタンスに追加したオプショングループを削除することはできません。タイムゾーンオプションを削除するには、タイムゾーンオプション追加前の DB インスタンスのスナップショットを復元します。

利用可能なタイムゾーン

タイムゾーンオプションには、以下の値を使用できます。

ゾーン	タイムゾーン
アフリカ	Africa/Cairo、Africa/Casablanca、Africa/Harare、Africa/Lagos、Africa/Luanda、Africa/Monrovia、Africa/Nairobi、Africa/Tripoli、Africa/Windhoek
南北アメリカ大陸	America/Araguaina、America/Argentina/Buenos_Aires、America/Asuncion、America/Bogota、America/Caracas、America/Chicago、America/Chihuahua、America/Cuiaba、America/Denver、America/Detroit、America/Fortaleza、America/Godthab、America/Guatemala、America/Halifax、America/Lima、America/Los_Angeles、America/Manaus、America/Matamoros、America/Mexico_City、America/Monterrey、America/Montevideo、America/New_York、America/Phoenix、America/Santiago、America/Sao_Paulo、America/Tijuana、America/Toronto
アジア	Asia/Amman、Asia/Ashgabat、Asia/Baghdad、Asia/Baku、Asia/Bangkok、Asia/Beirut、Asia/Calcutta、Asia/Damascus、Asia/Dhaka、Asia/Hong_Kong、Asia/Irkutsk、Asia/Jakarta、Asia/Jerusalem、Asia/Kabul、Asia/Karachi、Asia/Kathmandu、Asia/Kolkata、Asia/Krasnoyarsk、Asia/Magadan、Asia/Manila、Asia/Muscat、Asia/Novosibirsk、Asia/Rangoon、Asia/Riyadh、Asia/Seoul、Asia/Shanghai、Asia/Singapore、Asia/Taipei、Asia/Tehran、Asia/Tokyo、Asia/Ulaanbaatar、Asia/Vladivostok、Asia/Yakutsk、Asia/Yerevan
大西洋	Atlantic/Azores、Atlantic/Cape_Verde
オーストラリア	Australia/Adelaide、Australia/Brisbane、Australia/Darwin、Australia/Eucla、Australia/Hobart、Australia/Lord_Howe、Australia/Perth、Australia/Sydney
ブラジル	Brazil/DeNoronha、Brazil/East
カナダ	Canada/Newfoundland、Canada/Saskatchewan
ETC	Etc/GMT-3
欧州	Europe/Amsterdam、Europe/Athens、Europe/Berlin、Europe/Dublin、Europe/Helsinki、Europe/Kaliningrad、Europe/London、Europe/Madrid、Europe/Moscow、Europe/Paris、Europe/Prague、Europe/Rome、Europe/Sarajevo

ゾーン	タイムゾーン
太平洋	Pacific/Apia、Pacific/Auckland、Pacific/Chatham、Pacific/Fiji、Pacific/Guam、Pacific/Honolulu、Pacific/Kiritimati、Pacific/Marquesas、Pacific/Samoa、Pacific/Tongatapu、Pacific/Wake
米国	US/Alaska、US/Central、US/East-Indiana、US/Eastern、US/Pacific
UTC	UTC

Oracle のタイムゾーンファイルの自動アップグレード

TIMEZONE_FILE_AUTOUPGRADE オプションを使用すると、現在のタイムゾーンファイルを RDS for Oracle DB インスタンスの最新バージョンにアップグレードできます。

トピック

- Oracle のタイムゾーンファイルの概要
- タイムゾーンファイルを更新する戦略
- タイムゾーンファイル更新時のダウンタイム
- タイムゾーンファイル更新の準備
- タイムゾーンファイル自動アップグレードオプションの追加
- タイムゾーンファイル更新後のデータのチェック

Oracle のタイムゾーンファイルの概要

Oracle Database のタイムゾーンファイルには、次の情報が格納されます。

- 協定世界時 (UTC) との時差
- サマータイム (DST) の移行時期
- 標準時と DST の略語

Oracle Database には、複数のバージョンのタイムゾーンファイルが用意されています。オンプレミス環境で Oracle Database を作成するときには、タイムゾーンファイルのバージョンを選択します。詳細については、「Oracle Database グローバリゼーションサポートガイド」の「タイムゾーンファイルの選択」を参照してください。

DST のルールが変更された場合、Oracle は新しいタイムゾーンファイルを公開します。Oracle は、四半期ごとの Release Updates (RU) および Release Update Revisions (RUR) のスケジュールとは関係なく、これらの新しいタイムゾーンファイルをリリースします。タイムゾーンファイルは、データベースホストのディレクトリ $ORACLE_HOME/oracore/zoneinfo/ にあります。タイムゾーンファイル名は、DSTv35 のような形式 DSTv #####を使用します。

タイムゾーンファイルがデータ転送に与える影響

Oracle Database では、TIMESTAMP WITH TIME ZONE データ型は、タイムスタンプとタイムゾーンのデータを保存します。TIMESTAMP WITH TIME ZONE データ型のデータは、関連付けられたタ

イムゾーンファイルバージョンのルールを使用します。そのため、タイムゾーンファイルを更新すると、既存の TIMESTAMP WITH TIME ZONE データが影響を受けます。

異なるバージョンのタイムゾーンファイルを使用するデータベース間でデータを転送すると、問題が発生する可能性があります。例えば、ターゲットデータベースよりも高いバージョンのタイムゾーンファイルを持つソースデータベースからデータをインポートすると、データベースは ORA-39405 エラーを発行します。以前は、次のいずれかのテクニックを使用して、このエラーを回避する必要がありました。

- 必要なタイムゾーンファイルを使用して RDS for Oracle DB インスタンスを作成し、ソースデータベースからデータをエクスポートしてから、新しいデータベースにインポートします。
- AWS DMS または論理的なレプリケーションを使用して、データを移行します。

TIMEZONE_FILE_AUTOUPGRADE オプションを使用した自動アップデート

RDS for Oracle DB インスタンスにアタッチされたオプショングループに TIMEZONE_FILE_AUTOUPGRADEオプションが含まれている場合、RDS はタイムゾーンファイルを自動的に更新します。Oracle データベースが同じバージョンのタイムゾーンファイルを使用するようにすることで、異なる環境間でデータを移動する際に時間のかかる手動操作を回避できます。TIMEZONE_FILE_AUTOUPGRADE オプションは、コンテナデータベース (CDB) と非 CDB の両方に対してサポートされています。

TIMEZONE_FILE_AUTOUPGRADE オプションをオプショングループに追加するときには、オプションをすぐに追加するか、メンテナンスウィンドウで追加するかを選択できます。DB インスタンスが新しいオプションを適用すると、RDS は新しい DSTv #####ファイルをインストールできるかどうか確認します。対象の DST #####は以下の内容によって異なります。

- DB インスタンスが現在実行中のマイナーエンジンバージョン
- DB インスタンスのアップグレード先のマイナーエンジンバージョン

例えば、現在のタイムゾーンファイルのバージョンは DSTv33 である可能性があります。RDS がオプショングループに更新を適用すると、DSTv34 が DB インスタンスファイルシステムで現在利用可能であると判断される場合があります。その後、RDS はタイムゾーンファイルを自動的に DSTv34 に更新します。

サポートされる RDS リリース更新で利用可能な DST バージョンを確認するには、「Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle」(Amazon Relational Database Service (Amazon RDS) for Oracle リリースノート) のパッチを参照してください。例

えば、version 19.0.0.0.ru-2022-10.rur-2022-10.r1 には、パッチ 34533061: RDBMS - DSTV39 UPDATE - TZDATA2022C がリストされています。

タイムゾーンファイルを更新する戦略

DB エンジンをアップグレードし、オプショングループに TIMEZONE_FILE_AUTOUPGRADE オプションを追加するオペレーションは個別に行います。TIMEZONE_FILE_AUTOUPGRADE オプションを追加すると、最新のタイムゾーンファイルがある場合は、タイムゾーンファイルの更新が開始されます。すぐに、または次のメンテナンスウィンドウで、次のコマンドを実行します (関連するオプションのみが表示されます)。

- 次の RDS CLI コマンドを使用してのみ DB エンジンをアップグレードします。

```
modify-db-instance --engine-version name ...
```

- 次の CLI コマンドを使用してのみ TIMEZONE_FILE_AUTOUPGRADE オプションを追加します。

```
add-option-to-option-group --option-group-name name --options
 OptionName=TIMEZONE_FILE_AUTOUPGRADE ...
```

- 次の CLI コマンドを使用して、DB エンジンをアップグレードし、インスタンスに新しいオプショングループを追加します。

```
modify-db-instance --engine-version name --option-group-name name ...
```

更新戦略は、データベースとタイムゾーンファイルを一緒にアップグレードするか、これらのオペレーションのいずれかのみを実行するかによって異なります。オプショングループを更新し、別の API オペレーションで DB エンジンをアップグレードすると、DB エンジンをアップグレードするときにタイムゾーンファイルの更新が現在進行中になる可能性があることに注意してください。

このセクションの例では、以下を前提とします。

- DB インスタンスに現在関連付けられているオプショングループに、まだ TIMEZONE_FILE_AUTOUPGRADE が追加されていません。
- ご使用の DB インスタンスはデータベースバージョン 19.0.0.0.ru-2019-07.rur-2019-07.r1 およびタイムゾーンファイル DSTv33 を使用しています。
- DB インスタンスファイルシステムには、ファイル DSTv34 が含まれています。
- リリース更新プログラム 19.0.0.0.ru-2022-10.rur-2022-10.r1 には DSTv35 が含まれています。

タイムゾーンファイルを更新するには、以下の戦略を使用できます。

トピック

- エンジンをアップグレードせずにタイムゾーンファイルを更新する
- タイムゾーンファイルと DB エンジンバージョンをアップグレードする
- タイムゾーンファイルを更新せずに、DB エンジンのバージョンをアップグレードする

エンジンをアップグレードせずにタイムゾーンファイルを更新する

このシナリオでは、データベースに DSTv33 を使用していますが、ご使用の DB インスタンスファイルシステムでは DSTv34 が使用できます。DB インスタンスで使用されるタイムゾーンファイルを DSTv33 から DSTv34 に更新しても、エンジンは DSTv35 を含む新しいマイナーバージョンにアップグレードしたくありません。

`add-option-to-option-group` コマンドで、DB インスタンスで使用されるオプショングループに `TIMEZONE_FILE_AUTOUPGRADE` を追加します。オプションをすぐに追加するか、メンテナンスウィンドウまで遅延するかを指定します。`TIMEZONE_FILE_AUTOUPGRADE` オプションを適用すると、RDS は次の処理を行います。

1. 新しい DST バージョンをチェックする。
2. DSTv34 がファイルシステムで使用可能であることを決定する。
3. タイムゾーンファイルをすぐに更新する。

タイムゾーンファイルと DB エンジンバージョンをアップグレードする

このシナリオでは、データベースに DSTv33 を使用していますが、ご使用の DB インスタンスファイルシステムでは DSTv34 が使用できます。DB エンジンを DSTv35 を含むマイナーバージョン 19.0.0.0.ru-2022-10.rur-2022-10.r1 にアップグレードし、エンジンのアップグレード中にタイムゾーンファイルを DSTv35 に更新します。したがって、目標は DSTv34 をスキップして、タイムゾーンファイルを DSTv35 に直接更新することです。

エンジンとタイムゾーンファイルを一緒にアップグレードするには、`--option-group-name` および `--engine-version` オプションを使用して `modify-db-instance` を実行します。コマンドはすぐに実行することも、メンテナンスウィンドウ に延期することもできます。In `--option-group-name` は、`TIMEZONE_FILE_AUTOUPGRADE` オプションを含むオプショングループを指定します。例えば、次のようになります。

```
aws rds modify-db-instance
```

```
    --db-instance-identifier my-instance \
    --engine-version new-version \
    ----option-group-name og-with-timezone-file-autoupgrade \
    --apply-immediately
```

RDS は、エンジンの 19.0.0.0.ru-2022-10.rur-2022-10.r1 へのアップグレードを開始します。TIMEZONE_FILE_AUTOUPGRADE オプションが適用されると、RDS は新しい DST バージョンを確認して、DSTv35 が 19.0.0.0.ru-2022-10.rur-2022-10.r1 で使用可能であることがわかると、すぐに DSTv35 に更新を開始します。

エンジンをすぐにアップグレードし、タイムゾーンファイルをアップグレードするには、オペレーションを順番に実行します。

1. 次の CLI コマンドを使用してのみ DB エンジンをアップグレードします。

```
aws rds modify-db-instance \
    --db-instance-identifier my-instance \
    --engine-version new-version \
    --apply-immediately
```

2. 次の CLI コマンドを使用して、インスタンスにアタッチされたオプショングループに TIMEZONE_FILE_AUTOUPGRADE オプションを追加します。

```
aws rds add-option-to-option-group \
    --option-group-name og-in-use-by-your-instance \
    --options OptionName=TIMEZONE_FILE_AUTOUPGRADE \
    --apply-immediately
```

タイムゾーンファイルを更新せずに、DB エンジンのバージョンをアップグレードする

このシナリオでは、データベースに DSTv33 を使用していますが、ご使用の DB インスタンスファイルシステムでは DSTv34 が使用できます。DB エンジンを DSTv35 を含むバージョン 19.0.0.0.ru-2022-10.rur-2022-10.r1 にアップグレードしても、タイムゾーンファイル DSTv33 を保持します。次の理由から、この戦略が必要になる場合があります。

- データが TIMESTAMP WITH TIME ZONE データ型を使用しない。
- データは TIMESTAMP WITH TIME ZONE データ型を使用できますが、データがタイムゾーン変更の影響を受けません。

- 追加のダウンタイムを許容できないため、タイムゾーンファイルの更新を延期する必要があります。

戦略は、次のいずれが該当するによって異なります。

- DB インスタンスは TIMEZONE_FILE_AUTOUPGRADE を含むオプショングループに関連付けられていません。modify-db-instance コマンドでは、RDS がタイムゾーンファイルを更新しないように、新しいオプショングループを指定しないでください。
- DB インスタンスは現在、TIMEZONE_FILE_AUTOUPGRADE を含むオプショングループに関連付けられています。単一の modify-db-instance コマンド内で、TIMEZONE_FILE_AUTOUPGRADE を含まないオプショングループに DB インスタンスを関連付けてから、DB エンジンを 19.0.0.0.ru-2022-10.rur-2022-10.r1 にアップグレードします。

タイムゾーンファイル更新時のダウンタイム

RDS がタイムゾーンファイルを更新すると、TIMESTAMP WITH TIME ZONE を使用する既存のデータが変更される可能性があります。この場合、主な考慮事項は、ダウンタイムです。

> **Warning**
>
> TIMEZONE_FILE_AUTOUPGRADE オプションを追加すると、エンジンのアップグレードのためにダウンタイムが長くなる可能性があります。ラージデータベースのタイムゾーンデータの更新には、数時間または数日かかることがあります。

タイムゾーンファイルの更新の長さは、次のような要因によって異なります。

- データベース内の TIMESTAMP WITH TIME ZONE データの量
- DB インスタンスの設定
- DB インスタンスクラス
- ストレージ設定
- データベース設定
- データベースパラメータ設定

次の操作を行うと、追加のダウンタイムが発生する可能性があります。

- DB インスタンスが古いタイムゾーンファイルを使用しているときには、オプションをオプショングループに追加します。
- 新しいエンジンバージョンにタイムゾーンファイルの新しいバージョンが含まれている場合は、Oracle データベースエンジンをアップグレードします。

> **Note**
>
> タイムゾーンファイルの更新中に、RDS for Oracle は `PURGE DBA_RECYCLEBIN` を呼び出します。

タイムゾーンファイル更新の準備

タイムゾーンファイルのアップグレードには、準備とアップグレードの 2 つの段階があります。必須ではありませんが、準備ステップを実行することを強く推奨します。このステップでは、PL/SQL ステップ `DBMS_DST.FIND_AFFECTED_TABLES` を実行することによって影響を受けるデータを調べます。準備ウィンドウの詳細については、Oracle Database ドキュメントの Upgrading the Time Zone File and Timestamp with Time Zone Data を参照してください。

タイムゾーンファイル更新の準備を行うには

1. SQL クライアントを使用して、Oracle Database に接続します。
2. 現在使用されているタイムゾーンファイルのバージョンを確認します。

```
SELECT * FROM V$TIMEZONE_FILE;
```

3. DB インスタンスで使用可能な最新のタイムゾーンファイルのバージョンを決定します。

```
SELECT DBMS_DST.GET_LATEST_TIMEZONE_VERSION FROM DUAL;
```

4. `TIMESTAMP WITH LOCAL TIME ZONE` または `TIMESTAMP WITH TIME ZONE` 型の列を持つテーブルの合計サイズを決定します。

```
SELECT SUM(BYTES)/1024/1024/1024 "Total_size_w_TSTZ_columns_GB"
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE 'TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
         (SELECT OWNER, TABLE_NAME
          FROM   DBA_TAB_COLUMNS
```

```
          WHERE  DATA_TYPE LIKE 'TIMESTAMP%TIME ZONE');
```

5. TIMESTAMP WITH LOCAL TIME ZONE または TIMESTAMP WITH TIME ZONE 型の列を持つセグメントの名前とサイズを決定します。

```
SELECT OWNER, SEGMENT_NAME, SUM(BYTES)/1024/1024/1024
 "SEGMENT_SIZE_W_TSTZ_COLUMNS_GB"
FROM   DBA_SEGMENTS
WHERE  SEGMENT_TYPE LIKE 'TABLE%'
AND    (OWNER, SEGMENT_NAME) IN
         (SELECT OWNER, TABLE_NAME
          FROM   DBA_TAB_COLUMNS
          WHERE  DATA_TYPE LIKE 'TIMESTAMP%TIME ZONE')
GROUP BY OWNER, SEGMENT_NAME;
```

6. 準備ステップを実行します。

   - 手順 DBMS_DST.CREATE_AFFECTED_TABLE で、影響を受けるデータを保存するためのテーブルを作成します。このテーブル名を DBMS_DST.FIND_AFFECTED_TABLES 手順に渡します。詳細については、Oracle Database ドキュメントの CREATE_AFFECTED_TABLE Procedure を参照してください。
   - このプロシージャ CREATE_ERROR_TABLE は、エラーを記録するテーブルを作成します。詳細については、Oracle Database ドキュメントの CREATE_ERROR_TABLE Procedure を参照してください。

   次の例では、影響を受けるデータとエラーテーブルを作成し、影響を受けるすべてのテーブルを検索します。

```
EXEC DBMS_DST.CREATE_ERROR_TABLE('my_error_table')
EXEC DBMS_DST.CREATE_AFFECTED_TABLE('my_affected_table')

EXEC DBMS_DST.BEGIN_PREPARE(new_version);
EXEC DBMS_DST.FIND_AFFECTED_TABLES('my_affected_table', TRUE, 'my_error_table');
EXEC DBMS_DST.END_PREPARE;

SELECT * FROM my_affected_table;
SELECT * FROM my_error_table;
```

7. 影響を受けるテーブルとエラーテーブルをクエリします。

```
SELECT * FROM my_affected_table;
```

```
SELECT * FROM my_error_table;
```

タイムゾーンファイル自動アップグレードオプションの追加

オプションをオプショングループに追加すると、オプショングループは次のいずれかの状態になります。

- 既存のオプショングループは、現在、少なくとも 1 つの DB インスタンスにアタッチされています。オプションを追加すると、このオプショングループを使用するすべての DB インスタンスが自動的に再起動します。これにより、短時間の停止が発生します。
- 既存のオプショングループは、DB インスタンスにアタッチされていません。オプションを追加してから、既存のオプショングループを既存の DB インスタンスまたは新しい DB インスタンスに関連付ける予定です。
- 新しいオプショングループを作成し、オプションを追加します。新しいオプショングループを既存の DB インスタンスまたは新しい DB インスタンスに関連付ける予定です。

コンソール

タイムゾーンファイル自動アップグレードオプションを DB インスタンスに追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[オプショングループ] を選択します。
3. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] (エンジン) として、DB インスタンスの Oracle Database エディションを選択します。

   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
4. 変更するオプショングループを選択し、[オプションの追加] を選択します。
5. [Add option(オプションの追加)] ウィンドウで、以下の操作を行います。

a. [TIMEZONE_FILE_AUTOUPGRADE] を選択します。
b. オプションを追加後すぐに、関連付けられているすべての DB インスタンスに対して有効にするには、[Apply Immediately] で [Yes] を選択します。[No] を選択した場合 (デフォルト)、オプションは次のメンテナンス時間中に、関連付けられている各 DB インスタンスに対して有効になります。

6. 設定が希望どおりになったら、[オプションの追加] を選択します。

AWS CLI

次の例では、AWS CLI の add-option-to-option-group コマンドを使用して、TIMEZONE_FILE_AUTOUPGRADE オプションを myoptiongroup オプショングループに追加しています。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=TIMEZONE_FILE_AUTOUPGRADE" \
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=TIMEZONE_FILE_AUTOUPGRADE" ^
    --apply-immediately
```

タイムゾーンファイル更新後のデータのチェック

タイムゾーンファイルの更新後に、データをチェックすることをお勧めします。準備ステップでは、RDS for Oracle によって次のテーブルが自動的に作成されます。

- `rdsadmin.rds_dst_affected_tables` - 更新の影響を受けるデータを含むテーブルを一覧表示します
- `rdsadmin.rds_dst_error_table` - 更新中に生成されたエラーを一覧表示します

これらのテーブルは、準備ウィンドウで作成するテーブルから独立しています。更新の結果を表示するには、次のようにテーブルをクエリします。

```
SELECT * FROM rdsadmin.rds_dst_affected_tables;
SELECT * FROM rdsadmin.rds_dst_error_table;
```

影響を受けるデータおよびエラーテーブルのスキーマの詳細については、Oracle ドキュメントの FIND_AFFECTED_TABLES Procedure を参照してください。

Oracle Transparent Data Encryption

Amazon RDS は、Oracle Enterprise Edition で使用可能な Oracle Advanced Security オプションの機能である Oracle Transparent Data Encryption (TDE) をサポートしています。この機能は、ストレージへの書き込み前に自動的にデータを暗号化し、ストレージからのデータの読み取り時に自動的にデータを復号します。このオプションは、Bring Your Own License (BYOL) モデルでのみサポートされています。

TDE は、データファイルやバックアップが第三者によって取得された場合に機密データを暗号化する必要があるシナリオで便利です。TDE は、セキュリティ関連の規制に準拠する必要がある場合にも役立ちます。

Oracle Database TDE に関する詳細な説明は、このガイドでは取り上げません。詳細については、次の Oracle Database リソースを参照してください。

- Oracle Database ドキュメントの「透過的データ暗号化の概要」
- Oracle Database ドキュメントの「Oracle アドバンストセキュリティ」
- Oracle ホワイトペーパーの「Oracle アドバンストセキュリティの透過的データ暗号化のベストプラクティス」

RDS for Oracle で TDE を使用する方法の詳細については、次のブログを参照してください。

- Amazon RDS の Oracle Database 暗号化オプション
- を使用して、ダウンタイムを短縮してクロスアカウント TDE 対応 Amazon RDS for Oracle DB インスタンスを移行するAWS DMS

TDE 暗号化モード

Oracle Transparent Data Encryption では、TDE テーブルスペース暗号化と TDE 列暗号化の 2 つの暗号化モードがサポートされます。TDE テーブルスペース暗号化は、アプリケーションテーブル全体の暗号化に使用されます。TDE 列暗号化は、機密データを含む個々のデータ要素を暗号化するために使用されます。TDE のテーブルスペース暗号化と列暗号化の両方を使用するハイブリッド暗号化ソリューションを適用することもできます。

> **Note**
>
> DB インスタンス用の Oracle Wallet と TDE マスターキーは Amazon RDS によって管理されます。コマンド `ALTER SYSTEM set encryption key` を使用して暗号化キーを設定する必要はありません。

TDE オプションを有効にすると、次のコマンドを使用して Oracle Wallet のステータスを確認できます。

```
SELECT * FROM v$encryption_wallet;
```

暗号化されたテーブルスペースを作成するには、次のコマンドを使用します。

```
CREATE TABLESPACE encrypt_ts ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

暗号アルゴリズムを指定するには、以下のコマンドを実行します。

```
CREATE TABLESPACE encrypt_ts ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

テーブルスペースを暗号化するための前述のステートメントは、オンプレミスの Oracle データベースで使用するステートメントと同じです。

TDE オプションの制限事項

オプションは永続的かつ固定的です。DB インスタンスを TDE オプションが有効になっているオプショングループに関連付けると、次のアクションを実行することはできません。

- 現在関連付けられているオプショングループの TDE オプションを無効にします。
- TDE オプションを含まない異なるオプショングループに DB インスタンスを関連付けます。
- TDE オプションを使用している DB スナップショットを共有します。DB スナップショットの共有の詳細については、「Amazon RDS の DB スナップショットを共有する」を参照してください。

永続的および固定的オプションの詳細については、「永続オプションと固定オプション」を参照してください。

DB インスタンスが TDE を使用しているかどうかの確認

TDE オプションを含むオプショングループに DB インスタンスが関連付けられているかどうかを調べるたい場合があります。DB インスタンスが関連付けられているオプショングループを表示するには、RDS コンソール、AWS CLI コマンド (describe-db-instance)、または API オペレーション DescribeDBInstances を使用します。

TDE オプションの追加

> **Important**
>
> TDE オプションは永続的であり、DB インスタンスから削除することはできません。この決定を後で元に戻す必要がある場合は、すべてのデータを復号し、エクスポートして、TDE を使用しない新しい DB インスタンスにインポートする必要があります。TDE を有効にする前に、適切に計画します。

TDE オプションを DB インスタンスに追加するには、次のステップを完了します。

1. (推奨) DB インスタンスのスナップショットを取ります。
2. 次のいずれかのタスクを行います。
   - 新しいオプショングループを初めから作成します。詳細については、「オプショングループを作成する」を参照してください。
   - AWS CLI または API を使用して既存のオプショングループをコピーします。詳細については、「オプショングループをコピーする」を参照してください。
   - デフォルト以外の既存のオプショングループを再利用します。ベストプラクティスは、DB インスタンスまたはスナップショットに現在関連付けられていないオプショングループを使用することです。
3. 前のステップからオプショングループに新しいオプションを追加します。
4. 現在 DB インスタンスに関連付けられているオプショングループでオプションが有効になっている場合は、これらのオプションを新しいオプショングループに追加します。この戦略は、新しいオプションを有効にしている間に既存のオプションがアンインストールされるのを防ぐことができます。
5. DB インスタンスに新しいオプショングループを追加します。

コンソール

TDE オプションをオプショングループに追加して、DB インスタンスに関連付けるには

1. RDS コンソールで、[オプショングループ] を選択します。
2. オプションを追加するオプショングループの名前を選択します。
3. [オプションを追加] を選択します。
4. [オプション名] として [TDE] を選択し、オプション設定を行います。
5. [オプションを追加] を選択します。

   > **Important**
   >
   > 1 つ以上の DB インスタンスにアタッチされているオプショングループに TDE オプションを追加すると、すべての DB インスタンスが自動的に再起動する間に短い停止が発生します。

   オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
6. オプショングループを新規または既存の DB インスタンスに関連付けます。

   - 新しい DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
   - 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。DB インスタンスは、この操作の一部として再起動されません。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

AWS CLI

次の例では、AWS CLI add-option-to-option-group コマンドを使用して、TDE オプションを myoptiongroup という名前のオプショングループに追加します。

Linux、macOS、Unix の場合:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=TDE" \
```

```
    --apply-immediately
```

Windows の場合:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=TDE" ^
    --apply-immediately
```

TDE オプションを含まない DB インスタンスへのデータのコピー

DB インスタンスから TDE オプションを削除したり、TDE オプションを含まないオプショングループに関連付けることはできません。TDE オプションを含まないインスタンスにデータを移行するには、次の手順を実行します。

1. DB インスタンス上のデータを復号します。
2. TDE が有効になっているオプショングループに関連付けられていない新しい DB インスタンスにデータをコピーします。
3. 元の DB インスタンスを削除します。

新しいインスタンスには、前の DB インスタンスと同じ名前を付けることができます。

Oracle Data Pump で TDE を使用する場合の考慮事項

暗号化されたダンプファイルをインポートまたはエクスポートするには、Oracle Data Pump を使用します。Amazon RDS は、Oracle Data Pump 用のパスワード暗号化モード (ENCRYPTION_MODE=PASSWORD) をサポートしています。Oracle Data Pump 用の透過的暗号化モード (ENCRYPTION_MODE=TRANSPARENT) は、Amazon RDS でサポートされていません。詳細については、「Oracle Data Pump を使用したインポート」を参照してください。

Oracle UTL_MAIL

Amazon RDS では、UTL_MAIL オプションおよび SMTP サーバーを使用して Oracle UTL_MAIL をサポートしています。UTL_MAIL パッケージを使用して、データベースから直接電子メールを送信できます。Amazon RDS では、Oracle の次のバージョンで UTL_MAIL をサポートしています。

- Oracle Database 21c (21.0.0.0)、すべてのバージョン
- Oracle Database 19c (19.0.0.0)、すべてのバージョン

UTL_MAIL を使用する場合の制約をいくつか次に示します。

- UTL_MAIL は Transport Layer Security (TLS) をサポートしていないため、E メールは暗号化されません。

  カスタムの Oracle ウォレットを作成およびアップロードして、リモート SSL/TLS リソースに安全に接続するには、「証明書と Oracle ウォレットを使用した、UTL_HTTP アクセスの設定」の手順を行います。

  Wallet に必要な固有の証明書は、サービスによって異なります。AWS のサービスの場合は、通常、Amazon Trust Services リポジトリにあります。
- UTL_MAIL は、SMTP サーバーでの認証をサポートしていません。
- E メールでは 1 つの添付ファイルのみ送信できます。
- 32 K より大きい添付ファイルを送信することはできません。
- ASCII および Extended Binary Coded Decimal Interchange Code (EBCDIC) 文字エンコードのみ使用できます。
- SMTP ポート (25) は、Elastic Network Interface 所有者のポリシーに基づいてスロットリングされます。

UTL_MAIL を有効にすると、DB インスタンスのマスターユーザーのみに実行権限が付与されます。必要に応じて、UTL_MAIL を使用できるよう、マスターユーザーは他のユーザーに実行権限を付与することができます。

> **Important**
>
> UTL_MAIL プロシージャの使用を追跡するため、Oracle の組み込み監査機能を有効にすることをお勧めします。

Oracle UTL_MAIL の前提条件

Oracle UTL_MAIL を使用するための前提条件は次のとおりです。

- 1 つ以上の SMTP サーバーと、対応する IP アドレスまたはパブリック/プライベートドメインネームサーバー (DNS) 名。カスタム DNS サーバーを通じて解決されるプライベート DNS 名については、「カスタム DNS サーバーのセットアップ」を参照してください。

Oracle UTL_MAIL オプションの追加

DB インスタンスに Oracle UTL_MAIL オプションを追加する一般的な手順は以下のとおりです。

1. 新しいオプショングループを作成するか、既存のオプショングループをコピーまたは変更します。
2. オプショングループに [] オプションを追加します。
3. オプショングループを DB インスタンスに関連付けます。

UTL_MAIL オプションの追加後、オプショングループがアクティブになるとすぐに、UTL_MAIL がアクティブになります。

UTL_MAIL オプションを DB インスタンスに追加するには

1. 使用するオプショングループを決定します。新しいオプショングループを作成することも、既存のオプショングループを使用することもできます。既存のオプショングループを使用する場合は、次のステップは飛ばしてください。または、次の設定でカスタム DB オプショングループを作成します。

   a. [Engine] で、使用する Oracle のエディションを選択します。
   b. [メジャーエンジンのバージョン] で、DB インスタンスのバージョンを選択します。

   詳細については、「オプショングループを作成する」を参照してください。
2. オプショングループに [UTL_MAIL] オプションを追加します。オプションの追加方法の詳細については、「オプショングループにオプションを追加する」を参照してください。
3. 新規または既存の DB インスタンスに、DB オプショングループを適用します。

   - 新規 DB インスタンスの場合は、インスタンスを起動するときにオプショングループを適用します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

- 既存の DB インスタンスの場合は、インスタンスを修正し、新しいオプショングループを添付することで、オプショングループを適用します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle UTL_MAIL の使用

UTL_MAIL オプションを有効にした後、それを使用する前に、SMTP サーバーを設定する必要があります。

SMTP_OUT_SERVER パラメータを有効な IP アドレスまたはパブリック DNS 名に設定することで、SMTP サーバーを設定します。SMTP_OUT_SERVER パラメータでは、複数のサーバーのアドレスのカンマ区切りリストを指定できます。初期のサーバーが利用できない場合、UTL_MAIL は次のサーバーを順に試します。

DB パラメータグループを使用して、DB インスタンスのデフォルトの SMTP_OUT_SERVER を設定できます。DB インスタンスで、データベースで以下のコードを実行して、セッションの SMTP_OUT_SERVER パラメータを設定できます。

```
ALTER SESSION SET smtp_out_server = mailserver.domain.com:25;
```

UTL_MAIL オプションが有効になり、SMTP_OUT_SERVER が設定されたら、SEND プロシージャを使用して E メールを送信できます。詳細については、Oracle ドキュメントの UTL_MAIL を参照してください。

Oracle UTL_MAIL オプションの削除

DB インスタンスから Oracle UTL_MAIL を削除できます。

DB インスタンスから UTL_MAIL を削除するには、次のいずれかを実行します。

- 複数の DB インスタンスから UTL_MAIL を削除するには、属しているオプショングループから UTL_MAIL オプションを削除します。この変更はそのオプショングループを使用するすべての DB インスタンスに影響します。詳細については、「オプショングループからオプションを削除する」を参照してください。
- 単一の DB インスタンスから UTL_MAIL を削除するには、DB インスタンスを変更し、UTL_MAIL オプションが含まれていない別のオプショングループを指定します。デフォルト (空) のオプショングループや別のカスタムオプショングループを指定できます。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

トラブルシューティング

Amazon RDS で UTL_MAIL を使用する際に生じる可能性がある問題は以下のとおりです。

- Throttling。SMTP ポート (25) は、Elastic Network Interface 所有者のポリシーに基づいてスロットリングされます。UTL_MAIL を使用して正常に E メールを送信できたにもかかわらず、`ORA-29278: SMTP transient error: 421 Service not available` エラーが表示された場合は、スロットリングされている可能性があります。E メール配信がスロットリングされた場合は、バックオフアルゴリズムを実装することをお勧めします。バックオフアルゴリズムの詳細については、「AWS でのエラーの再試行とエクスポネンシャルバックオフ」および「"throttling Maximum sending rate exceeded" エラーの対処法」を参照してください。

  この調整の削除をリクエストできます。詳細については、「EC2 インスタンスからポート 25 のスロットルを削除する方法を教えてください。」を参照してください。

Oracle XML DB

Oracle XML DB は、DB インスタンスにネイティブ XML サポートを追加します。XML DB を使用すると、構造化 XML または非構造化 XML とリレーショナルデータを保存および取得できます。XML DB プロトコルサーバーは、RDS for Oracle ではサポートされていません。

XML DB は、Oracle Database 12c 以降にプレインストールされています。したがって、オプショングループを使用して XML DB を追加機能として明示的にインストールする必要はありません。

XML DB を設定して使用する方法については、Oracle Database ドキュメントの「Oracle XML DB 開発者ガイド」を参照してください。

RDS for Oracle DB エンジンのアップグレード

Amazon RDS が新バージョンの Oracle Database をサポートするときには、DB インスタンスを新バージョンにアップグレードできます。Amazon RDS で使用できる Oracle のバージョンの詳細については、Amazon RDS for Oracle リリースノートを参照してください。

> **Important**
>
> RDS for Oracle Database の 11g、12c、および 18c はサポートされなくなりました。Oracle Database の 11g、12c、または 18c スナップショットを維持する場合は、それ以降のリリースにアップグレードできます。詳細については、「Oracle DB スナップショットのアップグレード」を参照してください。

トピック

- RDS for Oracle エンジンのアップグレードの概要
- Oracle のメジャーバージョンのアップグレード
- Oracle のマイナーバージョンのアップグレード
- Oracle のデータベースアップグレードに関する考慮事項
- Oracle DB アップグレードのテスト
- RDS for Oracle DB インスタンスバージョンのアップグレード
- Oracle DB スナップショットのアップグレード

RDS for Oracle エンジンのアップグレードの概要

RDS for Oracle DB インスタンスをアップグレードする前に、次の概念を理解しましょう。

トピック

- メジャーバージョンとマイナーバージョンのアップグレード
- RDS for Oracle のサポート日と必須のアップグレード
- Oracle エンジンのバージョン管理
- アップグレード前のチェックリスト
- エンジンのアップグレード中の自動スナップショット

- マルチ AZ 配置での Oracle のアップグレード
- リードレプリカでの Oracle のアップグレード
- アップグレード後の検証

メジャーバージョンとマイナーバージョンのアップグレード

メジャーバージョンは、1～2 年ごとにリリースされる Oracle Database のメジャーリリースです。Oracle Database 19c、Oracle Database 21c、および Oracle Database 26ai は、メジャーリリースです。

RDS for Oracle は、四半期ごとに、サポートされているすべてのメジャーエンジンに対して新しいマイナーエンジンバージョンをリリースします。リリースアップデート (RU) エンジンバージョンには、指定された四半期の RU パッチを含めることで Oracle からのバグ修正が組み込まれています。例えば、21.0.0.0.ru-2024-10.rur-2024-10.r1 は、2024 年 10 月の RU を組み込んだ Oracle Database 21c のマイナーバージョンです。

Supplemental パッチバンドル (SPB) エンジンバージョンは、Oracle Spatial、Oracle Data Pump、Oracle GoldenGate などの特定のユースケースで Oracle が推奨する追加のデータベースパッチを含む RU エンジンバージョンです。例えば、19.0.0.0.ru-2026-04.spb-1.r1 は、エンジンバージョン 19.0.0.0.ru-2026-04.rur-2026-04.r1 の RU パッチと Supplemental パッチを含むマイナーエンジンバージョンです。通常、RDS for Oracle は、対応する RU の 2～3 週間後に SPB をリリースします。RU と SPB の違いの説明については、「リリースアップデート (RU) と Supplemental パッチバンドル (SPB)」を参照してください。サポートされている RU と SPB については、「Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle」を参照してください。

RDS for Oracle は、DB インスタンスへの次のアップグレードをサポートします。

アップグレードタイプ	アプリケーションの互換性	アップグレード方法	サンプルのアップグレードパス
メジャーバージョン	メジャーバージョンのアップグレードによって、既存のアプリケーションと互換性のない変更が導入されることがあります。	手動のみ	Oracle Database 19c から Oracle Database 26ai へ
マイナーバージョン	マイナーバージョンのアップグレードには、既存のアプリケー	自動または手動	21.0.0.0.ru-2023-07.rur-2022-07.r1 か

アップグレードタイプ	アプリケーションの互換性	アップグレード方法	サンプルのアップグレードパス
	ションとの下位互換性がある変更のみが含まれます。		ら 21.0.0.0.ru-2023-10.rur-2022-10.r1 へ

> **Important**
>
> DB エンジンをアップグレードすると、停止が発生します。停止時間の長さは、エンジンのバージョンと DB インスタンスのサイズによって異なります。
> 本稼働データベースにアップグレードを適用する前に、アップグレードを徹底的にテストしてアプリケーションが正常に動作することを確認してください。詳細については、「Oracle DB アップグレードのテスト」を参照してください。

RDS for Oracle のサポート日と必須のアップグレード

RDS for Oracle のデータベースバージョンには、サポート予定日があります。RDS for Oracle DB エンジンのメジャーバージョンまたはマイナーバージョンのサポート終了日が近づくと、RDS は強制アップグレードとも呼ばれる必須アップグレードを開始します。RDS は、次の情報を公開します。

- 非推奨バージョンからサポートされているバージョンへの手動アップグレードを開始することを推奨する通知
- サポートされていないバージョンでインスタンスを作成できなくなる日付
- メンテナンス期間中に RDS がインスタンスをサポートされているバージョンに自動的にアップグレードする日付
- メンテナンス期間外に RDS がインスタンスをサポートされているバージョンに自動的にアップグレードする日付

> **Important**
>
> 強制アップグレードは、CloudFormation スタックに予期しない結果をもたらすことがあります。RDS を使用して DB インスタンスを自動的にアップグレードすると、CloudFormation で問題が発生する場合があります。

このセクションは、以下のトピックで構成されます。

トピック

- RDS for Oracle メジャーリリースのサポート日
- RDS for Oracle のマイナーバージョンのサポート日

RDS for Oracle メジャーリリースのサポート日

RDS for Oracle メジャーバージョンは、少なくとも対応する Oracle Database リリースバージョンのサポート終了日までは利用可能です。次の日付を参考にすると、テストおよびアップグレードのサイクルを計画することができます。これらの日付は、新しいバージョンへのアップグレードが必要になる可能性がある最も早い日付を表します。Amazon は、RDS for Oracle バージョンのサポートを当初発表よりも長く延長した場合、新しい日付を反映してこの表を更新するようにします。

> **Note**
>
> describe-db-major-engine-versions AWS CLI コマンドを実行するか、DescribeDBMajorEngineVersions RDS API オペレーションを使用して、Oracle データベースのメジャーバージョンを表示できます。

> **Note**
>
> Oracle Database 26ai は Enterprise Edition でのみ使用できます。

Oracle Database メジャーリリースバージョン	新しいバージョンへのアップグレード予定日
Oracle Database 19c	2029 年 12 月 31 日 BYOL プレミアサポート (延長サポートの手数料が免除) 2032 年 12 月 31 日 BYOL 延長サポート (追加料金) または無制限ライセンス契約 2029 年 12 月 31 日、ライセンス込み (LI)
Oracle Database 21c	2027 年 7 月 31 日 (延長サポートでは使用できません)

Oracle Database メジャーリリースバージョン	新しいバージョンへのアップグレード予定日
Oracle Database 26ai	2031 年 12 月 31 日、BYOL プレミアサポート付き 後日発表、BYOL 延長サポート付き (追加料金)

RDS は、新しいメジャーバージョンにアップグレードする必要がある少なくとも 12 か月前に通知します。通知には、重要なマイルストーンのタイミング、DB インスタンスへの影響、推奨されるアクションなど、アップグレードプロセスについての説明が記載されています。データベースをメジャーバージョンにアップグレードする前に、新しい RDS for Oracle バージョンに関するアプリケーションのテストを、完全に完了することをお勧めします。

この事前通知期間後は、それ以降のメジャーバージョンへの自動アップグレードが、古いバージョンを実行しているR DS for Oracle DB インスタンスに適用されることを想定してください。その場合は、スケジュールされたメンテナンスウィンドウ中にアップグレードがスタートされます。

詳細については、「My Oracle Support」の「現在のデータベースリリースのリリーススケジュール」を参照してください。

RDS for Oracle のマイナーバージョンのサポート日

場合によっては、RDS for Oracle のメジャーリリースのマイナーバージョンのサポートを終了します。RDS は、新しいマイナーバージョンにアップグレードする必要がある少なくとも 6 か月前に通知します。通知には、重要なマイルストーンのタイミング、廃止されたマイナーバージョンを実行している DB インスタンスへの影響、推奨されるアクションなど、アップグレードプロセスについての説明が記載されています。データベースを新しいマイナーバージョンにアップグレードする前に、新しい RDS for Oracle バージョンに関するアプリケーションのテストを、完全に完了することをお勧めします。

廃止およびサポートが終了したマイナーバージョンの詳細については、「Release notes for Amazon Relational Database Service (Amazon RDS) for Oracle」を参照してください。

Oracle エンジンのバージョン管理

DB エンジンのバージョン管理により、データベースエンジンにパッチを適用してアップグレードするタイミングと方法を制御できます。データベースエンジンのパッチバージョンとの互換性を維持する柔軟性が得られます。また、RDS for Oracle の新しいパッチバージョンを本稼働環境でデプロイ

する前にテストして、アプリケーションで動作することを確認できます。さらに、独自の条件やタイムラインでバージョンをアップグレードします。

> **Note**
>
> Amazon RDS では、Amazon RDS 固有の DB エンジンのバージョンを使用して、Oracle データベースの公式パッチを定期的に収集します。Amazon RDS Oracle 固有のエンジンのバージョンに含まれている Oracle のパッチに関するリストについては、「Amazon RDS for Oracle リリースノート」を参照してください。

アップグレード前のチェックリスト

RDS for Oracle DB インスタンスのメジャーバージョンアップグレードを実行する前に、以下の準備手順を完了します。

- アプリケーションがターゲットの Oracle Database バージョンと互換性があることを確認します。ターゲットバージョンに対してアプリケーションコード、クエリ、ストアドプロシージャをテストします。
- ターゲットバージョンの非推奨の初期化パラメータを確認します。サポートされなくなったパラメータを削除または置き換えます。
- オプショングループがターゲットバージョンと互換性があることを確認します。一部のオプションでは、更新が必要であったり、メジャーバージョンによって設定が異なったりします。ターゲットバージョンで使用できるオプションを一覧表示するには、describe-option-group-options AWS CLI コマンドを実行します。次に例を示します。

```
aws rds describe-option-group-options --engine-name oracle-ee --major-engine-version
 21
```

- パラメータグループがターゲットバージョンと互換性があることを確認します。一部のパラメータは、新しいバージョンでは有効範囲やデフォルト値が異なります。
- DB インスタンスのバックアップ保持期間が 0 より大きいことを確認します。これにより、Amazon RDS はアップグレード前の自動スナップショットを取得し、それを復旧に使用できます。
- リードレプリカのアップグレードを計画します。Amazon RDS は、ソース DB インスタンスのアップグレードが完了した後、リードレプリカを自動的にアップグレードします。レプリカのための追加のダウンタイムをメンテナンスウィンドウの計画に含めます。

エンジンのアップグレード中の自動スナップショット

Oracle DB インスタンスをアップグレードする際、スナップショットはアップグレードの問題に対する保護を提供します。DB インスタンスのバックアップ保持期間を 0 より大きく設定した場合、Amazon RDS はアップグレード中に以下の DB スナップショットを作成します。

1. アップグレードの変更が行われる前の DB インスタンスのスナップショット。アップグレードが失敗した場合、このスナップショットを復元して、古いバージョンを実行する DB インスタンスを作成できます。
2. アップグレード完了後の DB インスタンスのスナップショット。

> **Note**
>
> バックアップ保持期間を変更するには、「Amazon RDS DB インスタンスを変更する」を参照してください。

アップグレード後は、以前のエンジンバージョンに戻すことはできません。ただし、アップグレード前のスナップショットを復元することで、新しい Oracle DB インスタンスを作成できます。

アップグレードの失敗または問題から復旧するには、restore-db-instance-from-db-snapshot AWS CLI コマンドを使用して、アップグレード前の自動スナップショットを復元します。復元された DB インスタンスには新しいエンドポイントがあることに注意してください。復元が完了したら、新しい DB インスタンスエンドポイントを指すようにアプリケーションの接続文字列を更新します。

マルチ AZ 配置での Oracle のアップグレード

DB インスタンスがマルチ AZ 配置にある場合、Amazon RDS はプライマリとスタンバイの両方のレプリカをアップグレードします。オペレーティングシステムの更新が不要な場合は、プライマリとスタンバイのアップグレードが同時に実行されます。インスタンスは、アップグレードが完了するまで使用できません。

マルチ AZ 配置でオペレーティングシステムの更新が必要な場合は、データベースのアップグレードをリクエストした時点で、Amazon RDS によって更新が適用されます。Amazon RDS は以下の手順を実行します。

1. 現在のスタンバイ DB インスタンスのオペレーティングシステムを更新します。
2. プライマリ DB インスタンスをスタンバイ DB インスタンスにフェイルオーバーします。

3. 新しいプライマリ DB インスタンス (元のスタンバイインスタンス) のデータベースバージョンをアップグレードします。プライマリデータベースは、アップグレード中は利用できません。
4. 新しいスタンバイ DB インスタンス (元のプライマリインスタンス) のオペレーティングシステムをアップグレードします。
5. 新しいスタンバイ DB インスタンスのデータベースバージョンをアップグレードします。
6. 新しいプライマリ DB インスタンスを元のプライマリ DB インスタンスにフェイルオーバーし、新しいスタンバイ DB インスタンスを元のスタンバイ DB インスタンスにフェイルオーバーします。したがって、Amazon RDS はレプリケーション設定を元の状態に戻します。

リードレプリカでの Oracle のアップグレード

ソース DB インスタンスとそのすべてのリードレプリカの Oracle DB エンジンバージョンは同じである必要があります。Amazon RDS は、以下の段階を踏んでアップグレードを実行します。

1. ソース DB インスタンスをアップグレードします。リードレプリカはこの段階で使用できます。
2. レプリカのメンテナンスウィンドウに関係なく、リードレプリカを並行してアップグレードします。ソース DB はこの段階で使用できます。

クロスリージョンリードレプリカのメジャーバージョンアップグレードの場合、Amazon RDS によって追加のアクションが実行されます。

- ターゲットバージョンのオプショングループを自動的に生成します。
- 元のオプショングループから新しいオプショングループにすべてのオプションとオプション設定をコピーします。
- アップグレードされたクロスリージョンリードレプリカを新しいオプショングループに関連付けます。

アップグレード後の検証

アップグレードが完了したら、新しいバージョンを確認し、無効なオブジェクトがないことを確認します。

```
SELECT VERSION_FULL FROM V$INSTANCE;

SELECT COMP_NAME, VERSION, STATUS FROM DBA_REGISTRY;
```

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE FROM DBA_OBJECTS WHERE STATUS = 'INVALID' ORDER
 BY OWNER, OBJECT_TYPE;
```

直前のクエリで、所有している無効なオブジェクトが返された場合は、マスターユーザーとして以下の手順を実行し、スキーマ名を置き換えて再コンパイルします。

```
EXEC DBMS_UTILITY.COMPILE_SCHEMA(schema => 'schema_name');
```

DBA_REGISTRY のコンポーネントのステータスが VALID 以外である場合、またはアップグレード後に Oracle が提供するオブジェクト (SYS または SYSTEM 所有) が無効のままの場合は、AWS サポートにお問い合わせください。

Oracle のメジャーバージョンのアップグレード

メジャーバージョンアップグレードを実行するためには、DB インスタンスを手動で変更します。メジャーバージョンのアップグレードは自動的に実行されません。

> **Important**
>
> 本稼働データベースにアップグレードを適用する前に、アップグレードを徹底的にテストしてアプリケーションが正常に動作することを確認してください。詳細については、「Oracle DB アップグレードのテスト」を参照してください。

トピック

- メジャーアップグレードでサポートされているバージョン
- メジャーアップグレードでサポートされているインスタンスクラス
- メジャーアップグレード前の統計の収集
- メジャーアップグレードの許可
- Oracle Database 26ai へのアップグレード

メジャーアップグレードでサポートされているバージョン

Amazon RDS では、以下のメジャーバージョンへのアップグレードがサポートされています。

> **Note**
>
> Oracle Database 26ai は Enterprise Edition でのみ使用できます。

現在のバージョン	サポートされているアップグレード
CDB アーキテクチャを使用する 21.0.0.0	26.0.0.0
CDB アーキテクチャを使用する 19.0.0.0	21.0.0.0、26.0.0.0

> **Note**
>
> Oracle Database のメジャーバージョンアップグレードは、同じ月以降にリリースされた Release Update (RU) にアップグレードする必要があります。メジャーバージョンダウングレードは Oracle Database のいずれのバージョンでもサポートされていません。Amazon RDS でサポートされている Oracle Database のバージョンの詳細については、「Amazon RDS for Oracle リリースノート」を参照してください。

メジャーアップグレードでサポートされているインスタンスクラス

現在の Oracle DB インスタンスは、アップグレードするバージョンではサポートされていない DB インスタンスクラスで実行される可能性があります。この場合、アップグレードする前に、サポートされている DB インスタンスクラスに DB インスタンスを移行します。Amazon RDS for Oracle の各バージョンおよびエディションでサポートされている DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

メジャーアップグレード前の統計の収集

メジャーバージョンアップグレードを実行する前に、アップグレードする DB インスタンスのオプティマイザ統計を収集することをお勧めします。これにより、アップグレード中の DB インスタンスのダウンタイムを短縮できます。

オプティマイザ統計を収集するには、DB インスタンスにマスターユーザーとして接続し、次の例のように `DBMS_STATS.GATHER_DICTIONARY_STATS` の手順を実行します。

```
EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

詳細については、Oracle ドキュメントの「GATHER_DICTIONARY_STATS プロシージャ」を参照してください。

メジャーアップグレードの許可

エンジンのメジャーバージョンアップグレードは、アプリケーションと互換性がない場合があります。アップグレードは元に戻せません。EngineVersion パラメータに現在のメジャーバージョンとは異なるメジャーバージョンを指定する場合は、メジャーバージョンアップグレードを許可する必要があります。

CLI コマンド modify-db-instance を使用して、メジャーバージョンをアップグレードする場合、`--allow-major-version-upgrade` を指定します。この設定は永続的ではないため、メジャーアップグレードを実行するたびに `--allow-major-version-upgrade` を指定する必要があります。このパラメータは、エンジンのマイナーバージョンアップグレードには影響しません。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

コンソールを使用してメジャーバージョンをアップグレードする場合は、アップグレードを許可するオプションを選択する必要はありません。代わりに、メジャーアップグレードは元に戻せないという警告がコンソールに表示されます。

Oracle Database 26ai へのアップグレード

次のソースバージョンから Oracle Database 26ai (26.0.0.0) へのメジャーバージョンアップグレードを実行できます。

- CDB アーキテクチャを使用した Oracle Database 21c
- CDB アーキテクチャを使用した Oracle Database 19c

Oracle Database 26ai は CDB アーキテクチャのみをサポートします。DB インスタンスが非 CDB として Oracle Database 19c を実行している場合は、まず CDB に変換してから、Oracle Database 26ai にアップグレードします。アップグレード中にデータベースアーキテクチャを変更することはできません。詳細については、「RDS for Oracle の 非 CDB から CDB への変換」を参照してください。

非 CDB を CDB に変換した後、データベースはシングルテナント設定を使用します。テナントデータベース (PDB) を追加するには、データベースをシングルテナント設定からマルチテナント設定に

変換します。詳細については、「シングルテナント設定からマルチテナント設定への変換」を参照してください。

Oracle Database 26ai にアップグレードする前に、次の変更を確認してください。

- Oracle Database 26ai の新しいパラメータとサポートが廃止されたパラメータを確認します。詳細については、「Oracle Database 26ai (26.0.0.0) の Amazon RDS パラメータ変更」を参照してください。
- オプションに次の変更が加えられています。
  - Oracle は、26ai メジャーエンジンバージョンで TLS 1.2 以降のみをサポートします。この変更は、OEM エージェントと SSL オプション、およびそのオプション設定、特に許可された暗号スイートと TLS バージョンに影響します。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」および「Oracle Secure Sockets Layer」を参照してください。
  - Oracle Database 26ai は、Oracle APEX バージョン 24.1.v1 以降のみをサポートします。詳細については、「Oracle Application Express (APEX)」を参照してください。
  - NNE オプションでは、Amazon RDS は Oracle Database 26ai の古いハッシュ関数と暗号化アルゴリズムのサポートを廃止しました。詳細については、「Oracle ネイティブネットワーク暗号化」を参照してください。
- RDS for Oracle は、Oracle Database 26ai の OEM Database Express および OLAP オプションをサポートしていません。Oracle は、OEM Database Express のサポートを終了し、Oracle Database 26ai で OLAP を廃止しました。詳細については、「Oracle Enterprise Manager Database Express」および「Oracle OLAP」を参照してください。
- RDS for Oracle は、Oracle Database 26ai の Locator オプションをサポートしていません。Oracle Spatial は Oracle Locator に取って代わるものであり、Locator の機能を包含しています。Oracle Database 26ai で Locator 機能を使用するには、代わりに Oracle Spatial オプションを使用します。詳細については、「Oracle Locator」および「Oracle Spatial」を参照してください。

Oracle のマイナーバージョンのアップグレード

RDS for Oracle では、マイナーバージョンアップグレードは、メジャー DB エンジンバージョンの更新です。RDS では、マイナーエンジンバージョンは、リリース更新 (RU)、Supplemental パッチバンドル (SPB)、または Oracle Database 26ai の場合は月次推奨パッチ (MRP) と RU の組み合わせです。例えば、DB インスタンスでメジャーバージョン Oracle Database 19c とマイナーバージョン 19.0.0.0.ru-2025-10.rur-2025-10.r1 を実行している場合、DB エンジンはマイナーバージョン

19.0.0.0.ru-2026-01.rur-2026-01.r1 にアップグレードできます。RDS for Oracle は、マイナーバージョンのダウングレードをサポートしません。

DB エンジンは、手動または自動でマイナーバージョンにアップグレードできます。手動でアップグレードする方法については、「エンジンバージョンの手動アップグレード」を参照してください。自動アップグレードの設定方法については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。手動でアップグレードするか自動アップグレードするかにかかわらず、マイナーバージョンのアップグレードにはダウンタイムが伴います。アップグレードを計画するときは、このダウンタイムを考慮してください。

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

> **Important**
>
> 本稼働データベースにアップグレードを適用する前に、アップグレードを徹底的にテストしてアプリケーションが正常に動作することを確認してください。詳細については、「Oracle DB アップグレードのテスト」を参照してください。

トピック

- リリースアップデート (RU) と Supplemental パッチバンドル (SPB)
- リリース更新と月次推奨パッチ
- Oracle 用自動マイナーバージョンアップグレードの実行
- 自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用
- Amazon RDS for Oracle でのマイナーバージョン自動アップグレードの通知
- Amazon RDS がマイナーバージョンの自動アップグレードをスケジュールする方法
- RDS for Oracle でのマイナーバージョン自動アップグレードの管理

リリースアップデート (RU) と Supplemental パッチバンドル (SPB)

RDS では、リリースアップデート (RU) は、Oracle Database のセキュリティ修正、バグ修正、新機能を含む四半期ごとのマイナーエンジンバージョンです。Supplemental パッチバン

ドル (SPB) は、Oracle Spatial、Oracle Data Pump、Oracle GoldenGate などの特定のユースケースで Oracle が推奨する追加のデータベースパッチを含む RU エンジンバージョンです。例えば、19.0.0.0.ru-2026-04.spb-1.r1 という名前の SPB には、対応する RU のすべてのパッチ 19.0.0.0.ru-2026-04.rur-2026-04.r1 と Supplemental パッチが含まれています。SPB は Oracle Database 19c でのみサポートされています。

インスタンスが自動マイナーバージョンアップグレード用に設定されている場合、RU と SPB は別々のアップグレードパスにあります。通常、SPB は対応する RU の 2～3 週間後にリリースされます。次の表は、Oracle Database 19c のマイナーバージョンの例を示しています。

標準 RU アップグレードパス	SPB アップグレードパス
19.0.0.0.ru-2025-01.rur-2025-01.r1	19.0.0.0.ru-2025-01.spb-1.r1
19.0.0.0.ru-2025-04.rur-2025-04.r1	19.0.0.0.ru-2025-04.spb-1.r1
19.0.0.0.ru-2025-07.rur-2025-07.r1	19.0.0.0.ru-2025-07.spb-1.r1
19.0.0.0.ru-2025-10.rur-2025-10.r1	19.0.0.0.ru-2025-10.spb-1.r1

DB インスタンスが自動アップグレード用に設定されている場合、インスタンスは現在のバージョンに対応するアップグレードパスにあります。例えば、DB インスタンスがバージョン 19.0.0.0.ru-2026-04.rur-2026-04.r1 を実行している場合、19.0.0.0.ru-2026-07.rur-2026-07.r1 がリリースされると、インスタンスは自動的にこの RU にアップグレードされます。同様に、DB インスタンスが 19.0.0.0.ru-2026-04.spb-1.r1 を実行している場合、19.0.0.0.ru-2026-07.spb-1.r1 がリリースされると、インスタンスは自動的にこの SPB にアップグレードされます。RU である 19.0.0.0.ru-2026-04.rur-2026-04.r1 を実行しているインスタンスは、別のアップグレードパスの SPB である 19.0.0.0.ru-2026-07.spb-1.r1 に自動的にアップグレードされません。

インスタンスに Oracle Spatial、Oracle Data Pump、Oracle GoldenGate などのユースケースが現在ない場合でも、DB インスタンスを SPB にアップグレードできます。RU から同じエンジンバージョン以上の SPB に手動でアップグレードできます。例えば、インスタンスを 19.0.0.0.ru-2026-04.rur-2026-04.r1 から次のいずれかのエンジンバージョンにアップグレードできます。

- 19.0.0.0.ru-2026-04.spb-1.r1
- 19.0.0.0.ru-2026-07.spb-1.r1

RU が上位のエンジンバージョンである場合にのみ、インスタンスを SPB から RU にアップグレードできます。例えば、SPB バージョン 19.0.0.0.ru-2026-04.spb-1.r1 から上位の RU バージョン 19.0.0.0.ru-2026-07.rur-2026-07.r1 にアップグレードできますが、同じ RU バージョン 19.0.0.0.ru-2026-04.rur-2026-04.r1 にアップグレードすることはできません。

DB インスタンスが自動マイナーバージョンアップグレード用に設定されていて、RU から SPB または SPB から RU に手動でアップグレードすると、自動アップグレードパスが変更されます。RU バージョン 19.0.0.0.ru-2026-04.rur-2026-04.r1 から SPB バージョン 19.0.0.0.ru-2026-04.spb-1.r1 に手動でアップグレードするとします。次回の自動マイナーバージョンアップグレードは SPB バージョン 19.0.0.0.ru-2026-07.spb-1.r1 になります。

SPB は RU として機能するため、インスタンスを RU と SPB にアップグレードするための RDS API は同じです。次のコマンドは、RU と SPB へのアップグレードを示しています。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version 19.0.0.0.ru-2026-04.rur-2026-04.r1

aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --engine-version 19.0.0.0.ru-2026-04.spb-1.r1
```

SPB が Oracle Spatial とどのように連携するかについての詳細は、「Supplemental パッチバンドル (SPB) の仕組み」を参照してください。Oracle Database 19c でサポートされている RU と SPB については、「Amazon RDS for Oracle Database 19c (19.0.0.0)」を参照してください。Oracle Database 26ai でサポートされている RU、MRP、SPB については、「Amazon RDS for Oracle Database 26ai (26.0.0.0)」を参照してください。

> **Note**
>
> Spatial パッチバンドルの名前が Supplemental パッチバンドル (SPB) に変更されました。SPB には、Data Pump や GoldenGate など、Oracle Spatial 以外のバンドルパッチが追加されました。省略形「SPB」は変更されません。既存の SPB エンジンバージョンはすべて、以前と同じように動作し続けます。

リリース更新と月次推奨パッチ

RDS for Oracle は、Oracle Database 26ai (26.0.0.0) の新しいマイナーエンジンバージョンを四半期ごとにリリースします。各 Oracle Database 26ai エンジンバージョンには、RU コンポーネントと

MRP コンポーネントが含まれています。RU には、Oracle が指定した四半期にリリースするパッチが含まれます。MRP には、Critical Security Patch Update と Oracle が推奨するその他の修正が含まれています。

Oracle Database 26ai エンジンバージョンは、*release*.ru-*ru-date*.mrp-*mrp-date*.r*rnumber* の形式を使用します。この形式には以下のコンポーネントが含まれます。

####

メジャーリリース番号。

ru-date

RU の日付。

mrp-date

MRP の日付。

rnumber

リビジョン番号。

例えば、2026 年 1 月の四半期 RU の名前は 26.0.0.0.ru-2026-01.mrp-2026-01.r1 です。

Oracle は、四半期ごとの RU の間の月に MRP をリリースすることがあります。新しい MRP はエンジンバージョンの MRP コンポーネントを強化しますが、RU コンポーネントは変わりません。例えば、26.0.0.0.ru-2026-01.mrp-2026-03.r1 には 2026 年 1 月の RU と 2026 年 3 月の MRP が含まれています。

Oracle は MRP を毎月リリースするわけではありません。1 か月間に RU に対する新しい推奨修正や緊急セキュリティパッチの更新がない場合、Oracle はその月に MRP をリリースしません。

RDS for Oracle は、最新の RU に対してのみ MRP をサポートします。以前の RU の最新のセキュリティ修正を取得するには、DB インスタンスを最新の RU にアップグレードします。

上位の Oracle Database 26ai エンジンバージョンにアップグレードできます。アップグレードは、新しい四半期ごとの RU、新しい MRP、またはその両方にすることができます。RDS for Oracle は、マイナーバージョンのダウングレードをサポートしません。Oracle Database 26ai でサポートされている RU と MRP の詳細については、「Amazon RDS for Oracle Database 26ai (26.0.0.0)」を参照してください。

Oracle 用自動マイナーバージョンアップグレードの実行

> **MRP の手動アップグレード**
>
> Oracle Database 26ai の場合、マイナーバージョンの自動アップグレードでは新しい四半期ごとの RU が適用されますが、MRP は適用されません。DB インスタンスを新しい MRP にアップグレードするには、アップグレードを手動で適用します。

マイナーバージョンの自動アップグレードでは、RDS は手動操作なしで、使用可能な最新のマイナーバージョンを Oracle データベースに適用します。Amazon RDS for Oracle DB インスタンスは、次の状況で、次のメンテナンス期間にアップグレードをスケジュールします。

- DB クラスターの [自動マイナーバージョンアップグレード] オプションが有効になっています。
- DB インスタンスで最新のマイナー DB エンジンバージョンが実行されていません。

自動アップグレードを有効にする方法については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。

自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用

Amazon RDS for Oracle は、複数のデータベースリソースと AWS アカウントにわたってマイナーバージョンの自動アップグレードを管理するための AWS Organizations アップグレードロールアウトポリシーをサポートしています。このポリシーにより、手動またはカスタムツールを使用してマイナーバージョンの自動アップグレードを調整する運用上のオーバーヘッドがなくなり、アップグレードは運用環境にロールアウトされる前に本番環境以外の環境で最初に適用されるようになります。新しいマイナーエンジンバージョンが利用可能になると、Amazon RDS は、設定されたアップグレードのロールアウト順序に基づいて DB インスタンスをアップグレードします。

ロールアウト順序のアップグレード	一般的なユースケース	アップグレードを開始するタイミング
First	開発環境とテスト環境	最も早い時間 - 新しいバージョンの検証に最適

ロールアウト順序のアップグレード	一般的なユースケース	アップグレードを開始するタイミング
秒	ステージング環境と重要度の低い運用環境	「最初」のフェーズが完了した後
Last	重要な運用環境	「2 番目」のフェーズが完了した後

> **Important**
>
> DB インスタンスのアップグレードロールアウト順序を設定しない場合、デフォルトで 2 番目に設定されます。

フェーズのタイミングと期間の詳細については、「Amazon RDS がマイナーバージョンの自動アップグレードをスケジュールする方法」を参照してください。AWS Organizations でのアップグレードロールアウトポリシーの設定については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

Amazon RDS for Oracle でのマイナーバージョン自動アップグレードの通知

DB インスタンスでマイナーバージョンの自動アップグレードが有効になっている場合、RDS for Oracle はアップグレードを適用する前に通知する保留中のメンテナンスアクションを作成します。これらの保留中のメンテナンスアクションは、Amazon RDS コンソールのデータベースの詳細ページの [メンテナンスとバックアップ] タブで表示できます。

新しいマイナーバージョンが利用可能になると、RDS for Oracle は早期通知 (メンテナンスアクションの保留) を発行します。この早期通知の形式は次のとおりです。

```
An automatic minor version upgrade to engine-version will be applied during your
 maintenance window on apply-date based on the upgrade rollout order rollout-order. You
 can change the upgrade rollout order or apply this upgrade manually at any time before
 the scheduled date through the AWS console or AWS CLI.
```

早期通知の `apply-date` は、Amazon RDS が DB インスタンスをアップグレードする日付です。`rollout-order` はアップグレードのロールアウト順序 (1 番目、2 番目、または最後) です。アップグレードロールアウトポリシーを設定していない場合、この値はデフォルトで 2 番目です。

詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

アップグレードのロールアウトフェーズが開始されると、保留中のメンテナンスアクションメッセージは次の形式に変わります。

```
Automatic minor version upgrade to engine-version
```

このメッセージは、アップグレードがスケジュールされており、スケジュールされた適用日のメンテナンス期間中に適用されることを示します。スケジュールされた適用日は、Amazon RDS コンソールのデータベースの詳細ページの [メンテナンスとバックアップ] タブ、または describe-pending-maintenance-actions API レスポンスの CurrentApplyDate フィールドで確認できます。

次の例は、AWS CLI の describe-pending-maintenance-actions コマンドを使用して、保留中のメンテナンスアクションの詳細を取得できることを示しています。

```
aws rds describe-pending-maintenance-actions

    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:orclinst1",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "db-upgrade",
                    "Description": "Automatic minor version upgrade to
 21.0.0.0.ru-2024-07.rur-2024-07.r1",
                    "CurrentApplyDate": "2024-12-02T08:10:00Z"
                }
            ]
        }, ...
```

describe-pending-maintenance-actions の詳細については、「AWS CLI コマンドリファレンス」を参照してください。

Amazon RDS がマイナーバージョンの自動アップグレードをスケジュールする方法

AWS Organizations のアップグレードロールアウトポリシーを使用する場合、Amazon RDS は設定されたロールアウト順序に基づいたフェーズで DB インスタンスをアップグレードします。このセクションでは、各フェーズのタイミングと期間について説明します。

フェーズ 0: 早期通知

RDS for Oracle が新しいマイナーバージョンをリリースすると (通常は Oracle の四半期 RU リリースから 3～4 週間後)、自動マイナーバージョンアップグレードが有効になっているすべての DB インスタンスは早期通知を受信します。この通知は、Amazon RDS コンソールのデータベースの詳細ページの [メンテナンスとバックアップ] タブと `describe-pending-maintenance-actions` API レスポンスに表示されます。早期通知フェーズは 2 週間続きます。このフェーズの期間中、自動アップグレードは行われません。

フェーズ 1: 最初のアップグレードロールアウト順序

早期通知フェーズの終了時に、RDS for Oracle はアップグレードロールアウト順序が最初の DB インスタンスのアップグレードを開始します。このフェーズは、1 月、4 月、7 月の四半期のマイナーバージョンでは 2～3 週間、10 月の四半期のマイナーバージョンでは 7～8 週間続きます。10 月のマイナーバージョンの延長期間は、年末の祝祭日シーズン中に新しいマイナーバージョンをテストするのに十分な時間を提供します。このフェーズ中に最初のアップグレードのロールアウト順序で作成された新しい DB インスタンスは、自動的にアップグレードされます。

フェーズ 2: 2 番目のアップグレードロールアウト順序

フェーズ 1 の終了時に、RDS for Oracle はアップグレードロールアウト順序が 2 番目の DB インスタンスのアップグレードを開始します。このフェーズは、すべての四半期のマイナーバージョンで 2 週間続きます。このフェーズ中に最初のまたは 2 番目のアップグレードロールアウト順序で作成された新しい DB インスタンスは、自動的にアップグレードされます。

フェーズ 3: 最後のアップグレードロールアウト順序

フェーズ 2 の終了時に、RDS for Oracle はアップグレードロールアウト順序が最後の DB インスタンスのアップグレードを開始します。このフェーズは、次の四半期のマイナーバージョンリリースまで続きます。このフェーズ中に最初、2 番目、または最後のアップグレードロールアウト順序で作成された新しい DB インスタンスは、自動的にアップグレードされます。

[Phase]	開始するタイミング	時間	保留中のメンテナンスアクションメッセージ
フェーズ 0: 早期通知	RDS for Oracle が新しいマイナーバージョンをリリースしたとき	2 週間	######### へのマイナーバージョンの

[Phase]	開始するタイミング	時間	保留中のメンテナンスアクションメッセージ
			自動アップグレードは、アップグレードのロールアウト順序の *rollout-order* に基づき、*apply-date* にメンテナンスウィンドウ中に適用されます。AWS コンソールまたは AWS CLI を使用して、アップグレードのロールアウト順序を変更したり、スケジュールされた日より前にいつでもこのアップグレードを手動で適用したりできます。

[Phase]	開始するタイミング	時間	保留中のメンテナンスアクションメッセージ
フェーズ 1: 最初のアップグレードロールアウト順序	早期通知フェーズの終了	1 月/4 月/7 月のマイナーバージョンでは 2 ~ 4 週間、10 月のマイナーバージョンでは 7 ~ 9 週間	*engine-version* への自動マイナーバージョンアップグレード
フェーズ 2: 2 番目のアップグレードロールアウト順序	フェーズ 1 の終了	2 週間	*engine-version* への自動マイナーバージョンアップグレード
フェーズ 3: 最後のアップグレードロールアウト順序	フェーズ 2 の終了	次の四半期ごとのマイナーバージョンリリースまで	*engine-version* への自動マイナーバージョンアップグレード

RDS for Oracle でのマイナーバージョン自動アップグレードの管理

DB インスタンスでマイナーバージョンの自動アップグレードが有効になっている場合、Amazon RDS はメンテナンスウィンドウ中に DB インスタンスを最新のマイナーバージョンに自動的にアップグレードします。ただし、AWS CLI を使用するか、データベースの詳細ページの [メンテナンスとバックアップ] タブから、スケジュール日より前に手動でアップグレードを適用できます。

スケジュールされたメンテナンスウィンドウを待たずに DB インスタンスをすぐにアップグレードするには

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:orclinst1 \
    --apply-action db-upgrade \
    --opt-in-type immediate
```

スケジュールされた適用日ではなく、次のメンテナンスウィンドウ中にアップグレードを適用するには

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:orclinst1 \
    --apply-action db-upgrade \
    --opt-in-type next-maintenance
```

自動マイナーバージョンアップグレードをオプトアウトするには、DB インスタンスを変更し、自動マイナーバージョンアップグレードオプションをオフにします。これにより、保留中の自動アップグレードのスケジュールが解除されます。

マイナーバージョンの自動アップグレードをオフにする方法の詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。マイナーバージョンの自動アップグレードの無効化についてサポートが必要な場合は、AWS サポートにお問い合わせください。

RDS が以前のマイナーバージョンを適用する前に、新しいマイナーバージョンが利用可能になる場合があります。例えば、both `21.0.0.0.ru-2025-10.rur-2025-10.r1` および `21.0.0.0.ru-2026-01.rur-2026-01.r1` がアップグレードターゲットとして利用可能な場合、インスタンスは `21.0.0.0.ru-2025-07.rur-2025-07.r1` で実行されています。この場合、DB インスタンスの不要なダウンタイムを回避するために、RDS はマイナーバージョンの自動アップグレードを最新バージョンにスケジュールし、以前のバージョンへのアップグレードをスキップします。この例では、RDS はインスタンスを `21.0.0.0.ru-2025-07.rur-2025-07.r1` から `21.0.0.0.ru-2026-01.rur-2026-01.r1` に直接アップグレードします。

Oracle のデータベースアップグレードに関する考慮事項

Oracle インスタンスをアップグレードする前に、次の情報を確認してください。

トピック

- Oracle マルチテナントに関する考慮事項
- オプショングループに関する考慮事項
- パラメータグループに関する考慮事項
- タイムゾーンに関する考慮事項
- Supplemental パッチバンドル (SPB) に関する考慮事項

Oracle マルチテナントに関する考慮事項

次の表では、さまざまなリリースでサポートされる Oracle Database アーキテクチャについて説明します。

Oracle Database のリリース	RDS のサポートステータス	アーキテクチャ
Oracle Database 26ai	サポート対象	CDB のみ
Oracle Database 21c	サポート対象	CDB のみ
Oracle Database 19c	サポート対象	CDB または非 CDB

次の表に、サポートされているアップグレードパスとサポートされていないアップグレードパスを示します。

アップグレードパス	サポート対象?
CDB から CDB	はい
非 CDB から CDB	いいえ。ただし、非 CDB を CDB に変換してからアップグレードできます。
CDB から非 CDB	いいえ

Oracle で RDS を使用するOracle のマルチテナントに関する詳細は、「CDB アーキテクチャのシングルテナント設定」を参照してください。

オプショングループに関する考慮事項

DB インスタンスがカスタムオプショングループを使用している場合、Amazon RDS が新しいオプショングループを自動的に割り当てられないことがあります。例えば、この状況は、新しいメジャーバージョンにアップグレードするときに発生します。このような場合、アップグレード時に新しいオプショングループを指定します。新しいオプショングループを作成し、このオプショングループに既存のカスタムオプショングループと同じオプションを追加することをお勧めします。

詳細については、「オプショングループを作成する」または「オプショングループをコピーする」を参照してください。

DB インスタンスで使用するカスタムオプショングループに APEX オプションと APEX-DEV オプションが含まれている場合、アップグレード時間を短縮できる可能性があります。そのためには、使用しているバージョンの APEX を DB インスタンスと同時にアップグレードします。詳細については、「Oracle APEX バージョンのアップグレード」を参照してください。

パラメータグループに関する考慮事項

DB インスタンスでカスタムパラメータグループを使用している場合、Amazon RDS で DB インスタンスに新しいパラメータグループを自動的に割り当てられないことがあります。例えば、この状況は、新しいメジャーバージョンにアップグレードするときに発生します。このような場合、アップグレード時に必ず新しいパラメータグループを指定する必要があります。新しいパラメータグループを作成し、そのパラメータの設定を既存のカスタムパラメータグループと同じにすることをお勧めします。

詳細については、「Amazon RDS での DB パラメータグループの作成」または「Amazon RDS での DB パラメータグループのコピー」を参照してください。

タイムゾーンに関する考慮事項

タイムゾーンオプションを使用して、Oracle DB インスタンスで使用するシステムのタイムゾーンを変更することができます。例えば、オンプレミス環境またはレガシーアプリケーションとの互換性があるように、DB インスタンスのタイムゾーンで変更が必要になることがあります。タイムゾーンオプションでは、ホストレベルでタイムゾーンが変更されます。Amazon RDS for Oracle では、システムタイムゾーンは年間を通して自動的に更新されます。システムのタイムゾーンの詳細については、「Oracle のタイムゾーン」を参照してください。

Oracle DB インスタンスを作成すると、データベースによって データベースのタイムゾーンが自動的に設定されます。データベースのタイムゾーンは、夏時間 (DST) タイムゾーンとも呼ばれます。データベースのタイムゾーンは、システムのタイムゾーンとは異なります。

Oracle Database の各リリース間には、パッチセットまたは個々のパッチに、新しい DST バージョンが含まれる場合があります。これらのパッチは、さまざまなタイムゾーンリージョンの移行ルールの変更を反映しています。例えば、DST が有効になると、政府機関が変わる場合があります。DST ルールを変更すると、`TIMESTAMP WITH TIME ZONE` データ型の既存のデータに影響する場合があります。

RDS for Oracle DB インスタンスをアップグレードする場合、Amazon RDS はデータベースのタイムゾーンファイルを自動的にアップグレードしません。タイムゾーンファイルを自動的にアップグレードするには、エンジンバージョンのアップグレード中またはアップグレード後に、DB インスタ

ンスに関連付けられたオプショングループに TIMEZONE_FILE_AUTOUPGRADE オプションを追加します。詳しくは、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。

データベースのタイムゾーンファイルを手動でアップグレードするには、必要な DST パッチを持つ新しい Oracle DB インスタンスを作成します。ただし、TIMEZONE_FILE_AUTOUPGRADE オプションを使用して、データベースのタイムゾーンファイルをアップグレードすることをお勧めします。

タイムゾーンファイルのアップグレード後、現在のインスタンスから新しいインスタンスにデータを移行します。データを移行するには、以下を含む複数の手法を使用できます。

- AWS Database Migration Service
- Oracle GoldenGate
- Oracle Data Pump
- 元のエクスポート/インポート (一般的な使用に対してはサポート終了)

> **Note**
>
> Oracle Data Pump を使用してデータを移行すると、ターゲットのタイムゾーンバージョンがソースのタイムゾーンバージョンよりも古い場合、エラー ORA-39405 が発生します。

詳細については、Oracle ドキュメントの「TIMESTAMP WITH TIMEZONE Restrictions」を参照してください。

Supplemental パッチバンドル (SPB) に関する考慮事項

RDS for Oracle では、リリースアップデート (RU) は、Oracle Database のセキュリティ修正、バグ修正、新機能を含むマイナーエンジンバージョンです。Supplemental パッチバンドル (SPB) は、Oracle Spatial、Oracle Data Pump、Oracle GoldenGate などの特定のユースケースで Oracle が推奨する追加のデータベースパッチを含む RU エンジンバージョンです。例えば、19.0.0.0.ru-2025-01.spb-1.r1 は、エンジンバージョン 19.0.0.0.ru-2025-01.rur-2025-01.r1 の RU パッチと Supplemental パッチを含むマイナーエンジンバージョンです。

データベースを SPB にアップグレードする場合は、以下を考慮してください。

- SPB は Oracle Database 19c でのみサポートされています。
- DB インスタンスの自動マイナーバージョンアップグレードを有効にする場合、アップグレードパスは、インスタンスが現在 SPB または RU のどちらを使用しているかによって異なります。イ

ンスタンスが SPB を使用している場合、RDS はインスタンスを最新の SPB に自動的にアップグレードします。インスタンスが RU を使用している場合、RDS はインスタンスを最新の RU に自動的にアップグレードします。

- SPB が現在の RU と同じエンジンバージョン以上である場合にのみ、DB インスタンスを RU から SPB に手動でアップグレードできます。
- RU が上位バージョンである場合にのみ、DB インスタンスを SPB から RU に手動でアップグレードできます。

Oracle DB アップグレードのテスト

DB インスタンスをメジャーバージョンにアップグレードする前に、データベースとデータベースにアクセスするすべてのアプリケーションについて、新しいバージョンとの互換性を綿密にテストする必要があります。以下の手順を実行することをお勧めします。

メジャーバージョンのアップグレードをテストするには

1. データベースエンジンの新しいバージョンについて Oracle アップグレードドキュメントを参照して、データベースやアプリケーションに影響を与える可能性のある互換性の問題があるかどうかを確認します。詳細については、Oracle ドキュメントの「Database Upgrade Guide」を参照してください。
2. DB インスタンスでカスタムオプショングループを使用している場合は、アップグレード先の新しいバージョンと互換性がある新しいオプショングループを作成します。詳細については、「オプショングループに関する考慮事項」を参照してください。
3. DB インスタンスでカスタムパラメータグループを使用している場合は、アップグレード先の新しいバージョンと互換性がある新しいパラメータグループを作成します。詳細については、「パラメータグループに関する考慮事項」を参照してください。
4. アップグレードする DB インスタンスの DB スナップショットを作成します。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。
5. DB スナップショットを復元して、新しいテスト DB インスタンスを作成します。詳細については、「DB インスタンスへの復元」を参照してください。
6. この新しいテスト DB インスタンスを変更して新しいバージョンにアップグレードするには、次に説明するいずれかの方法を使用します。

    - コンソール
    - AWS CLI

- RDS API

7. テストを実行します。

   - データベースとアプリケーションが新しいバージョンで正常に動作することが確認されるまで、アップグレードした DB インスタンスに対する品質保証テストを必要な回数だけ実行します。
   - 手順 1 で特定した互換性の問題の影響を評価するための新しいテストを実行します。
   - すべてのストアドプロシージャ、関数、トリガーをテストします。
   - アプリケーションのテストバージョンを、アップグレードした DB インスタンスに割り振ります。アプリケーションが新しいバージョンで正しく動作することを確認します。
   - アップグレードしたインスタンスによって使用されるストレージを評価して、アップグレードに追加のストレージが必要かどうかを判断します。本稼働で新しいバージョンをサポートするために、より大きなインスタンスのクラスを選択する必要がある場合もあります。詳細については、「DB インスタンスクラス」を参照してください。

8. すべてのテストに合格したら、本番稼働用 DB インスタンスをアップグレードします。DB インスタンスへの書き込みオペレーションを許可する前に、DB インスタンスが正しく機能していることを確認することをお勧めします。

RDS for Oracle DB インスタンスバージョンのアップグレード

RDS for Oracle DB インスタンスの DB エンジンバージョンを手動でアップグレードするには、AWS マネジメントコンソール、AWS CLI または RDS API を使用します。データベースのアップグレードに関する一般的な情報については、「RDS for Oracle DB インスタンスバージョンのアップグレード」を参照してください。有効なアップグレードターゲットを取得するには、AWS CLIdescribe-db-engine-versions コマンドを使用します。

コンソール

コンソールを使用して RDS for Oracle DB インスタンスのエンジンバージョンをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、アップグレードする DB インスタンスを選択します。

3. [変更] を選択します。
4. [DB エンジンバージョン] で、上位のデータベースバージョンを選択します。
5. [続行] を選択して、変更の概要を確認します。データベースバージョンアップグレードの影響を理解していることを確認します。アップグレードした DB インスタンスを以前のバージョンに戻すことはできません。続行する前に、データベースとアプリケーションの両方を新しいバージョンでテストしていることを確認します。
6. DB インスタンスのアップグレードをスケジュールするタイミングを決定します。変更をすぐに反映させるには、[Apply immediately] を選択します。このオプションを選択すると、停止状態になる場合があります。詳細については、「スケジュール変更設定の使用」を参照してください。
7. 確認ページで、変更内容を確認します。正しい場合は、[Modify DB Instance (DB インスタンスを変更)] を選択して変更を保存します。

   または、[Back] を選択して変更を編集するか、[Cancel] を選択して変更をキャンセルします。

AWS CLI

RDS for Oracle DB インスタンスのエンジンバージョンをアップグレードするには、CLI の modify-db-instance コマンドを使用します。以下のパラメータを指定します。

- `--db-instance-identifier` – RDS for Oracle DB インスタンスの名前。
- `--engine-version` - アップグレード先のデータベースエンジンのバージョン番号です。

  有効なエンジンバージョンの詳細については、AWS CLI の describe-db-engine-versions コマンドを参照してください。
- `--allow-major-version-upgrade` – DB エンジンバージョンのアップグレード
- `--no-apply-immediately` – 次のメンテナンス時間中に変更を適用します。今すぐ変更を適用するには、`--apply-immediately` を使用します。

Example

次の例は、`myorainst` という名前の CDB インスタンスを現在のバージョン `19.0.0.0.ru-2024-01.rur-2024-01.r1` からバージョン `21.0.0.0.ru-2024-04.rur-2024-04.r1` にアップグレードします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
```

```
    --db-instance-identifier myorainst \
    --engine-version 21.0.0.0.ru-2024-04.rur-2024-04.r1 \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier myorainst ^
    --engine-version 21.0.0.0.ru-2024-04.rur-2024-04.r1 ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

RDS API

RDS for Oracle DB インスタンスをアップグレードするには、ModifyDBInstance アクションを使用します。以下のパラメータを指定します。

- `DBInstanceIdentifier` – DB インスタンスの名前、例えば *myorainst* です。
- `EngineVersion` – アップグレード先のデータベースエンジンのバージョン番号です。有効なエンジンバージョンについては、DescribeDBEngineVersions オペレーションを使用します。
- `AllowMajorVersionUpgrade` – メジャーバージョンのアップグレードを許可するかどうか。そのためには、値を `true` に設定します。
- `ApplyImmediately` – 変更をすぐに適用するか、次のメンテナンスウィンドウ中に適用するかを指定します。今すぐ変更を適用するには、値を `true` に設定します。次のメンテナンスウィンドウ中に変更を適用するには、値を `false` に設定します。

Oracle DB スナップショットのアップグレード

Amazon RDS で Oracle DB スナップショットをアップグレードすると、データベースの安全性、互換性、および完全なサポートが確保されます。古い Oracle バージョンのパッチサポートが終了した場合、これらのバージョンに関連付けられた手動 DB スナップショットをアップグレードして、潜在的な脆弱性やサービスの制限を回避できます。詳細については、「Oracle エンジンのバージョン管理」を参照してください。

Amazon RDS はすべての AWS リージョン のスナップショットのアップグレードに対応しています。

コンソール

Oracle DB スナップショットのアップグレード方法

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[スナップショット] を選択し、アップグレードする DB スナップショットを選択します。
3. [アクション] は、[スナップショットのアップグレード] を選択します。[スナップショットのアップグレード] ページが表示されます。
4. [New engine version (新しいエンジンバージョン)] で、スナップショットをアップグレードするバージョンを選択します。
5. (オプション) [Option group] には、アップグレードした DB スナップショットのオプショングループを選択します。DB インスタンスをアップグレードするときと同じように DB スナップショットをアップグレードする場合も、同じオプショングループの考慮事項が適用されます。詳細については、「オプショングループに関する考慮事項」を参照してください。
6. [Save changes (変更の保存)] を選択して、変更を保存します。

   アップグレード中は、この DB スナップショットに関するスナップショット操作が、すべて無効になります。また、DB スナップショットのステータスは利用可能からアップグレード中に変わり、プロセスが完了すると有効になります。スナップショットの破損問題で DB スナップショットをアップグレードできない場合、ステータスは使用不可になります。この状態からスナップショットを復元することはできません。

   > **Note**
   >
   > DB スナップショットアップグレードが失敗した場合、スナップショットは元の状態にロールバックします。

AWS CLI

AWS CLI を使用して Oracle DB スナップショットをアップグレードするには、以下のパラメータを指定して modify-db-snapshot コマンドを呼び出します。

- `--db-snapshot-identifier` – DB スナップショットの名前です。
- `--engine-version` – スナップショットをアップグレードするバージョンです。

必要に応じて、以下のパラメータを含める場合もあります。DB インスタンスをアップグレードするときと同じように DB スナップショットをアップグレードする場合も、同じオプショングループの考慮事項が適用されます。詳細については、「オプショングループに関する考慮事項」を参照してください。

- `--option-group-name` – アップグレードした DB スナップショットのオプショングループ。

Example

以下の例では、DB スナップショットをアップグレードします。

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier mydbsnapshot \
    --engine-version 19.0.0.0.ru-2020-10.rur-2020-10.r1 \
    --option-group-name default:oracle-se2-19
```

Windows の場合:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier mydbsnapshot ^
    --engine-version 19.0.0.0.ru-2020-10.rur-2020-10.r1 ^
    --option-group-name default:oracle-se2-19
```

RDS API

Amazon RDS API を使用して Oracle DB スナップショットをアップグレードするには、以下のパラメータを指定して ModifyDBSnapshot オペレーションを呼び出します。

- `DBSnapshotIdentifier` – DB スナップショットの名前です。
- `EngineVersion` – スナップショットをアップグレードするバージョンです。

また、`OptionGroupName` パラメータを含めなければならない場合もあります。DB インスタンスをアップグレードするときと同じように DB スナップショットをアップグレードする場合も、同じオプショングループの考慮事項が適用されます。詳細については、「オプショングループに関する考慮事項」を参照してください。

RDS for Oracle DB インスタンスでのサードパーティーソフトウェアの使用

ツールとサードパーティーソフトウェアをサポートする RDS for Oracle DB インスタンスをホストできます。

トピック

- Oracle GoldenGate と Amazon RDS for Oracle の使用
- Amazon RDS for Oracle での Oracle リポジトリ作成ユーティリティの使用
- Amazon EC2 インスタンスでの Oracle Connection Manager の設定
- Amazon RDS での Oracle への Siebel Database のインストール

Oracle GoldenGate と Amazon RDS for Oracle の使用

Oracle GoldenGate は、データベース間でトランザクションデータを収集、レプリケート、管理します。このツールは、ログベースの変更データキャプチャ (CDC) であり、オンライントランザクション処理 (OLTP) システムのデータベースで使用されるレプリケーションソフトウェアパッケージです。Oracle GoldenGate によって、ソースデータベースで最近変更されたデータを含んでいるトレイルファイルが作成されます。次に、これらのファイルをサーバーにプッシュします。サーバーでは、プロセスがトレイルファイルを標準 SQL に変換してターゲットデータベースに適用します。

Oracle GoldenGate と RDS for Oracle では、次の機能をサポートしています。

- アクティブ-アクティブデータベースレプリケーション
- 災害対策
- データ保護
- リージョン内およびクロスリージョンレプリケーション
- ダウンタイムのない移行とアップグレード
- RDS for Oracle DB インスタンスと Oracle 以外のデータベース間のデータレプリケーション

> **Note**
>
> サポートされているデータベースの一覧については、Oracle ドキュメントの「Oracle Fusion Middleware Supported System Configurations」(Oracle Fusion ミドルウェアでサポートされているシステム設定) を参照してください。

RDS for Oracle で Oracle GoldenGate を使用して、Oracle Database のメジャーバージョンにアップグレードすることができます。例えば、Amazon RDS DB インスタンスで Oracle GoldenGate を使用して、Oracle Database 11g オンプレミスデータベースから Oracle Database 19c にアップグレードすることができます。

トピック

- Oracle GoldenGate でサポートされているバージョンとライセンスオプション
- Oracle GoldenGate の要件と制限
- Oracle GoldenGate アーキテクチャ
- Oracle GoldenGate のセットアップ
- Oracle GoldenGate の EXTRACT ユーティリティと REPLICAT ユーティリティを使用する

- Oracle GoldenGate のモニタリング
- Oracle GoldenGate のトラブルシューティング

Oracle GoldenGate でサポートされているバージョンとライセンスオプション

RDS for Oracle の Standard Edition 2 (SE2) または Enterprise Edition (EE) を Oracle GoldenGate バージョン 12c 以降と使用できます。Oracle GoldenGate の次の機能を使用できます。

- Oracle GoldenGate Remote キャプチャ (抽出) がサポートされています。
- キャプチャ (抽出) は、従来の非 CDB データベースアーキテクチャを使用する RDS for Oracle DB インスタンスでサポートされています。Oracle GoldenGate リモート PDB キャプチャは、Oracle Database 26ai、Oracle Database 21c または Oracle Database 19c バージョン 19.0.0.0.ru-2024-04.rur-2024-04.r1 以降を実行している CDB でサポートされています。
- Oracle GoldenGate Remote Delivery (replicat) は、非 CDB または CDB アーキテクチャを使用する RDS for Oracle DB インスタンスでサポートされています。Remote Delivery は、Integrated Replicat、Parallel Replicat、Coordinated Replicat、およびクラシック Replicat をサポートします。
- RDS for Oracle は、Oracle GoldenGate のクラシックアーキテクチャとマイクロサービスアーキテクチャをサポートしています。
- 統合キャプチャモードを使用する場合、Oracle GoldenGate DDL およびシーケンス値のレプリケーションがサポートされます。

お客様には、すべての AWS リージョン で Amazon RDS で使用するために、Oracle GoldenGate ライセンス (BYOL) を管理する責任があります。詳細については、「RDS for Oracle のライセンスオプション」を参照してください。

Oracle GoldenGate の要件と制限

Oracle GoldenGate および RDS for Oracle を使用する場合、次の要件と制限を考慮してください。

- ユーザーには、RDS for Oracle で使用するために Oracle GoldenGate を設定および管理する責任があります。
- ユーザーには、ソースデータベースとターゲットデータベースで認定された Oracle GoldenGate バージョンを設定する責任があります。詳細については、Oracle ドキュメントの「Oracle Fusion Middleware Supported System Configurations」を参照してください。

- Oracle GoldenGate は、さまざまな AWS 環境でさまざまなユースケースに使用できます。Oracle GoldenGate に関するサポート関連の問題がある場合は、Oracle サポートサービスにお問い合わせください。
- Oracle Transparent Data Encryption (TDE) を使用する RDS for Oracle DB インスタンスで、Oracle GoldenGate を使用できます。レプリケートされたデータの整合性を維持するには、Amazon EBS 暗号化ボリュームまたはトレイルファイル暗号化を使用して Oracle GoldenGate ハブの暗号化を設定します。また、Oracle GoldenGate ハブと、ソースおよびターゲットのデータベースインスタンスの間で送信されるデータの暗号化も設定します。RDS for Oracle DB インスタンスは、Oracle Secure Sockets Layer または Oracle ネイティブネットワーク暗号化 を使用する暗号化をサポートしています。

Oracle GoldenGate アーキテクチャ

Amazon RDS で使用される Oracle GoldenGate アーキテクチャは、次の独立したモジュールで構成されています。

ソースデータベース

ソースデータベースは、オンプレミスの Oracle データベース、Amazon EC2 インスタンス上の Oracle データベース、Amazon RDS DB インスタンス上の Oracle データベースのいずれかです。

Oracle GoldenGate ハブ

GoldenGate ハブは、トランザクション情報をソースデータベースからターゲットデータベースに移動します。ハブは次のいずれかになります。

- Oracle Database と Oracle GoldenGate がインストールされた Amazon EC2 インスタンス
- オンプレミスの Oracle インストール

複数の Amazon EC2 ハブを使用できます。クロスリージョンレプリケーションで Oracle GoldenGate を使用する場合は、2 つのハブを使用することをお勧めします。

ターゲットデータベース

ターゲットデータベースは、 Amazon RDS DB インスタンス、 Amazon EC2 インスタンス、オンプレミスの場所のいずれかに配置できます。

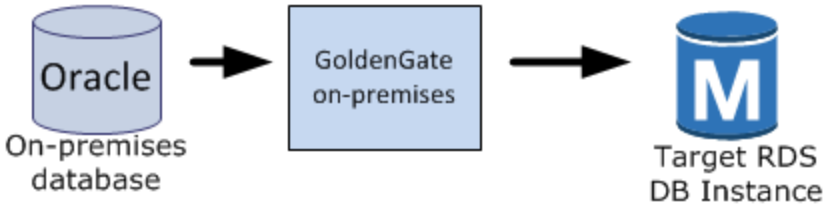
次のセクションでは、Amazon RDS での Oracle GoldenGate の一般的なシナリオについて説明します。

トピック

- オンプレミスのソースデータベースと Oracle GoldenGate ハブ
- オンプレミスのソースデータベースおよび Amazon EC2 ハブ
- Amazon RDS ソースデータベースおよび Amazon EC2 ハブ
- Amazon EC2 ソースデータベースと Amazon EC2 ハブ
- 異なる AWS リージョンの Amazon EC2 ハブ

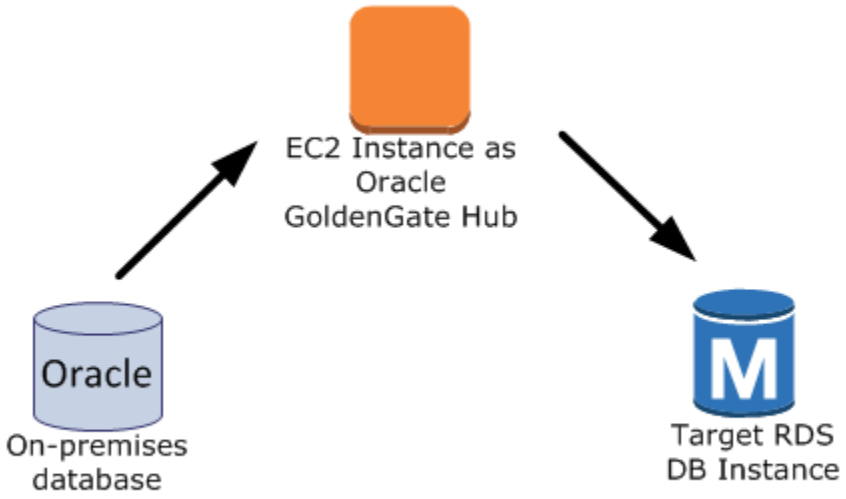
オンプレミスのソースデータベースと Oracle GoldenGate ハブ

このシナリオでは、オンプレミスの Oracle ソースデータベースとオンプレミスの Oracle GoldenGate ハブから、ターゲットとなる Amazon RDS DB インスタンスにデータが提供されます。



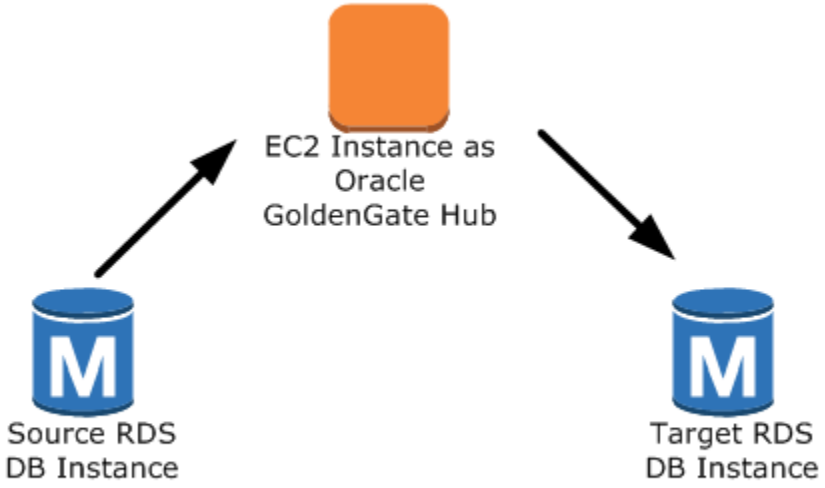
オンプレミスのソースデータベースおよび Amazon EC2 ハブ

このシナリオでは、オンプレミス Oracle データベースがソースデータベースとして機能します。Amazon EC2 インスタンスハブに接続されています。このハブから、ターゲットの RDS for Oracle DB インスタンスにデータが提供されます。



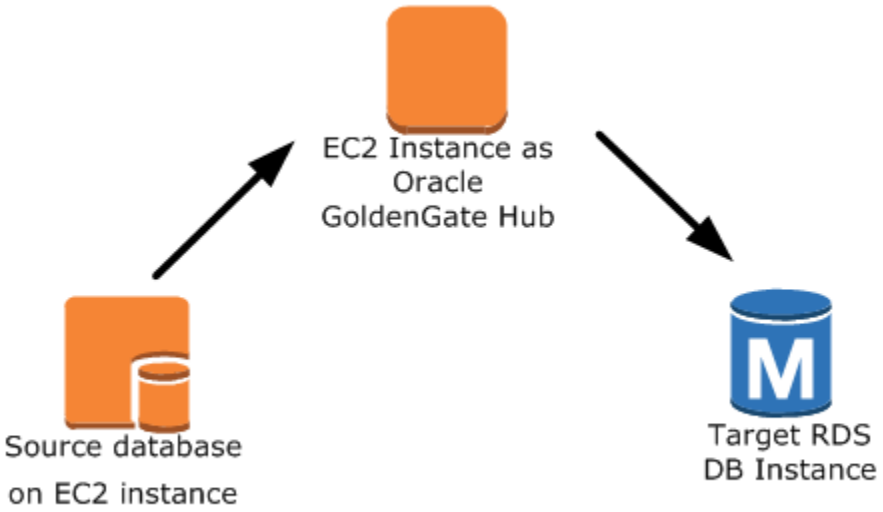
Amazon RDS ソースデータベースおよび Amazon EC2 ハブ

このシナリオでは、RDS for Oracle DB インスタンスがソースデータベースとして機能します。Amazon EC2 インスタンスハブに接続されています。このハブから、ターゲットの RDS for Oracle DB インスタンスにデータが提供されます。



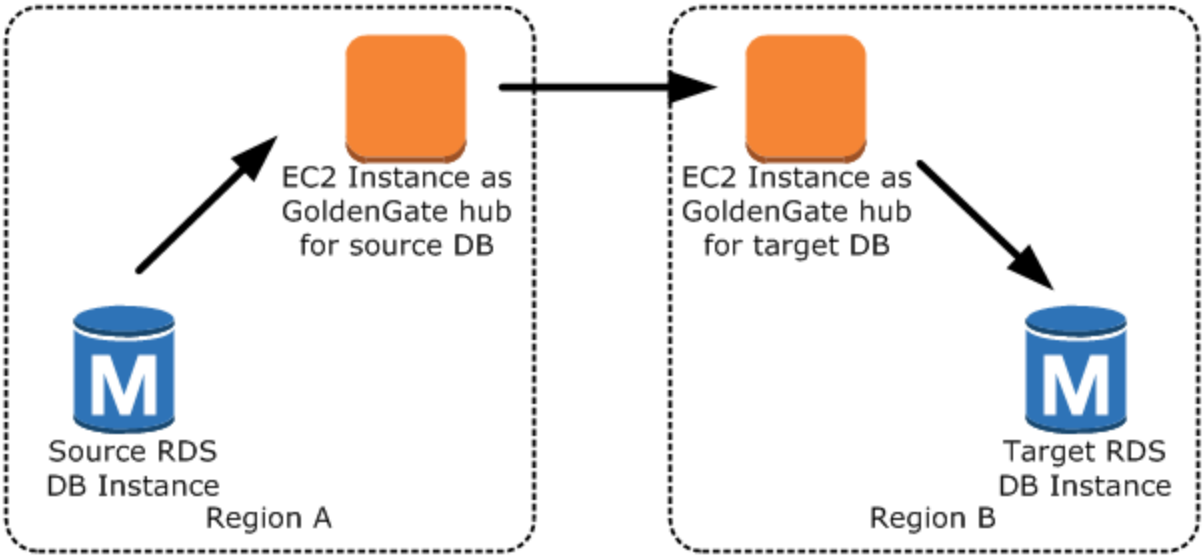
Amazon EC2 ソースデータベースと Amazon EC2 ハブ

このシナリオでは、Amazon EC2 インスタンス上の Oracle データベースがソースデータベースとして機能します。Amazon EC2 インスタンスハブに接続されています。このハブから、ターゲットの RDS for Oracle DB インスタンスにデータが提供されます。



異なる AWS リージョンの Amazon EC2 ハブ

このシナリオでは、Amazon RDS DB インスタンス上の Oracle データベースが、同じ AWS リージョンの Amazon EC2 インスタンスハブに接続されています。ハブは別の AWS リージョンの Amazon EC2 インスタンスハブに接続されています。この 2 番目のハブは、2 番目の Amazon EC2 インスタンスハブとして、同じ AWS リージョン内のターゲット RDS for Oracle DB インスタンスにデータを提供します。



> **Note**
>
> オンプレミス環境で実行中の Oracle GoldenGate に影響する問題は、AWS で実行中の Oracle GoldenGate にも影響します。Oracle GoldenGate ハブをモニタリングし、フェイルオーバーが発生した場合に `EXTRACT` や `REPLICAT` が再起動されていることを確認することを強くお勧めします。Oracle GoldenGate ハブは Amazon EC2 インスタンスで実行されるため、Amazon RDS では Oracle GoldenGate ハブが管理されず、実行中であるかどうかを確認できません。

Oracle GoldenGate のセットアップ

Amazon RDS を使用して Oracle GoldenGate をセットアップするには、Amazon EC2 インスタンス上にハブを設定し、ソースデータベースとターゲットデータベースを設定します。以下のセクションは、Amazon RDS for Oracle で使用するために Oracle GoldenGate を設定する方法を示しています。

トピック

- Amazon EC2 での Oracle GoldenGate ハブのセットアップ
- Amazon RDS の Oracle GoldenGate 用の出典データベースのセットアップ
- Amazon RDS で Oracle GoldenGate 用のターゲットデータベースをセットアップする

Amazon EC2 での Oracle GoldenGate ハブのセットアップ

Amazon EC2 インスタンスに Oracle GoldenGate ハブを作成するには、まず Oracle RDBMS の完全なクライアントインストールで Amazon EC2 インスタンスを作成します。Amazon EC2 インスタンスには、Oracle GoldenGate ソフトウェアもインストールしている必要があります。Oracle GoldenGate ソフトウェアのバージョンは、ソースおよびターゲットのデータベースのバージョンにより決定されます。Oracle GoldenGate のインストールの詳細については、Oracle GoldenGate のドキュメントを参照してください。

Oracle GoldenGate ハブとして機能する Amazon EC2 インスタンスは、ソースデータベースのトランザクション情報をトレイルファイルに保存し、処理します。このプロセスをサポートするには、次の要件を満たしていることを確認してください。

- トレイルファイル用に十分なストレージを割り当てています。
- Amazon EC2 インスタンスには、データ量を管理するのに十分な処理能力がある。

- トレイルファイルに書き込む前に、EC2 インスタンスにトランザクション情報を保存するのに十分なメモリがある。

Amazon EC2 インスタンスに Oracle GoldenGate クラシックアーキテクチャハブを設定するには

1. Oracle GoldenGate ディレクトリにサブディレクトリを作成します。

   Amazon EC2 コマンドラインシェルで、Oracle GoldenGate コマンドインタープリタ `ggsci` を起動します。`CREATE SUBDIRS` コマンドは、パラメータ、レポート、チェックポイントの各ファイル用の `/gg` ディレクトリの下にサブディレクトリを作成します。

   ```
   prompt$ cd /gg
   prompt$ ./ggsci

   GGSCI> CREATE SUBDIRS
   ```

2. `mgr.prm` ファイルを設定します。

   次の例では、`$GGHOME/dirprm/mgr.prm` ファイルに行を追加します。

   ```
   PORT 8199
   PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
   ```

3. マネージャーを起動します。

   次の例では、`ggsci` を起動し、`start mgr` コマンドを実行します。

   ```
   GGSCI> start mgr
   ```

これで、Oracle GoldenGate ハブを使用する準備ができました。

Amazon RDS の Oracle GoldenGate 用の出典データベースのセットアップ

Oracle GoldenGate で使用するソースデータベースを設定するには、次のタスクを実行します。

セットアップステップ

- ステップ 1: ソースデータベースでの補足的な口グ記録を有効にする
- ステップ 2: ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定する
- ステップ 3: ソースデータベースでのログの保持期間を設定する

- ステップ 4: ソースデータベースに Oracle GoldenGate ユーザーアカウントを作成する
- ステップ 5: ソースデータベースでユーザーに特権を付与する
- ステップ 6: ソースデータベースに TNS エイリアスを追加する

ステップ 1: ソースデータベースでの補足的なログ記録を有効にする

最小限のデータベースレベルの補足的なログ記録を有効にするには、次の PL/SQL プロシージャを実行します。

```
EXEC rdsadmin.rdsadmin_util.alter_supplemental_logging(p_action => 'ADD')
```

ステップ 2: ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定する

ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定すると、データベースサービスが論理レプリケーションをサポートできるようになります。ソースデータベースが Amazon RDS DB インスタンスにある場合、ENABLE_GOLDENGATE_REPLICATION インストールパラメータを true に設定して、DB インスタンスにパラメータグループを割り当てる必要があります。ENABLE_GOLDENGATE_REPLICATION の初期化パラメータの詳細については、Oracle Database のドキュメントを参照してください。

> **Note**
>
> これは動的パラメータです。この変更は、変更後のパラメータグループを DB インスタンスに適用した後に有効になります。再起動する必要はありません。設定を確認するには、SHOW PARAMETER ENABLE_GOLDENGATE_REPLICATION を実行します。

ステップ 3: ソースデータベースでのログの保持期間を設定する

アーカイブされた REDO ログを保持するようにソースデータベースを必ず設定してください。以下のガイドラインを検討します。

- ログの保持期間を時間単位で指定します。最小値は 1 時間です。
- この期間は、ソース DB インスタンスの潜在的なダウンタイム、潜在的な通信期間、ソースインスタンスのネットワーク問題の潜在的な期間を超えるように設定します。このような期間により、Oracle GoldenGate は必要に応じて出典インスタンスからログを復元できます。
- ファイル用の十分なストレージがインスタンスにあることを確認します。

例えば、アーカイブ REDO ログの保持期間を 24 時間に設定します。

```
EXEC rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24)
```

ログの保持を有効にしていない場合や、保持の値が小さすぎる場合、次のようなエラーメッセージが表示されます。

```
2022-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log for sequence
 1306
Not able to establish initial position for begin time 2022-03-06 06:16:55.
```

DB インスタンスにはアーカイブされた REDO ログが保持されるため、ファイル用の十分なスペースがあることを確認してください。過去 *num_hours* 時間で使用した領域の量を確認するには、*num_hours* を時間数に置き換えて、次のクエリを実行します。

```
SELECT SUM(BLOCKS * BLOCK_SIZE) BYTES FROM V$ARCHIVED_LOG
   WHERE NEXT_TIME>=SYSDATE-num_hours/24 AND DEST_ID=1;
```

ステップ 4: ソースデータベースに Oracle GoldenGate ユーザーアカウントを作成する

Oracle GoldenGate はデータベースユーザーとして実行され、出典データベースの REDO ログとアーカイブ REDO ログにアクセスするには、適切なデータベース権限が必要です。これらを指定するには、ソースデータベースにユーザーアカウントを作成します。Oracle GoldenGate のユーザーアカウントに関するアクセス許可の詳細については、Oracle のドキュメントを参照してください。

次のステートメントでは、oggadm1 という名前のユーザーアカウントが作成されます。

```
CREATE TABLESPACE administrator;
CREATE USER oggadm1  IDENTIFIED BY "password"
   DEFAULT TABLESPACE ADMINISTRATOR TEMPORARY TABLESPACE TEMP;
ALTER USER oggadm1 QUOTA UNLIMITED ON administrator;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

ステップ 5: ソースデータベースでユーザーに特権を付与する

このタスクでは、ソースデータベースでデータベースユーザーに必要なアカウント権限を付与します。

ソースデータベースでユーザーに特権を付与するには

1. SQL コマンド `grant` と `rdsadmin.rdsadmin_util` プロシージャ `grant_sys_object` を使用して、Oracle GoldenGate ユーザーアカウントに必要な権限を付与します。次のステートメントでは、oggadm1 という名前のユーザーに権限が付与されます。

```
GRANT CREATE SESSION, ALTER SESSION TO oggadm1;
GRANT RESOURCE TO oggadm1;
GRANT SELECT ANY DICTIONARY TO oggadm1;
GRANT FLASHBACK ANY TABLE TO oggadm1;
GRANT SELECT ANY TABLE TO oggadm1;
GRANT SELECT_CATALOG_ROLE TO rds_master_user_name WITH ADMIN OPTION;
EXEC rdsadmin.rdsadmin_util.grant_sys_object ('DBA_CLUSTERS', 'OGGADM1');
GRANT EXECUTE ON DBMS_FLASHBACK TO oggadm1;
GRANT SELECT ON SYS.V_$DATABASE TO oggadm1;
GRANT ALTER ANY TABLE TO oggadm1;
```

2. Oracle GoldenGate 管理者になるためにユーザーアカウントに必要な権限を付与します。以下の PL/SQL プログラムを実行します。

```
EXEC rdsadmin.rdsadmin_dbms_goldengate_auth.grant_admin_privilege (
    grantee                 => 'OGGADM1',
    privilege_type          => 'capture',
    grant_select_privileges => true,
    do_grants               => TRUE);
```

   権限を取り消すには、同じパッケージのプロシージャ `revoke_admin_privilege` を使用します。

ステップ 6: ソースデータベースに TNS エイリアスを追加する

`$ORACLE_HOME/network/admin/tnsnames.ora` プロセスで使用する Oracle Home で、次のエントリを EXTRACT に追加します。`tnsnames.ora` ファイルの詳細については、Oracle のドキュメントを参照してください。

```
OGGSOURCE=
```

```
   (DESCRIPTION=
        (ENABLE=BROKEN)
        (ADDRESS_LIST=
            (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-source.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=1521)))
        (CONNECT_DATA=(SERVICE_NAME=ORCL))
    )
```

Amazon RDS で Oracle GoldenGate 用のターゲットデータベースをセットアップする

このタスクでは、Oracle GoldenGate で使用するターゲット DB インスタンスを設定します。

セットアップステップ

- ステップ 1: ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定する
- ステップ 2: ターゲットデータベースで GoldenGate のユーザーアカウントを作成する
- ステップ 3: ターゲットデータベースでアカウント権限を付与する
- ステップ 4: ターゲットデータベースに TNS エイリアスを追加する

ステップ 1: ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定する

ENABLE_GOLDENGATE_REPLICATION 初期化パラメータを true に設定すると、データベースサービスが論理レプリケーションをサポートできるようになります。ソースデータベースが Amazon RDS DB インスタンスにある場合、ENABLE_GOLDENGATE_REPLICATION インストールパラメータを true に設定して、DB インスタンスにパラメータグループを割り当てる必要があります。ENABLE_GOLDENGATE_REPLICATION の初期化パラメータの詳細については、Oracle Database のドキュメントを参照してください。

ステップ 2: ターゲットデータベースで GoldenGate のユーザーアカウントを作成する

Oracle GoldenGate はデータベースユーザーとして動作し、適切なデータベース権限が必要です。これらの権限があることを確認するには、ターゲットデータベースにユーザーアカウントを作成します。

次のステートメントでは、oggadm1 という名前のユーザーを作成します。

```
CREATE TABLESPACE administrator;
CREATE USER oggadm1  IDENTIFIED BY "password"
   DEFAULT TABLESPACE administrator
   TEMPORARY TABLESPACE temp;
ALTER USER oggadm1 QUOTA UNLIMITED ON administrator;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

ステップ 3: ターゲットデータベースでアカウント権限を付与する

このタスクでは、ターゲットデータベースでデータベースユーザーに必要なアカウント権限を付与します。

ターゲットデータベースでユーザーに特権を付与するには

1. ターゲットデータベースの Oracle GoldenGate ユーザーアカウントに必要な権限を付与します。次の例では、oggadm1 に権限を付与します。

```
GRANT CREATE SESSION        TO oggadm1;
GRANT ALTER SESSION         TO oggadm1;
GRANT CREATE CLUSTER        TO oggadm1;
GRANT CREATE INDEXTYPE      TO oggadm1;
GRANT CREATE OPERATOR       TO oggadm1;
GRANT CREATE PROCEDURE      TO oggadm1;
GRANT CREATE SEQUENCE       TO oggadm1;
GRANT CREATE TABLE          TO oggadm1;
GRANT CREATE TRIGGER        TO oggadm1;
GRANT CREATE TYPE           TO oggadm1;
GRANT SELECT ANY DICTIONARY TO oggadm1;
GRANT CREATE ANY TABLE      TO oggadm1;
GRANT ALTER ANY TABLE       TO oggadm1;
GRANT LOCK ANY TABLE        TO oggadm1;
GRANT SELECT ANY TABLE      TO oggadm1;
GRANT INSERT ANY TABLE      TO oggadm1;
GRANT UPDATE ANY TABLE      TO oggadm1;
GRANT DELETE ANY TABLE      TO oggadm1;
```

2. Oracle GoldenGate 管理者になるためにユーザーアカウントに必要な権限を付与します。以下の PL/SQL プログラムを実行します。

```
EXEC rdsadmin.rdsadmin_dbms_goldengate_auth.grant_admin_privilege (
    grantee                 => 'OGGADM1',
    privilege_type          => 'apply',
    grant_select_privileges => true,
```

```
    do_grants               => TRUE);
```

権限を取り消すには、同じパッケージのプロシージャ `revoke_admin_privilege` を使用します。

ステップ 4: ターゲットデータベースに TNS エイリアスを追加する

`$ORACLE_HOME/network/admin/tnsnames.ora` プロセスで使用する Oracle Home で、次のエントリを `REPLICAT` に追加します。Oracle マルチテナントデータベースの場合、TNS エイリアスが PDB のサービス名を指していることを確認してください。`tnsnames.ora` ファイルの詳細については、Oracle のドキュメントを参照してください。

```
OGGTARGET=
    (DESCRIPTION=
        (ENABLE=BROKEN)
        (ADDRESS_LIST=
            (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-target.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=1521)))
        (CONNECT_DATA=(SERVICE_NAME=ORCL))
    )
```

Oracle GoldenGate の EXTRACT ユーティリティと REPLICAT ユーティリティを使用する

Oracle GoldenGate のユーティリティである `EXTRACT` と `REPLICAT` は連携して動作し、トレイルファイルを使った増分トランザクションのレプリケーションを利用して、出典データベースとターゲットデータベースの同期を維持します。ソースデータベースで行われたすべての変更は `EXTRACT` によって自動的に検出され、その後整形されて、オンプレミスの Oracle GoldenGate または Amazon EC2 インスタンスハブ上のトレイルファイルに転送されます。トレイルファイルの初回の読み取りが完了すると、`REPLICAT` ユーティリティによってデータがこれらのファイルから読み取られ、ターゲットデータベースにレプリケートされます。

Oracle GoldenGate の EXTRACT ユーティリティの実行

`EXTRACT` ユーティリティは、ソースデータベースのデータを取得し、変換して、トレイルファイルに出力します。基本的なプロセスは、次のとおりです。

1. `EXTRACT` は、トランザクションの詳細をメモリまたは一時ディスクストレージ上のキューに入れます。

2. ソースデータベースがトランザクションをコミットします。
3. EXTRACT は、トランザクションの詳細をトレイルファイルに書き込みます。
4. トレイルファイルは、これらの詳細を Oracle GoldenGate オンプレミスまたは Amazon EC2 インスタンスハブにルーティングしてから、ターゲットデータベースにルーティングします。

次の手順では、EXTRACT ユーティリティを起動し、ソースデータベース OGGSOURCE で EXAMPLE.TABLE からデータをキャプチャして、トレイルファイルを作成します。

EXTRACT ユーティリティを実行するには

1. Oracle GoldenGate ハブ (オンプレミスまたは Amazon EC2 インスタンス) にある EXTRACT パラメータファイルを設定します。$GGHOME/dirprm/eabc.prm という名前のサンプルの EXTRACT パラメータファイルを次に示します。

```
EXTRACT EABC

USERID oggadm1@OGGSOURCE, PASSWORD "my-password"
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

2. Oracle GoldenGate ハブで、ソースデータベースにログインし、Oracle GoldenGate コマンドラインインターフェイス ggsci を起動します。次の例は、ログインの形式を示しています。

```
dblogin userid oggadm1@OGGSOURCE
```

3. トランザクションデータを追加して、データベーステーブルの補足的なログ記録を有効にします。

```
add trandata EXAMPLE.TABLE
```

4. ggsci コマンドラインを使用する場合、次のコマンドを使用して、EXTRACT ユーティリティを有効にします。

```
add extract EABC tranlog, INTEGRATED tranlog, begin now
add exttrail /path/to/goldengate/dirdat/ab
```

```
   extract EABC,
   MEGABYTES 100
```

5. アーカイブログが削除されないように、データベースに EXTRACT ユーティリティを登録します。このタスクにより、以前のコミットされていないトランザクションを必要に応じて復旧できます。データベースに EXTRACT ユーティリティを登録するには、次のコマンドを使用します。

```
register EXTRACT EABC, DATABASE
```

6. 次のコマンドを使用して EXTRACT ユーティリティを起動します。

```
start EABC
```

Oracle GoldenGate の REPLICAT ユーティリティの実行

REPLICAT ユーティリティは、トレイルファイルのトランザクション情報をターゲットデータベースに「プッシュ」します。

次のステップでは、REPLICAT ユーティリティを有効にして起動し、キャプチャしたデータをターゲットデータベース OGGTARGET のテーブル EXAMPLE.TABLE にレプリケートできるようにします。

REPLICAT ユーティリティを実行するには

1. Oracle GoldenGate ハブ (オンプレミスまたは EC2 インスタンス) にある REPLICAT パラメータファイルを設定します。$GGHOME/dirprm/rabc.prm という名前のサンプルの REPLICAT パラメータファイルを次に示します。

```
REPLICAT RABC

USERID oggadm1@OGGTARGET, password "my-password"

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

2. ターゲットデータベースにログインし、Oracle GoldenGate コマンドラインインターフェイス (ggsci) を起動します。次の例は、ログインの形式を示しています。

```
dblogin userid oggadm1@OGGTARGET
```

3. ggsci コマンドラインを使用して、チェックポイントテーブルを追加します。ここで指定するユーザーは、ターゲットテーブルスキーマの所有者ではなく、Oracle GoldenGate のユーザーアカウントであることが必要です。次の例では、gg_checkpoint という名前のチェックポイントテーブルを作成します。

```
add checkpointtable oggadm1.oggchkpt
```

4. REPLICAT ユーティリティを有効にするには、次のコマンドを使用します。

```
add replicat RABC EXTTRAIL /path/to/goldengate/dirdat/ab CHECKPOINTTABLE
 oggadm1.oggchkpt
```

5. 次のコマンドを使用して、REPLICAT ユーティリティを起動します。

```
start RABC
```

Oracle GoldenGate のモニタリング

Oracle GoldenGate をレプリケーションに使用する場合は、Oracle GoldenGate プロセスが稼働していて、ソースデータベースとターゲットデータベースが同期していることを確認してください。次のモニタリングツールを使用できます。

- Amazon CloudWatch は、このようなパターンで GoldenGate エラーログをモニタリングするために使用されるモニタリングサービスです。
- Amazon SNS は、このようなパターンでメール通知を送信するために使用されるメッセージ通知サービスです。

詳細な説明については、「Monitor Oracle GoldenGate logs by using Amazon CloudWatch」(Amazon CloudWatch を使用して Oracle GoldenGate ログをモニタリングする) を参照してください。

Oracle GoldenGate のトラブルシューティング

このセクションでは、Amazon RDS for Oracle で Oracle GoldenGate を使用する場合に発生する最も一般的な問題について説明します。

トピック

- オンライン REDO ログを開くときにエラーが発生しました
- Oracle GoldenGate は適切に設定されているようだが、レプリケーションが機能していない
- SYS."_DBA_APPLY_CDR_INFO" に対するクエリが原因で、統合 REPLICAT が遅い

オンライン REDO ログを開くときにエラーが発生しました

アーカイブされた REDO ログを保持するようにデータベースを設定していることを確認してください。以下のガイドラインを検討します。

- ログの保持期間を時間単位で指定します。最小値は 1 時間です。
- この期間は、ソースインスタンスの潜在的なダウンタイム、潜在的な通信期間、ソースインスタンスのネットワーク問題の潜在的な期間を超えるように設定します。このような期間により、Oracle GoldenGate は必要に応じてソース DB インスタンスからログを復元できます。
- ファイル用の十分なストレージがインスタンスにあることを確認します。

ログの保持を有効にしていない場合や、保持の値が小さすぎる場合、次のようなエラーメッセージが表示されます。

```
2022-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log for sequence
 1306
Not able to establish initial position for begin time 2022-03-06 06:16:55.
```

Oracle GoldenGate は適切に設定されているようだが、レプリケーションが機能していない

既存のテーブルについては、Oracle GoldenGate の動作元の SCN を指定する必要があります。

この問題を修正するには

1. ソースデータベースにログインし、Oracle GoldenGate コマンドラインインターフェイス (ggsci) を起動します。次の例は、ログインの形式を示しています。

```
dblogin userid oggadm1@OGGSOURCE
```

2. ggsci コマンドラインを使用して、EXTRACT プロセスのスタート SCN を設定します。次の例では、EXTRACT の SCN を 223274 に設定しています。

```
ALTER EXTRACT EABC SCN 223274
start EABC
```

3. ターゲットデータベースにログインします。次の例は、ログインの形式を示しています。

```
dblogin userid oggadm1@OGGTARGET
```

4. ggsci コマンドラインを使用して、REPLICAT プロセスのスタート SCN を設定します。次の例では、REPLICAT の SCN を 223274 に設定しています。

```
start RABC atcsn 223274
```

SYS."_DBA_APPLY_CDR_INFO" に対するクエリが原因で、統合 REPLICAT が遅い

Oracle GoldenGate 競合の検出および解決 (CDR) には、ベーシックな競合の解決ルーチンが用意されています。例えば、CDR は、INSERT ステートメントの一意の競合を解決できます。

CDR が衝突を解決すると、一時的に例外テーブル _DBA_APPLY_CDR_INFO にレコードを挿入できます。Integrated REPLICAT はこれらのレコードを後で削除します。まれなシナリオでは、統合 REPLICAT は多数の衝突を処理できますが、新しい統合 REPLICAT では置き換えられません。_DBA_APPLY_CDR_INFO の既存の行は、削除されるのではなく、孤立します。任意の新しい統合 REPLICAT プロセスは、_DBA_APPLY_CDR_INFO で孤立した行をクエリするため、処理速度が低下します。

_DBA_APPLY_CDR_INFO からすべての行を削除するには、Amazon RDS の手順 `rdsadmin.rdsadmin_util.truncate_apply$_cdr_info` を使用します。この手順は、2020 年 10 月のリリースおよびパッチアップデートの一部としてリリースされます。この手順は、次のデータベースバージョンで使用できます。

- バージョン 21.0.0.0.ru-2022-01.rur-2022-01.r1 以上
- バージョン 19.0.0.0.ru-2020-10.rur-2020-10.r1 以降

次の例では、テーブル _DBA_APPLY_CDR_INFO を切り捨てます。

```
SET SERVEROUTPUT ON SIZE 2000
EXEC rdsadmin.rdsadmin_util.truncate_apply$_cdr_info;
```

Amazon RDS for Oracle での Oracle リポジトリ作成ユーティリティの使用

Amazon RDS を使用すると、Oracle Fusion Middleware コンポーネントをサポートするスキーマを保持する RDS for Oracle DB インスタンスをホストできます。Fusion Middleware コンポーネントを使用する前に、データベースでそれらのコンポーネント用のスキーマを作成してデータを入力します。スキーマの作成とデータ入力は、Oracle Repository Creation Utility (RCU) を使用して行います。

RCU でサポートされているバージョンとライセンスオプション

Amazon RDS では、Oracle Repository Creation Utility (RCU) バージョン 12c のみサポートされています。RCU は以下の構成で使用できます。

- Oracle データベース 21c を使用する RCU 12c
- Oracle データベース 19c を使用する RCU 12c

RCU を使用する前に、以下を実行します。

- Oracle Fusion Midleware のライセンスを取得する。
- リポジトリをホストする Oracle データベースの Oracle ライセンスガイドラインに従う。詳細については、Oracle ドキュメントの「Oracle Fusion Middleware Licensing Information User Manual」を参照してください。

Fusion Middleware は、Oracle Database Enterprise Edition および Standard Edition 2 上のリポジトリをサポートしています。Oracle では、オンラインインデックスの再構築を必要とするパーティション化とインストールが必要な本稼働インストールには、Enterprise Edition を推奨しています。

RDS for Oracle DB インスタンスを作成する前に、デプロイするコンポーネントをサポートするために必要な Oracle データベースバージョンを確認します。デプロイする Fusion Middleware コンポーネントおよびバージョンの要件を確認するには、認定マトリックスを使用します。詳細については、Oracle ドキュメントの「Oracle Fusion Middleware Supported System Configurations」を参照してください。

Amazon RDS では、必要に応じて Oracle データベースバージョンのアップグレードがサポートされます。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

RCU の要件と制限

RCU を使用するには、Amazon VPC が必要です。Amazon RDS DB インスタンスは、パブリックインターネットではなく、Fusion Middleware コンポーネントでのみ使用できるようにします。そのため、Amazon RDS DB インスタンスをプライベートサブネットでホストして、セキュリティを強化します。RDS for Oracle DB インスタンスも必要です。詳細については、「Oracle DB インスタンスを作成して接続する」を参照してください。

どの Fusion Middleware コンポーネントのスキーマでも Amazon RDS DB インスタンスに保存できます。次のスキーマは、正しくインストールされることが検証されています。

- Analytics (ACTIVITIES)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Discussions (DISCUSSIONS)
- Metadata Services (MDS)
- Oracle Business Intelligence (BIPLATFORM)
- Oracle Platform Security Services (OPSS)
- Portal and Services (WEBCENTER)
- Portlet Producers (PORTLET)
- Service Table (STB)
- SOA Infrastructure (SOAINFRA)
- User Messaging Service (UCSUMS)
- WebLogic Services (WLS)

RCU を使用するガイドライン

このシナリオで DB インスタンスを使用する場合の推奨事項を以下に示します。

- マルチ AZ は、本稼働ワークロードに使用することをお勧めします。複数のアベイラビリティーゾーンの使用の詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。
- セキュリティを高めるため、Oracle では、Transparent Data Encryption (TDE) を使用して保管時のデータを暗号化することを推奨しています。高度なセキュリティオプションを含む Enterprise

Edition のライセンスをお持ちの場合、TDE オプションを使用して保管時の暗号化を有効にできます。詳細については、「Oracle Transparent Data Encryption」を参照してください。

Amazon RDS では、すべてのデータベースエディション用の保管時の暗号化オプションも用意しています。詳細については、「Amazon RDS リソースの暗号化」を参照してください。

- アプリケーションサーバーと Amazon RDS DB インスタンスの間の通信が許可されるように、VPC セキュリティグループを設定します。Fusion Middleware コンポーネントをホストするアプリケーションサーバーは、Amazon EC2 またはオンプレミスにすることができます。

RCU の実行

Fusion Middleware コンポーネントをサポートするためのスキーマを作成し、データを入力するには、Oracle Repository Creation Utility (RCU) を使用します。RCU は、さまざまな方法で実行できます。

トピック

- コマンドラインを使用した 1 ステップでの RCU の実行
- コマンドラインを使用した複数ステップでの RCU の実行
- インタラクティブモードでの RCU の実行

コマンドラインを使用した 1 ステップでの RCU の実行

データを入力する前にスキーマを編集する必要がない場合は、単一ステップで RCU を実行できます。それ以外の場合、複数ステップでの RCU の実行について次のセクションを参照してください。

コマンドラインパラメータ `-silent` を使用して、RCU をサイレントモードで実行できます。サイレントモードで RCU を実行すると、パスワードを含むテキストファイルを作成することで、コマンドラインにパスワードを入力する必要がなくなります。1 行目に `dbUser` のパスワードを含み、それ以降の行に各コンポーネントのパスワードを含むテキストファイルを作成します。RCU コマンドの最後のパラメータとして、パスワードファイルの名前を指定します。

Example

次の例では、SOA Infrastructure コンポーネント (およびその依存関係) のスキーマを単一ステップで作成し、データを入力します。

Linux、macOS、Unix の場合:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

詳細については、Oracle ドキュメントの「Running Repository Creation Utility from the Command Line」を参照してください。

コマンドラインを使用した複数ステップでの RCU の実行

スキーマスクリプトを手動で編集するには、次の複数のステップで RCU を実行します。

1. `-generateScript` コマンドラインパラメータを使用して [Prepare Scripts for System Load] モードで RCU を実行し、スキーマのスクリプトを作成します。
2. 手動で編集し、生成したスクリプト `script_systemLoad.sql` を実行します。
3. `-dataLoad` コマンドラインパラメータを使用して [Perform Product Load] モードでもう一度 RCU を実行し、スキーマにデータを入力します。
4. 生成したクリーンアップのスクリプト `script_postDataLoad.sql` を実行します。

RCU をサイレントモードで実行するには、コマンドラインパラメータ `-silent` を指定します。サイレントモードで RCU を実行すると、パスワードを含むテキストファイルを作成することで、コマンドラインにパスワードを入力する必要がなくなります。1 行目に dbUser のパスワードを含み、それ以降の行に各コンポーネントのパスワードを含むテキストファイルを作成します。RCU コマンドの最後のパラメータとして、パスワードファイルの名前を指定します。

Example

次の例では、SOA Infrastructure コンポーネントおよびその依存関係のスキーマスクリプトを作成します。

Linux、macOS、Unix の場合:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-generateScript \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
[-encryptTablespace true] \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-scriptLocation /tmp/rcuscripts \
-f < /tmp/passwordfile.txt
```

ここでは、生成されたスクリプトを編集し、Oracle DB インスタンスに接続してスクリプトを実行できます。生成したスクリプトの名前は、`script_systemLoad.sql` です。Oracle DB インスタンスへの接続の詳細については、「ステップ 3: SQL クライアントを Oracle DB インスタンスに接続する」を参照してください。

次の例では、SOA Infrastructure コンポーネント (およびその依存関係) のスキーマにデータを入力します。

Linux、macOS、Unix の場合:

```
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
```

```
-silent \
-dataLoad \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

終了するには、Oracle DB インスタンスに接続し、クリーンアップスクリプトを実行します。生成したスクリプトの名前は、script_postDataLoad.sql です。

詳細については、Oracle ドキュメントの「Running Repository Creation Utility from the Command Line」を参照してください。

インタラクティブモードでの RCU の実行

RCU グラフィカルユーザーインターフェイスを使用するには、インタラクティブモードで RCU を実行します。-interactive パラメータを指定し、-silent パラメータは指定しません。詳細については、Oracle ドキュメントの「Understanding Repository Creation Utility Screens」を参照してください。

Example

次の例では、RCU をインタラクティブモードで起動し、接続情報を事前入力します。

Linux、macOS、Unix の場合:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-interactive \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
```

```
-dbUser ${dbuser} \
-dbRole Normal
```

RCU のトラブルシューティング

次の点に注意してください。

トピック

- Oracle Managed Files (OMF)
- オブジェクト権限
- Enterprise Scheduler Service

Oracle Managed Files (OMF)

Amazon RDS は、OMF データファイルを使用してストレージ管理を簡素化します。サイズや拡張管理などのテーブルスペース属性をカスタマイズできます。ただし、RCU の実行時にデータファイル名を指定すると、テーブルスペースコードが ORA-20900 で失敗します。RCU は、以下の方法で OMF と使用できます。

- RCU 12.2.1.0 以降で、-honorOMF コマンドラインパラメータを使用します。
- RCU 12.1.0.3 以降で、複数のステップを使用して生成されたスクリプトを編集します。詳細については、「コマンドラインを使用した複数ステップでの RCU の実行」を参照してください。

オブジェクト権限

Amazon RDS はマネージ型サービスであるため、RDS for Oracle DB インスタンスへのフル SYSDBA アクセス権はありません。ただし、RCU 12c では低い権限のユーザーがサポートされます。ほとんどの場合、リポジトリを作成するにはマスターユーザー権限で十分です。

マスターアカウントは、WITH GRANT OPTION がすでに付与されている権限を直接付与できます。場合によっては、SYS オブジェクト権限を付与しようとすると RCU が ORA-01031 で失敗することがあります。次の例に示すように、rdsadmin_util.grant_sys_object ストアドプロシージャを再試行して実行することができます。

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('GV_$SESSION','MY_DBA','SELECT');
END;
/
```

オブジェクト SCHEMA_VERSION_REGISTRY に対する SYS 権限を付与しようとすると、このオペレーションが ORA-20199: Error in rdsadmin_util.grant_sys_object で失敗することがあります。テーブル SCHEMA_VERSION_REGISTRY$ とビュー SCHEMA_VERSION_REGISTRY をスキーマ所有者の名前 (SYSTEM) で修飾して、オペレーションを再試行できます。または、シノニムを作成することもできます。マスターユーザーとしてログインし、次のステートメントを実行します。

```
CREATE OR REPLACE VIEW SYSTEM.SCHEMA_VERSION_REGISTRY
  AS SELECT * FROM SYSTEM.SCHEMA_VERSION_REGISTRY$;
CREATE OR REPLACE PUBLIC SYNONYM SCHEMA_VERSION_REGISTRY FOR
 SYSTEM.SCHEMA_VERSION_REGISTRY;
CREATE OR REPLACE PUBLIC SYNONYM SCHEMA_VERSION_REGISTRY$ FOR SCHEMA_VERSION_REGISTRY;
```

Enterprise Scheduler Service

RCU を使用して Enterprise Scheduler Service リポジトリを中断すると、RCU が Error: Component drop check failed で失敗することがあります。

Amazon EC2 インスタンスでの Oracle Connection Manager の設定

Oracle Connection Manager (CMAN) は、データベースサーバーまたは他のプロキシサーバーに接続リクエストを転送するプロキシサーバーです。CMAN 使用して、次を設定できます。

アクセスコントロール

ユーザーが指定したクライアントリクエストを除外し、他のクライアントリクエストを受け入れるルールを作成できます。

セッションの多重化

共有サーバーの宛先へのネットワーク接続を介して、複数のクライアントセッションをファネルできます。

通常、CMAN は、データベースサーバーおよびクライアントホストとは別のホストに存在します。詳細については、「Oracle Database ドキュメント」の「Configuring Oracle Connection Manager」(Oracle Connection Manager の構成) を参照してください。

トピック

- CMAN でサポートされているバージョンとライセンスオプション
- CMAN における要件と制限
- CMAN の設定

CMAN でサポートされているバージョンとライセンスオプション

CMAN は、Amazon RDS がサポートする Oracle Database のすべてのバージョンのエンタープライズエディションをサポートしています。詳細については、「RDS for Oracle のリリース」を参照してください。

Oracle Database がインストールされているホストとは別のホストに Oracle Connection Manager をインストールできます。CMAN を実行するホスト用に別のライセンスは必要ありません。

CMAN における要件と制限

Amazon RDS では、フルマネージドエクスペリエンスを提供するため、オペレーティングシステムへのアクセスを制限しています。オペレーティングシステムのアクセスを必要とするデータベースパラメータは変更できません。そのため Amazon RDS では、オペレーティングシステムへの口グインを必要とする CMAN の機能をサポートしていません。

CMAN の設定

CMAN を設定する際は、RDS for Oracle データベースの外部でほとんどの作業を実行します。

トピック

- ステップ 1: RDS for Oracle インスタンスと同じ VPC の Amazon EC2 インスタンスに CMAN を設定する
- ステップ 2: CMAN のデータベースパラメータを設定する
- ステップ 3: DB インスタンスを DB パラメータグループに関連付ける

ステップ 1: RDS for Oracle インスタンスと同じ VPC の Amazon EC2 インスタンスに CMAN を設定する

CMAN の設定方法については、ブログ記事「Configuring and using Oracle Connection Manager on Amazon EC2 for Amazon RDS for Oracle」(Amazon RDS for Oracle 用に Amazon EC2 で Oracle Connection Manager を設定および使用する) の詳細な手順に従ってください。

ステップ 2: CMAN のデータベースパラメータを設定する

Traffic Director Mode やセッションの多重化などの CMAN の機能については、`REMOTE_LISTENER` パラメータをDB パラメータグループ内の CMAN のインスタンスのアドレスに設定します。次のシナリオを考えてみます。

- CMAN のインスタンスが、IP アドレス `10.0.159.100` を持つホストに存在しており、ポート `1521` を使用しているとします。
- データベース `orcla`、`orclb`、および `orclc` は、Oracle DB インスタンスの別の RDS に存在しているとします。

次の表は、`REMOTE_LISTENER` 値の設定方法を示しています。`LOCAL_LISTENER` 値は、Amazon RDS によって自動的に設定されます。

DB インスタンス名	DB インスタンス IP	ローカルリスナーの値 (自動的に設定)	リモートリスナーの値 (ユーザーが設定)
`orcla`	`10.0.159.200`	`( address= (protocol=tcp) (host=10.0.159.200)`	`10.0.159.100:1521`

DB インスタンス名	DB インスタンス IP	ローカルリスナーの値 (自動的に設定)	リモートリスナーの値 (ユーザーが設定)
		(port=1521))	
orclb	10.0.159.30	(address= (protocol=tcp) (host=10.0.159.30) (port=1521))	10.0.159.100:1521
orclc	10.0.159.40	(address= (protocol=tcp) (host=10.0.159.40) (port=1521))	10.0.159.100:1521

ステップ 3: DB インスタンスを DB パラメータグループに関連付ける

ステップ 2: CMAN のデータベースパラメータを設定する で設定したパラメータグループを使用するように DB インスタンスを作成または変更します。詳細については、「Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける」を参照してください。

Amazon RDS での Oracle への Siebel Database のインストール

Amazon RDS を使用して Oracle DB インスタンスで Siebel Database をホストできます。Siebel Database は Siebel Customer Relationship Management (CRM) アプリケーションアーキテクチャの一部です。図については、「Generic Architecture of Siebel Business Application」を参照してください。

次のトピックでは、Amazon RDS の Oracle DB インスタンスで Siebel Database をセットアップする方法について説明します。また、Amazon Web Services を使用して Siebel CRM アプリケーションアーキテクチャに必要なそのほかのコンポーネントをサポートする方法も説明します。

> **Note**
>
> Amazon RDS 上の Oracle に Siebel Database をインストールするには、マスターユーザーアカウントを使用する必要があります。SYSDBA 権限は必要ありません。マスターユーザー権限で十分です。詳細については、「マスターユーザーアカウント権限」を参照してください。

ライセンスとバージョン

Siebel Database を Amazon RDS にインストールするには、お客様ご自身の Oracle Database ライセンスと Siebel ライセンスを使用する必要があります。DB インスタンスクラスと Oracle Database のエディション用の適切な Oracle Database ライセンス (ソフトウェア更新ライセンスおよびサポート) を所持している必要があります。詳細については、「RDS for Oracle のライセンスオプション」を参照してください。

Oracle Database Enterprise Edition は、このシナリオで唯一の Siebel 認定エディションです。Amazon RDS では、Siebel CRM バージョン 15.0 または 16.0 がサポートされています。

Amazon RDS では、データベースのバージョンのアップグレードがサポートされています。詳細については、「DB インスタンスのエンジンバージョンのアップグレード」を参照してください。

スタートする前に

スタートする前に、Amazon VPC が必要です。Amazon RDS DB インスタンスは Siebel Enterprise Server でのみ使用できればよく、パブリックインターネットは必要ないため、Amazon RDS DB インスタンスはより優れたセキュリティを提供するプライベートサブネットでホストされます。Siebel CRM で使用する Amazon VPC を作成する方法については、「Oracle DB インスタンスを作成して接続する」を参照してください。

スタートする前に、Oracle DB インスタンスも必要です。Siebel CRM で使用する Oracle DB インスタンスを作成する方法については、「Amazon RDS DB インスタンスの作成」を参照してください。

Siebel Database のインストールと設定

Oracle DB インスタンスを作成した後、Siebel Database をインストールできます。テーブル所有者および管理者アカウントを作成し、保存手順と機能をインストールして、その後 Siebel Database Configuration Wizard を実行することで、データベースをインストールします。詳細については、「Installing the Siebel Database on the RDBMS」を参照してください。

Siebel Database Configuration Wizard を実行するには、マスターユーザーアカウントを使用する必要があります。SYSDBA 権限は必要ありません。マスターユーザー権限で十分です。詳細については、「マスターユーザーアカウント権限」を参照してください。

Siebel Database によるその他の Amazon RDS 機能の使用

Oracle DB インスタンスを作成すると、Siebel Database のカスタマイズに役立つ追加の Amazon RDS 機能を使用できます。

Oracle Statspack オプションでの統計情報の収集

DB オプショングループのオプションを使用して DB インスタンスに機能を追加できます。Oracle DB インスタンスを作成するときには、デフォルトの DB オプショングループを使用しました。データベースに機能を追加する場合は、DB インスタンスに対して新しいオプショングループを作成できます。

Siebel Database のパフォーマンス統計情報を収集する場合は、Oracle Statspack 機能を追加できます。詳細については、「Oracle Statspack」を参照してください。

オプションの変更は、直ちに適用されるものと、DB インスタンスの次のメンテナンス時間中に適用されるものがあります。詳細については、「オプショングループを使用する」を参照してください。カスタマイズされたオプショングループを作成後、DB インスタンスを修正してアタッチします。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

パラメータによるパフォーマンスのチューニング

DB パラメータグループのパラメータを使用して DB エンジンの設定を管理します。Oracle DB インスタンスを作成するときには、デフォルトの DB パラメータグループを使用しました。データベースの設定をカスタマイズする場合は、DB インスタンスに対して新しいパラメータグループを作成できます。

パラメータの種類によって、パラメータの変更はすぐに適用されるものと、DB インスタンスを手動で再起動した後に適用されるものがあります。詳細については、「Amazon RDS のパラメータグループ」を参照してください。カスタマイズされたパラメータグループを作成後、DB インスタンスを修正してアタッチします。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Oracle DB インスタンスを Siebel CRM 用に最適化するために、特定のパラメータをカスタマイズできます。次の表は、推奨されるパラメータ設定の一部です。Siebel CRM のパフォーマンスチューニングの詳細については、「Siebel CRM Performance Tuning Guide」を参照してください。

パラメータ名	デフォルト値	Siebel CRM の最適なパフォーマンスのガイダンス
_always_semi_join	CHOOSE	OFF
_b_tree_bitmap_plans	TRUE	FALSE
_like_with_bind_as_equality	FALSE	TRUE
_no_or_expansion	FALSE	FALSE
_optimizer_join_sel_sanity_check	TRUE	TRUE
_optimizer_max_permutations	2000	100
_optimizer_sortmer	TRUE	FALSE

パラメータ名	デフォルト値	Siebel CRM の最適なパフォーマンスのガイダンス
`ge_join_enabled`		
`_partition_view_enabled`	`TRUE`	`FALSE`
`open_cursors`	`300`	少なくとも **`2000`**。

スナップショットの作成

Siebel Database を作成した後、Amazon RDS のスナップショット機能を使用してデータベースをコピーできます。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」および「DB インスタンスへの復元」を参照してください。

その他の Siebel CRM コンポーネントのサポート

Siebel Database に加えて、Amazon Web Services を使用して Siebel CRM アプリケーションアーキテクチャの他のコンポーネントをサポートすることもできます。追加の Siebel CRM コンポーネントに対して Amazon AWS により提供されるサポートに関する詳細を次の表に示します。

Siebel CRM コンポーネント	Amazon AWS サポート
Siebel Enterprise (Siebel Server 1 つ以上含む)	Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで Siebel Server をホストできます。Amazon EC2 を使用して必要な分だけ仮想サーバーを起動できます。Amazon EC2 を使用すると、需要の変化に対応して簡単にスケールアップやスケールダウンができます。詳細については、「Amazon EC2 とは」を参照してください。 サーバーを DB インスタンスと同じ VPC に配置し、データベースへのアクセスに VPC のセキュリティグループを使用できます。詳細については、「VPC 内の DB インスタンスの使用」を参照してください。

Siebel CRM コンポーネント	Amazon AWS サポート
ウェブサーバー (Siebel Web Server Extension を含む)	複数の EC2 インスタンスに複数のウェブサーバーをインストールできます。その後、Elastic Load Balancing を使用して着信トラフィックをインスタンスに分散できます。詳細については、「Elastic Load Balancing とは」を参照してください。
Siebel Gateway Name Server	EC2 インスタンスで Siebel Gateway Name Server をホストできます。その後、サーバーを DB インスタンスと同じ VPC に配置し、データベースへのアクセスに VPC のセキュリティグループを使用できます。詳細については、「VPC 内の DB インスタンスの使用」を参照してください。

Oracle データベースエンジンのリリースノート

Amazon RDS for Oracle DB インスタンスは、更新によって常に最新の状態に保たれます。更新を適用することで、Oracle と Amazon の両方でテスト済みのバージョンのデータベースソフトウェアが、DB インスタンスで確実に実行されます。個々の RDS for Oracle DB インスタンスへのワンオフパッチの適用はサポートされていません。

新しい DB インスタンスを作成するときは、現在サポートされているいずれかの Oracle Database バージョンを指定できます。メジャーバージョン (Oracle Database 19c など) と、指定したメジャーバージョンでサポートされている任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、サポートされているいずれかのバージョン (通常最新のバージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合、Amazon RDS では、指定されたメジャーバージョンの最新リリースにデフォルトで設定されます。サポートされているバージョンのリストと、新しく作成された DB インスタンスのデフォルトを表示するには、`describe-db-engine-versions` AWS CLI コマンドを使用します。

Amazon RDS でサポートされている Oracle Database のバージョンの詳細については、「Amazon RDS for Oracle リリースノート」を参照してください。

Amazon RDS for PostgreSQL

Amazon RDS では、PostgreSQL の複数のバージョンを実行する DB インスタンスがサポートされています。利用可能なバージョンのリストについては、「利用可能な PostgreSQL データベースのバージョン」を参照してください。

DB インスタンス、DB スナップショット、ポイントインタイムの復元とバックアップを作成できます。PostgreSQL を実行する DB インスタンスでは、マルチ AZ 配置、リードレプリカ、プロビジョンド IOPS がサポートされています。これらのインスタンスは、仮想プライベートクラウド (VPC) 内で作成できます。PostgreSQL を実行する DB インスタンスには Secure Socket Layer (SSL) でも接続できます。

DB インスタンスを作成する前に、必ず「Amazon RDS 環境のセットアップ」の手順を完了してください。

スタンダード的な SQL クライアントアプリケーションを使用して、クライアントコンピュータからインスタンスに対してコマンドを実行できます。例えば、pgAdmin (PostgreSQL 用に普及しているオープンソースの管理および開発ツール) や psql (PostgreSQL インストールに含まれるコマンドラインユーティリティ) などのアプリケーションを使用できます。マネージド型サービスエクスペリエンスを提供するために、Amazon RDS では DB インスタンスへのホストアクセスが許可されません。また、アドバンストの特権を必要とする特定のシステムの手順やテーブルへのアクセスも制限されいます。Amazon RDS は、任意のスタンダード SQL クライアントアプリケーションによる、DB インスタンス上のデータベースへのアクセスがサポートされています。Amazon RDS では、Telnet またはセキュアシェル (SSH) を使用しての、DB インスタンスへのダイレクトホストアクセスは許可されません。

Amazon RDS for PostgreSQL は、多くの業界スタンダードに準拠しています。例えば、Amazon RDS for PostgreSQL データベースを使用して、HIPAA 準拠アプリケーションをビルドしたり、医療関連の情報を保存したりできます。これには、AWS との間で締結された事業提携契約 (BAA) に基づく保護されるべき医療情報 (PHI) の保存が含まれます。また、Amazon RDS for PostgreSQL は、Federal Risk and Authorization Management Program (FedRAMP) のセキュリティ要件を満たしています。Amazon RDS for PostgreSQL は、AWS GovCloud (US) リージョンにおいて、FedRAMP Joint Authorization Board (JAB) の Provisional Authority to Operate (P-ATO) として、FedRAMP High ベースラインの認証を受けています。サポートされるコンプライアンススタンダードの詳細については、AWS クラウドコンプライアンスを参照してください。

PostgreSQL データを DB インスタンスにインポートするには、「Amazon RDS の PostgreSQL にデータをインポートする」セクションの説明に従ってください。

> **Important**
>
> RDS for PostgreSQL DB インスタンスで問題が発生した場合、AWS サポートエージェントはデータベースのヘルスに関する詳細情報を必要とする場合があります。目標は、AWS サポートが必要な情報をできるだけ早く取得できるようにすることです。
> PG コレクターを使用すると、統合された HTML ファイルに貴重なデータベース情報を収集できます。PG コレクター、その実行方法、HTML レポートのダウンロード方法の詳細については、「PG Collector」を参照してください。
> 正常に完了すると、特に指定しない限り、スクリプトは読み取り可能な HTML 形式で出力を返します。スクリプトは、ビジネスを危険にさらす可能性があるデータやセキュリティの詳細を HTML から除外するように設計されています。また、データベースや環境を変更することはありません。ただし、HTML 内に共有したくない情報がある場合は、HTML をアップロードする前に該当の情報を削除してください。HTML が許容できる内容となったら、サポートケースの [ケースの詳細] で [添付ファイル] セクションを使用してアップロードします。

トピック

- Amazon RDS for PostgreSQL の一般的な管理タスク
- データベースプレビュー環境の使用
- 利用可能な PostgreSQL データベースのバージョン
- RDS for PostgreSQL 増分リリースプロセスについて
- サポートされている PostgreSQL 拡張機能バージョン
- Amazon RDS for PostgreSQL でサポートされている PostgreSQL の機能を使用する
- PostgreSQL データベースエンジンを実行する DB インスタンスへの接続
- SSL/TLS を使用して RDS for PostgreSQL への接続を保護する
- Amazon RDS for PostgreSQL で Kerberos 認証を使用する
- アウトバウンドネットワークアクセスでカスタム DNS サーバーを使用する
- RDS for PostgreSQL DB エンジンのアップグレード
- PostgreSQL DB スナップショットエンジンのバージョンのアップグレード
- Amazon RDS for PostgreSQL でのリードレプリカの使用
- Amazon RDS Optimized Reads による RDS for PostgreSQL のクエリパフォーマンスの向上
- Amazon RDS の PostgreSQL にデータをインポートする

- RDS for PostgreSQL DB インスタンスから Amazon S3 へのデータのエクスポート
- RDS for PostgreSQL DB インスタンスから AWS Lambda 関数を呼び出す
- Amazon RDS for PostgreSQL の一般的な DBA タスク
- RDS for PostgreSQL の待機イベントでのチューニング
- Amazon DevOps Guru のプロアクティブインサイトによる RDS for PostgreSQL のチューニング
- Amazon RDS for PostgreSQL で PostgreSQL 拡張機能を使用する
- Amazon RDS for PostgreSQL でサポートされている外部データラッパーを使用する
- Trusted Language Extensions for PostgreSQL を使用した操作

Amazon RDS for PostgreSQL の一般的な管理タスク

以下に示しているのは、Amazon RDS for PostgreSQL DB インスタンスで実行する一般的な管理タスクと、各タスクの関連ドキュメントへのリンクです。

タスク領域	関連資料
Amazon RDS を初めてセットアップして使う DB インスタンスを作成する前に、いくつかの前提条件を満たしていることを確認してください。例えば、DB インスタンスが作成されると、アクセスを禁止するファイアウォールがデフォルトで設定されます。したがって、DB インスタンスにアクセスするために、IP アドレスとネットワーク設定が正しく指定されたセキュリティグループを作成する必要があります。	Amazon RDS 環境のセットアップ
Amazon RDS DB インスタンスを理解する 本稼働用に DB インスタンスを作成する場合、インスタンスクラス、ストレージタイプ、およびプロビジョンド IOPS が Amazon RDS でどのように機能するか理解する必要があります。	DB インスタンスクラス Amazon RDS ストレージタイプ プロビジョンド IOPS SSD ストレージ
利用可能な PostgreSQL バージョンの検索	利用可能な PostgreSQL データベースのバージョン

タスク領域	関連資料
Amazon RDS は、PostgreSQL の複数のバージョンをサポートします。	
高可用性およびフェイルオーバーサポートのセットアップ 本稼働 DB インスタンスは、マルチ AZ 配置を使用する必要があります。マルチ AZ 配置は、DB インスタンスの拡張された可用性、データ堅牢性、および耐障害性を提供します。	Amazon RDS でのマルチ AZ 配置の設定と管理
Amazon Virtual Private Cloud (VPC) ネットワークについて AWS アカウントにデフォルト VPC がある場合、DB インスタンスがデフォルト VPC 内に自動的に作成されます。場合によっては、アカウントにデフォルトの VPC がないため、VPC に DB インスタンスが必要な場合もあります。このような場合、DB インスタンスを作成する前に VPC とサブネットグループを作成します。	VPC 内の DB インスタンスの使用
Amazon RDS PostgreSQL にデータをインポートする Amazon RDS の PostgreSQL DB インスタンスにデータをインポートするには、さまざまなツールを使用できます。	Amazon RDS の PostgreSQL にデータをインポートする
読み取り専用リードレプリカ (プライマリおよびスタンバイ) のセットアップ RDS for PostgreSQL では、同じ AWS リージョン、および、プライマリインスタンスとは別の AWS リージョンの両方で、リードレプリカがサポートされます。	DB インスタンスのリードレプリカの操作 Amazon RDS for PostgreSQL でのリードレプリカの使用 別の でのリードレプリカの作成AWS リージョン

タスク領域	関連資料
セキュリティグループの理解 デフォルトでは、DB インスタンスが作成されると、アクセスを禁止するファイアウォールが設定されます。そのファイアウォールを介してアクセスできるようにするには、DB インスタンスをホストする VPC に関連付けられている VPC セキュリティグループのインバウンドルールを編集します。	セキュリティグループによるアクセス制御
パラメータグループおよび機能のセットアップ DB インスタンスのデフォルトパラメータを変更するには、カスタム DB パラメータグループを作成し、設定をそのパラメータグループに変更します。DB インスタンスを作成する前にこの操作を行うと、インスタンスの作成時にカスタム DB パラメータグループを選択できます。	Amazon RDS のパラメータグループ
PostgreSQL DB インスタンスへの接続 セキュリティグループを作成し、それを DB インスタンスに関連付けると、`psql` や `pgAdmin` などの標準的な SQL クライアントアプリケーションを使用して DB インスタンスに接続できます。	PostgreSQL データベースエンジンを実行する DB インスタンスへの接続 PostgreSQL DB インスタンスで SSL を使用する
DB インスタンスのバックアップと復元 バックアップが自動的に作成されるように DB インスタンスを設定したり、スナップショットを手動で作成したりできます。そうすることで後で、そのバックアップまたはスナップショットからインスタンスを復元できます。	データのバックアップ、復元、エクスポート
DB インスタンスのアクティビティとパフォーマンスのモニタリング CloudWatch Amazon RDS メトリクス、イベント、および拡張モニタリングを使用することで、PostgreSQL DB インスタンスをモニタリングできます。	Amazon RDS コンソールでのメトリクスの表示 Amazon RDS イベントの表示

タスク領域	関連資料
PostgreSQL データベースバージョンのアップグレード PostgreSQL DB インスタンスのメジャーバージョンとマイナーバージョンの両方をアップグレードできます。	RDS for PostgreSQL DB エンジンのアップグレード RDS for PostgreSQL のメジャーバージョンアップグレードの選択
ログファイルの操作 PostgreSQL DB インスタンスのログファイルにアクセスできます。	RDS for PostgreSQL データベースログファイル
PostgreSQL DB インスタンスに関するベストプラクティスの理解 Amazon RDS での PostgreSQL の使用に関するいくつかのベストプラクティスが見つかります。	PostgreSQL を使用するためのベストプラクティス

RDS for PostgreSQL の重要な機能の理解と使用に役立つ、このガイドの他のセクションのリストは次のとおりです。

- PostgreSQL のロールとアクセス権限について
- PostgreSQL データベースへのユーザーアクセスのコントロール
- RDS for PostgreSQL DB インスタンスでのパラメータの使用
- RDS for PostgreSQL でサポートされているログ記録メカニズムについて
- Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用
- アウトバウンドネットワークアクセスでカスタム DNS サーバーを使用する

データベースプレビュー環境の使用

PostgreSQL コミュニティは、ベータバージョンを含め、新しい PostgreSQL バージョンとエクステンションを絶えずリリースしています。これにより、PostgreSQL ユーザーは新しい PostgreSQL バージョンを早期に試すことができます。PostgreSQL コミュニティのベータリリースプロセスの詳細については、PostgreSQL ドキュメントの「ベータ情報」を参照してください。同様に、Amazon RDS では特定の PostgreSQL ベータバージョンをプレビューリリースとして提供するようにしています。これにより、プレビューバージョンを使用して DB インスタンスを作成し、その機能をデータベースプレビュー環境でテストできます。

データベースプレビュー環境の RDS for PostgreSQL DB インスタンスは、機能的に他の RDS for PostgreSQL インスタンスに似ています。ただし、プレビューバージョンは本稼働に使用できません。

次の重要な制約事項に留意してください。

- すべての DB インスタンスは、作成から 60 日後にバックアップおよびスナップショットとともに削除されます。
- DB インスタンスは、Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) でのみ作成できます。
- 汎用 SSD およびプロビジョンド IOPS SSD ストレージのみを使用できます。
- DB インスタンスに関して AWS サポートからヘルプを受けることはできません。代わりに、AWS マネージド Q&A コミュニティ、AWS re:Post に質問を投稿できます。
- DB インスタンスのスナップショットを本稼働環境にコピーすることはできません。

プレビューでは、以下のオプションがサポートされています。

- DB インスタンスは、M6i、R6i、M6g、M5、T3、R6g、および R5 インスタンスタイプのみで作成できます。RDS インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。
- シングル AZ 配置とマルチ AZ 配置の両方を使用できます。
- スタンダードの PostgreSQL ダンプおよびロード機能を使用して、データベースをデータベースプレビュー環境にエクスポートしたり、データベースプレビュー環境にインポートしたりできます。

トピック

- データベースプレビュー環境でサポートされない機能

- データベースプレビュー環境の PostgreSQL バージョン 17
- データベースプレビュー環境での新しい DB インスタンスの作成

データベースプレビュー環境でサポートされない機能

以下の機能は、データベースプレビュー環境で使用できません。

- クロスリージョンスナップショットのコピー
- クロスリージョンリードレプリカ

データベースプレビュー環境の PostgreSQL バージョン 17

> **Note**
>
> これは Amazon RDS PostgreSQL バージョン 17 のプレビュードキュメントです。このドキュメントは変更される可能性があります。

PostgreSQL バージョン 17.0 が Amazon RDS データベースプレビュー環境で利用可能になりました。PostgreSQL バージョン 17 には、次の PostgreSQL ドキュメントに記載されているいくつかの改善点が含まれています:「PostgreSQL 17 がリリースされました」。

データベースプレビュー環境の詳細については、「the section called “データベースプレビュー環境の使用”」を参照してください。コンソールからプレビュー環境にアクセスするには、https://console.aws.amazon.com/rds-preview/ を選択します。

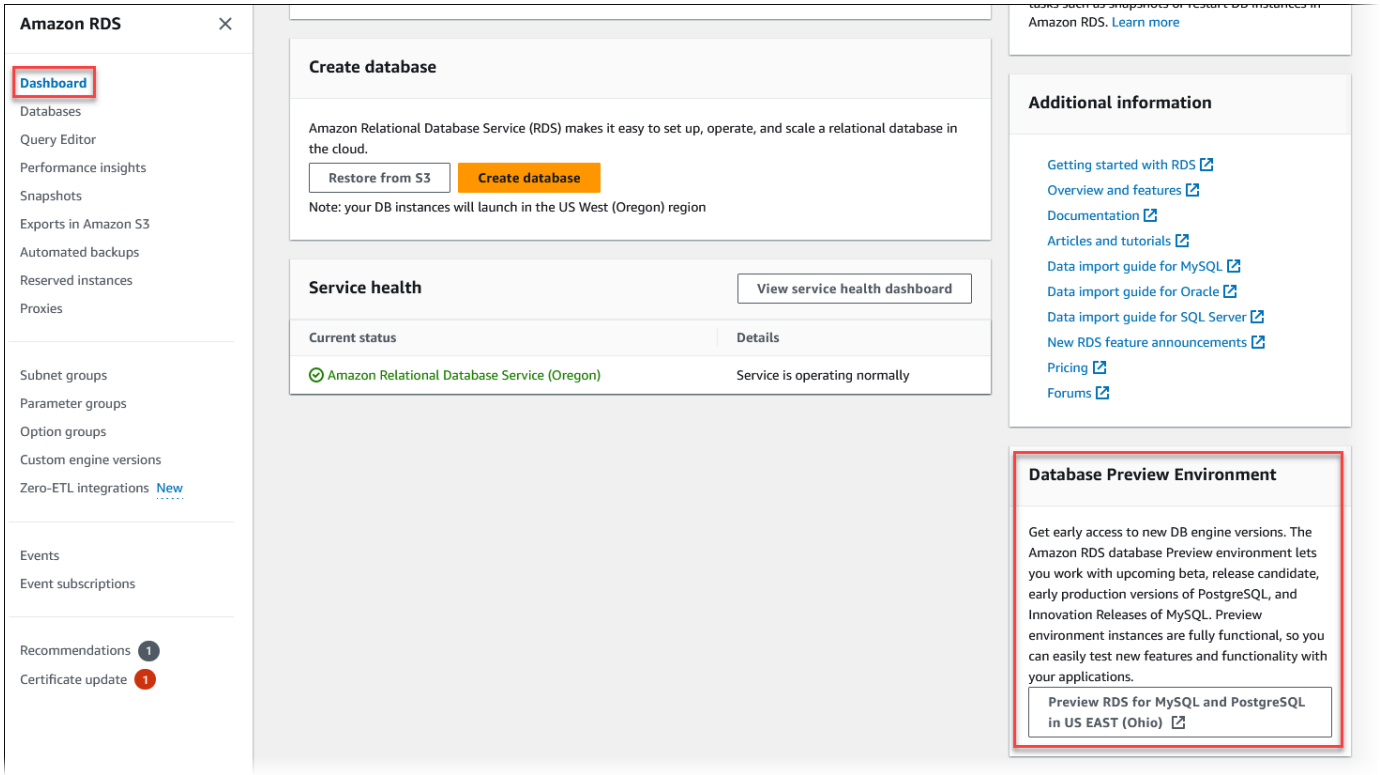
データベースプレビュー環境での新しい DB インスタンスの作成

プレビュー環境で DB インスタンスを作成するには、次の手順を使用します。

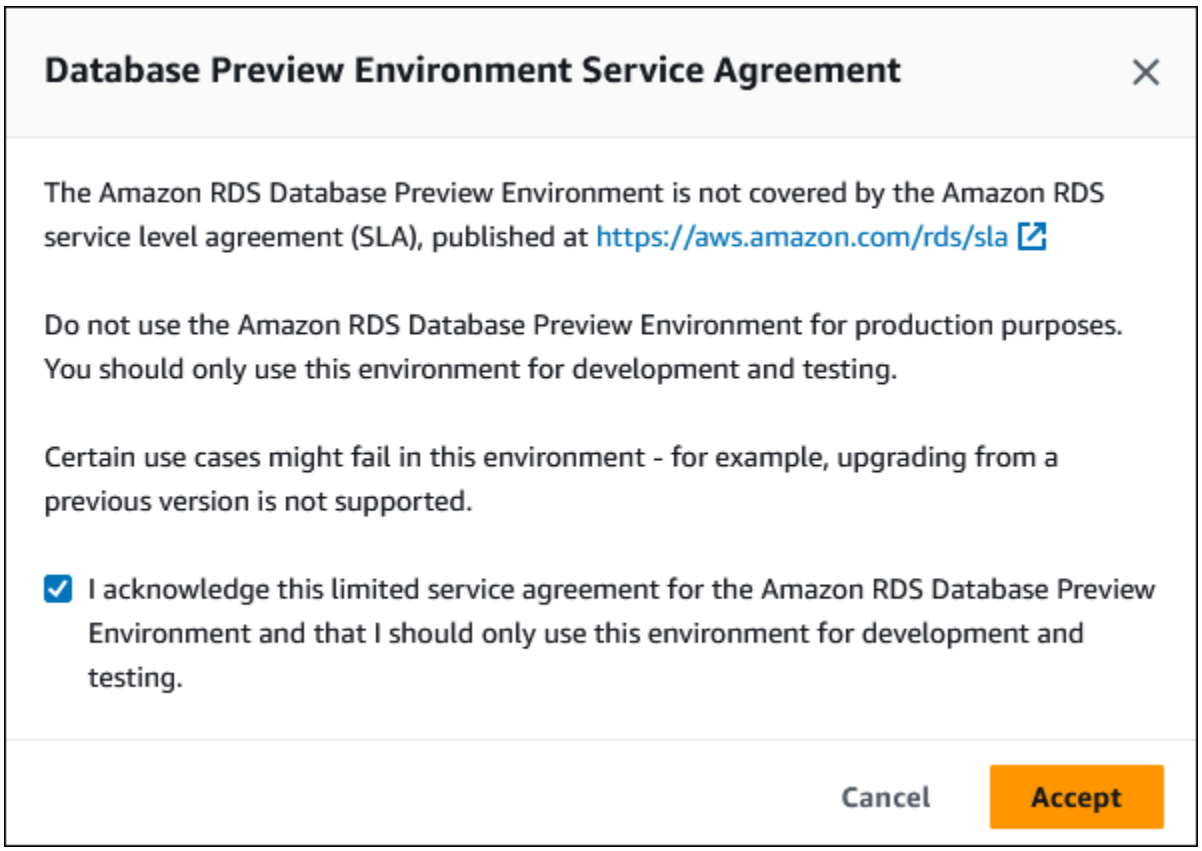
データベースプレビュー環境で新しい DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[ダッシュボード] を選択します。

3. [Dashboard] (ダッシュボード) ページで、次の図に示すように、[Dashboard] (ダッシュボード) ページの [Database Preview Environment] (データベースプレビュー環境) セクションを見つけます。



また、[データベースプレビュー環境] に直接移動することもできます。先に進む前に、制限事項を確認して同意する必要があります。



4. RDS for PostgreSQL DB インスタンスを作成するには、任意の Amazon RDS DB インスタンスを作成する場合と同じプロセスに従います。詳細については、DB インスタンスの作成 下の コンソール 手順を参照してください。

RDS API または AWS CLI を使用してデータベースプレビュー環境でインスタンスを作成するには、次のエンドポイントを使用します。

```
rds-preview.us-east-2.amazonaws.com
```

利用可能な PostgreSQL データベースのバージョン

Amazon RDS では、PostgreSQL の複数のエディションを実行する DB インスタンスがサポートされています。新しい DB インスタンスを作成する際、現在利用可能な PostgreSQL バージョンであればどれでも指定できます。メジャーバージョン (PostgreSQL 14 など) と、指定したメジャーバージョンで利用可能な任意のマイナーバージョンを指定できます。バージョンを指定しない場合、Amazon RDS では、利用可能なバージョン (通常は最新バージョン) がデフォルトで設定されます。マイナーバージョンではなく、メジャーバージョンを指定した場合は、Amazon RDS では、お客様が指定したメジャーバージョンの最新リリースにデフォルトで設定されます。

利用可能なバージョンのリストと、新しく作成された DB インスタンスのデフォルト設定を表示するには、AWS CLI の `describe-db-engine-versions` コマンドを使用します。例えば、デフォルトの PostgreSQL エンジンのバージョンを表示するには、次のコマンドを使用します。

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

Amazon RDS でサポートされている PostgreSQL バージョンの詳細については、Amazon RDS for PostgreSQL リリースノートを参照してください。describe-db-major-engine-versions AWS CLI コマンドを実行するか、DescribeDBMajorEngineVersions RDS API オペレーションを使用して、メジャーエンジンバージョンのサポート日に関する情報を表示することもできます。

RDS 標準サポート終了日より前に新しいメジャーエンジンバージョンに手動でアップグレードする準備ができていない場合、Amazon RDS は RDS 標準サポート終了日以降にデータベースを Amazon RDS 延長サポートに自動的に登録します。その後、RDS for PostgreSQL バージョン 11 以降を引き続き実行できます。詳細については、「Amazon RDS の Amazon RDS 延長サポート」および「Amazon RDS の料金」を参照してください。

Amazon RDS for PostgreSQL の非推奨バージョン

以下の廃止バージョンに注意してください。

- RDS for PostgreSQL 10 は 2023 年 2 月に廃止されました。
- RDS for PostgreSQL 9.6 は 2022 年 3 月に廃止されました。
- RDS for PostgreSQL 9.5 は、2021 年 3 月に廃止されました。

RDS for PostgreSQL の非推奨ポリシーの詳細については、「Amazon RDS のよくある質問」を参照してください。PostgreSQL のバージョンの詳細については、PostgreSQL のドキュメントの「Versioning Policy」(バージョニングポリシー) を参照してください。

RDS for PostgreSQL 増分リリースプロセスについて

RDS for PostgreSQL は、マイナーバージョンの互換性を維持しながら、増分リリースを通じてセキュリティ修正、パフォーマンスの向上、新機能を提供します。これらのリリースには、R1、R2、R3 などのラベルが付けられます。

リリースバージョンの命名規則

- R1 はマイナーバージョンの初期リリースです。これには、新機能、拡張機能、または既存の拡張機能へのアップグレードが含まれることがあります。
- 以降のリリースバージョン (R2、R3 以降) には以下が含まれます。
  - セキュリティ更新
  - フォーマンスの向上
  - バグ修正
  - 拡張機能の更新

RDS for PostgreSQL 増分リリースプロセスの利点

増分リリースプロセスには、次の利点があります。

- 新しい PostgreSQL コミュニティリリースを迅速に導入し、以降のリリースで RDS 固有の機能強化を個別に管理します。これにより、リリースプロセスが合理化され、重要な更新を迅速に配信できます。
- PostgreSQL マイナーバージョンとの互換性を維持しながら、バグ修正、新機能、セキュリティ更新、拡張機能の更新にアクセスできます。

リリース更新の管理

Amazon RDS は、AWS マネジメントコンソールで保留中のメンテナンスアクションを通じて、新しい増分リリースについて通知します。データベースは、次のいずれかの方法を使用して更新できます。

- スケジュールされたメンテナンス期間中に自動更新を有効にする。
- 保留中のメンテナンスアクションを通じて更新を手動で適用する。

- ダウンタイムを最小限に抑えるには、物理レプリケーションでブルー/グリーンデプロイを使用します。詳細については、「ブルー/グリーンデプロイ RDS for PostgreSQL におけるマイナーバージョンアップグレードをサポート」を参照してください。

データベースを更新する前に次の重要な点を考慮してください。

- 物理レプリケーションでブルー/グリーンデプロイを使用しない場合、更新にはデータベースの再起動が必要です。
- 一部の増分リリース、特にセキュリティ修正を含むリリースは必須です。

Amazon RDS DB インスタンスの更新の詳細については、「PostgreSQL 信頼できるエクステンション」および「apply-pending-maintenance-action」を参照してください。

サポートされている PostgreSQL 拡張機能バージョン

RDS for PostgreSQL は、多くの PostgreSQL エクステンションをサポートしています。PostgreSQL コミュニティでは、これらをモジュールと呼ぶことがあります。エクステンションは、PostgreSQL エンジンが提供する機能を拡張します。Amazon RDS がサポートするエクステンションのリストは、該当する PostgreSQL バージョンのデフォルトの DB パラメータグループで確認できます。また、次の例に示すように、psql で rds.extensions パラメータを表示することで、最新のエクステンションのリストを確認することもできます。

```
SHOW rds.extensions;
```

> **Note**
>
> マイナーバージョンのリリースで追加されたパラメータは、rds.extensions で psql パラメータを使用する際に正しく表示されない場合があります。

RDS for PostgreSQL 13 以降、rds_superuser 以外のデータベースユーザーがインストールできる拡張機能があります。これらは、信頼できる拡張機能として知られています。詳細についてはPostgreSQL 信頼できるエクステンションを参照してください。

RDS for PostgreSQL の特定のバージョンでは、rds.allowed_extensions パラメータに対応しています。このパラメータにより、rds_superuser は RDS for PostgreSQL DB インスタンスにインストールできる拡張機能を制限します。(詳しくは、「PostgreSQL エクステンションのインストールを制限する」を参照してください。)

利用できる各 RDS for PostgreSQL バージョンでサポートされている PostgreSQL 拡張機能のリストについては、Amazon RDS for PostgreSQL リリースノートの「Amazon RDS でサポートされる PostgreSQL の拡張機能」を参照してください。

PostgreSQL エクステンションのインストールを制限する

PostgreSQL DB インスタンスにインストールできるエクステンションを制限できます。デフォルトでは、このパラメータは設定されていないため、ユーザーに権限がある場合は、サポートされている任意の拡張機能を追加できます。そのためには、rds.allowed_extensions パラメータをカンマで区切った拡張子名の文字列に設定します。このパラメータに拡張機能のリストを追加すると、RDS for PostgreSQL DB インスタンスで使用できる拡張機能が明示的に特定できます。その

後、PostgreSQL DB インスタンスにインストールできるのは、これらのエクステンションだけです。

rds.allowed_extensions パラメータのデフォルトの文字列は '*' です。これは、そのエンジンのバージョンで使用できるエクステンションは何でもインストールできることを意味します。動的パラメータであるため、rds.allowed_extensions パラメータを変更しても、データベースを再起動する必要はありません。

rds.allowed_extensions パラメータを使用するには、PostgreSQL DB インスタンスエンジンは次のバージョンのいずれかである必要があります。

- すべての PostgreSQL 16 バージョン
- PostgreSQL 15 以降のすべてのバージョン
- PostgreSQL 14 以降のすべてのバージョン
- PostgreSQL 13.3 以降のマイナーバージョン
- PostgreSQL 12.7 以降のマイナーバージョン

どのエクステンションのインストールが許可されているかを確認するには、次の psql コマンドを使用します。

```
postgres=> SHOW rds.allowed_extensions;
 rds.allowed_extensions
------------------------
 *
```

エクステンションが rds.allowed_extensions パラメータのリストから除外される前にインストールされていた場合でも、エクステンションは正常に使用でき、ALTER EXTENSION や DROP EXTENSION などのコマンドは引き続き機能します。ただし、エクステンションが制限されると、制限されたエクステンションの CREATE EXTENSION コマンドは失敗します。

CREATE EXTENSION CASCADE とのエクステンションの依存関係のインストールも制限されています。エクステンションとその依存関係は、rds.allowed_extensions で指定する必要があります。拡張依存のインストールが失敗すると、CREATE EXTENSION CASCADE 文全体が失敗します。

rds.allowed_extensions パラメータにエクステンションが含まれていない場合、インストールしようとすると、次のようなエラーが表示されます。

```
ERROR: permission denied to create extension "extension-name"
```

```
HINT: This extension is not specified in "rds.allowed_extensions".
```

PostgreSQL 信頼できるエクステンション

ほとんどの PostgreSQL エクステンションをインストールするには、rds_superuser 権限が必要です。PostgreSQL 13 では、信頼できるエクステンションが導入されました。これにより、一般ユーザーに rds_superuser 特権を付与する必要が少なくなります。この機能を使用すると、ユーザーは、CREATE ロールを要求する代わりに、現在のデータベースに対する rds_superuser 権限を持っている場合、多くのエクステンションをインストールできます。詳細については、PostgreSQL ドキュメントの SQL CREATE EXTENSION コマンドを参照してください。

以下に、現在のデータベースに対する CREATE 権限を持ち、rds_superuser ロールを必要としないユーザーがインストールできるエクステンションを示します。

- bool_plperl
- btree_gin
- btree_gist
- citext
- cube
- dict_int
- fuzzystrmatch
- hstore
- intarray
- isn
- jsonb_plperl
- ltree
- pg_trgm
- pgcrypto
- plperl
- plpgsql
- pltcl
- tablefunc
- tsm_system_rows

- tsm_system_time
- unaccent
- uuid-ossp

利用できる各 RDS for PostgreSQL バージョンでサポートされている PostgreSQL 拡張機能のリストについては、Amazon RDS for PostgreSQL リリースノートの「Amazon RDS でサポートされる PostgreSQL の拡張機能」を参照してください。

Amazon RDS for PostgreSQL でサポートされている PostgreSQL の機能を使用する

Amazon RDS for PostgreSQL は、PostgreSQL の代表的な機能の多くをサポートしています。例えば、PostgreSQL には、データベースの定期的なメンテナンスを実行する自動バキューム機能があります。autovacuum 機能は、デフォルトでアクティブになっています。この機能をオフにすることはできますが、オンにしておくことを強くお勧めします。この機能を理解し、それを正しく動作させるために何ができるかを把握することは、どの DBA にとっても基本的なタスクです。自動バキュームの詳細については、「Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用」を参照してください。その他の一般的な DBA タスクの詳細については、「Amazon RDS for PostgreSQL の一般的な DBA タスク」を参照してください。

RDS for PostgreSQL では、DB インスタンスに重要な機能性を追加する拡張機能もサポートされています。例えば、PostGIS 拡張機能を使用して空間データを操作したり、pg_cron 拡張機能を使用してインスタンス内からメンテナンスをスケジュールしたりできます。PostgreSQL 拡張機能の詳細については、「Amazon RDS for PostgreSQL で PostgreSQL 拡張機能を使用する」を参照してください。

外部データラッパーは、RDS for PostgreSQL DB インスタンスが他の商用データベースまたはデータ型と連携できるように設計された特定のタイプの拡張機能です。RDS for PostgreSQL でサポートされている外部データラッパーの詳細については、「Amazon RDS for PostgreSQL でサポートされている外部データラッパーを使用する」を参照してください。

RDS for PostgreSQL でサポートされているその他の機能については、次で説明します。

トピック

- RDS for PostgreSQL でのカスタムデータ型および列挙型
- RDS for PostgreSQL のイベントトリガー
- RDS for PostgreSQL の ヒュージページ
- Amazon RDS for PostgreSQL の論理レプリケーションの実行
- 論理レプリケーション接続の IAM 認証の設定
- stats_temp_directory の RAM ディスク
- RDS for PostgreSQL のテーブルスペース
- EBCDIC やその他のメインフレーム移行のための RDS for PostgreSQL 照合順序
- RDS for PostgreSQL の論理スロット同期の管理

RDS for PostgreSQL でのカスタムデータ型および列挙型

PostgreSQL では、カスタムデータ型の作成と列挙型の操作がサポートされています。列挙型とその他のデータ型の作成および操作の詳細については、「PostgreSQL のドキュメント」の「列挙型」を参照してください。

次は、列挙型として型を作成し、テーブルに値を挿入する例です。

```
CREATE TYPE rainbow AS ENUM ('red', 'orange', 'yellow', 'green', 'blue', 'purple');
CREATE TYPE
CREATE TABLE t1 (colors rainbow);
CREATE TABLE
INSERT INTO t1 VALUES ('red'), ( 'orange');
INSERT 0 2
SELECT * from t1;
colors
--------
red
orange
(2 rows)
postgres=> ALTER TYPE rainbow RENAME VALUE 'red' TO 'crimson';
ALTER TYPE
postgres=> SELECT * from t1;
colors
---------
crimson
orange
(2 rows)
```

RDS for PostgreSQL のイベントトリガー

現在すべての PostgreSQL バージョンでイベントトリガーをサポートしているため、RDS for PostgreSQL のすべての利用可能なバージョンも、同様にイベントトリガーをサポートしています。メインユーザーアカウント (デフォルトでは postgres) を使用して、イベントトリガーを作成、変更、名前変更、および削除できます。イベントトリガーは DB インスタンスレベルであるため、インスタンスのすべてのデータベースに適用できます。

例えば、次のコードは、各データ定義言語 (DDL) コマンドの最後に現在のユーザーを表示するイベントトリガーを作成します。

```
CREATE OR REPLACE FUNCTION raise_notice_func()
```

```
    RETURNS event_trigger
    LANGUAGE plpgsql AS
$$
BEGIN
    RAISE NOTICE 'In trigger function: %', current_user;
END;
$$;

CREATE EVENT TRIGGER event_trigger_1
    ON ddl_command_end
EXECUTE PROCEDURE raise_notice_func();
```

PostgreSQL イベントトリガーの詳細については、PostgreSQL ドキュメントの「Event Triggers」を参照してください。

Amazon RDS で PostgreSQL イベントトリガーを使用する場合、いくつかの制限があります。これには以下が含まれます。

- リードレプリカでイベントトリガーを作成することはできません。ただし、イベントトリガーはリードレプリカソースで作成できます。作成したイベントトリガーは、リードレプリカにコピーされます。リードレプリカのイベントトリガーは、ソースから変更がプッシュされたときにリードレプリカでは起動しません。ただし、リードレプリカが昇格されると、データベースオペレーションが発生したときに既存のイベントトリガーが起動します。
- イベントトリガーを使用する PostgreSQL DB インスタンスへのメジャーバージョンアップグレードを実行する場合は、インスタンスのアップグレード前に必ずイベントトリガーを削除してください。

RDS for PostgreSQL の ヒュージページ

Huge pages はメモリ管理機能です。DB インスタンスが、共有バッファで使用されるような、大きく連続したメモリチャンクで動作しているときのオーバーヘッドを軽減します。この PostgreSQL の機能は、現在の PostgreSQL バージョンで利用可能なすべての RDS でサポートされています。mmapまたはSYSV共有メモリへの呼び出しを使用して、アプリケーションに巨大なページを割り当てます。RDS for PostgreSQL では、4 KB と 2 MB の両方のページサイズがサポートされています。

huge_pages パラメータの値を変更することで、Huge pages のオンとオフを切り替えることができます。この機能は、micro、small、medium 以外のすべての DB インスタンスクラスで、デフォルトでオンになっています。

RDS for PostgreSQL では、利用可能な共有メモリに基づき、ヒュージページを使用します。共有メモリの制約のために DB インスタンスが巨大なページを使用できない場合、Amazon RDS は DB インスタンスの起動を防ぎます。この場合、Amazon RDS により、DB インスタンスのステータスが互換性のないパラメータ状態に設定されます。これが起こると、huge_pagesパラメータをoffに設定して Amazon RDS で DB インスタンスの起動を許可します。

shared_buffers パラメータは huge ページを使用するために必要な共有メモリプールの設定に重要です。shared_buffersパラメータのデフォルト値は、データベースパラメータマクロを使用します。このマクロは、DB インスタンスのメモリで使用できる 8 KBのページの合計のパーセント割合を設定します。巨大なページを使用する場合、それらのページは巨大なページと一緒に配置されます。共有メモリパラメータで DB インスタンスの 90 パーセントを超えるメモリを使用するように設定すると、Amazon RDS は DB インスタンスを互換性のないパラメータ状態に設定します。

PostgreSQL のメモリ管理については、PostgreSQL のドキュメントの「Resource Consumption」(資源の消費) を参照してください。

Amazon RDS for PostgreSQL の論理レプリケーションの実行

バージョン 10.4 以降、RDS for PostgreSQL は、PostgreSQL 10 で導入されたパブリケーションおよびサブスクリプション SQL 構文をサポートしています。詳細については、「PostgreSQL のドキュメント」の「論理レプリケーション」を参照してください。

> **Note**
>
> PostgreSQL 10 で導入されたネイティブの PostgreSQL 論理レプリケーション機能に加えて、PostgreSQL 用 RDS は pglogical 拡張機能もサポートしています。詳細については、「pglogical を使用してインスタンス間でデータを同期する」を参照してください。

RDS for PostgreSQL DB インスタンスに対する論理レプリケーションの設定については、次で説明します。

トピック

- 論理レプリケーションと論理デコードについて
- 論理的なレプリケーションスロットの使用
- 論理レプリケーションを使用したテーブルレベルのデータのレプリケーション

論理レプリケーションと論理デコードについて

RDS for PostgreSQL は、PostgreSQL の論理レプリケーションスロットを使用した、ログ先行書き込み (WAL) 変更のストリーミングをサポートしています。また、ロジカルデコーディングの使用もサポートしています。インスタンスで論理的なレプリケーションスロットをセットアップし、それらのスロットを通じてデータベースの変更を pg_recvlogical などのクライアントにストリーミングできます。データベースレベルで論理レプリケーションスロットを作成すると、1 つのデータベースへのレプリケーション接続がサポートされます。

PostgreSQL 論理レプリケーション用の最も一般的なクライアントは、AWS Database Migration Service、または Amazon EC2 インスタンスのカスタム管理ホストです。論理レプリケーションスロットには、ストリームの受信者に関する情報が含まれていません。また、ターゲットをレプリカデータベースとする必要はありません。論理的なレプリケーションスロットをセットアップし、スロットから読み取りを行わない場合、データが書き込まれて、DB インスタンスのストレージがすぐにいっぱいになる可能性があります。

パラメータ、レプリケーション接続タイプ、およびセキュリティロールを使用して、Amazon RDS の PostgreSQL 論理レプリケーションおよび論理デコードをオンにします。論理デコード用のクライアントは、PostgreSQL DB インスタンスのデータベースにレプリケーション接続を確立できる任意のクライアントとすることができます。

RDS for PostgreSQL DB インスタンスに対して論理デコードをオンにするには

1. 使用しているユーザーアカウントに次のロールがあることを確認します。
   - 論理レプリケーションをオンにできるようにする rds_superuser ロール
   - 論理スロットを管理し、論理スロットを使用してデータをストリーミングするためのアクセス許可を付与する rds_replication ロール
2. rds.logical_replication 静的パラメータを 1 に設定します。このパラメータを適用する一環として、wal_level、max_wal_senders、max_replication_slots、max_connections の各パラメータも設定します。これらのパラメータの変更により、生成される WAL が増えることがあるため、論理スロットを使用する場合にのみ、rds.logical_replication パラメータを設定してください。
3. 静的 rds.logical_replication パラメータの DB インスタンスを再起動して有効にします。
4. 次のセクションの説明に従って論理的なレプリケーションスロットを作成します。このプロセスでは、デコードプラグインを指定する必要があります。現在、RDS for PostgreSQL

は、PostgreSQL に付属する出力プラグイン test_decoding および wal2json をサポートしています。

PostgreSQL 論理的なデコードの使用の詳細については、PostgreSQL のドキュメントを参照してください。

論理的なレプリケーションスロットの使用

SQL コマンドを使用して、論理的なスロットを操作できます。例えば、次のコマンドは、デフォルトの PostgreSQL 出力プラグイン `test_slot` を使用して、`test_decoding` という論理的なスロットを作成します。

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'test_decoding');
slot_name    | xlog_position
-----------------+---------------
regression_slot | 0/16B1970
(1 row)
```

論理的なスロットを一覧表示するには、次のコマンドを使用します。

```
SELECT * FROM pg_replication_slots;
```

論理的なスロットを削除するには、次のコマンドを使用します。

```
SELECT pg_drop_replication_slot('test_slot');
pg_drop_replication_slot
-----------------------
(1 row)
```

論理的なレプリケーションスロットのその他の使用例については、PostgreSQL ドキュメントの「Logical Decoding Examples」を参照してください。

論理的なレプリケーションスロットを作成すると、ストリーミングをスタートできます。次の例は、ストリーミングレプリケーションプロトコルで論理的なデコードがどのように制御されるかを示しています。この例では、PostgreSQL ディストリビューションに含まれているプログラム pg_recvlogical を使用しています。これを行うには、レプリケーション接続を許可するようにクライアント認証が設定されている必要があります。

```
pg_recvlogical -d postgres --slot test_slot -U postgres
    --host -instance-name.111122223333.aws-region.rds.amazonaws.com
```

```
    -f -  --start
```

pg_replication_origin_statusビューの内容を表示するには、pg_show_replication_origin_status 関数を照会します。

```
SELECT * FROM pg_show_replication_origin_status();
local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
(0 rows)
```

論理レプリケーションを使用したテーブルレベルのデータのレプリケーション

論理レプリケーションを使用して、RDS for PostgreSQL のソーステーブルからターゲットテーブルにデータをレプリケートできます。論理レプリケーションは、まずソーステーブルから既存のデータの初期ロードを実行し、引き続いて以降の変更をレプリケートします。

1. ソーステーブルを作成する

   RDS for PostgreSQL DB インスタンスのソースデータベースに接続します。

   ```
   source=> CREATE TABLE testtab (slno int primary key);
   CREATE TABLE
   ```

2. ソーステーブルにデータを挿入する

   ```
   source=> INSERT INTO testtab VALUES (generate_series(1,1000));
   INSERT 0 1000
   ```

3. ソーステーブルのパブリケーションを作成する

   - ソーステーブルのパブリケーションを作成します。

     ```
     source=> CREATE PUBLICATION testpub FOR TABLE testtab;
     CREATE PUBLICATION
     ```

   - SELECT クエリを使用して、作成したパブリケーションの詳細を確認します。

```
source=> SELECT * FROM pg_publication;
  oid   | pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete |
 pubtruncate | pubviaroot
--------+---------+----------+--------------+-----------+-----------+-----------
+-------------+------------
 115069 | testpub |    16395 | f            | t         | t         | t         |
 t           | f
(1 row)
```

- ソーステーブルがパブリケーションに追加されていることを確認します。

```
source=> SELECT * FROM pg_publication_tables;
pubname | schemaname | tablename
---------+------------+-----------
 testpub | public     | testtab
(1 rows)
```

- データベース内のすべてのテーブルをレプリケートするには、以下を使用します。

```
CREATE PUBLICATION testpub FOR ALL TABLES;
```

- パブリケーションが個々のテーブル用に既に作成されており、新しいテーブルを追加する必要がある場合は、以下のクエリを実行して、既存のパブリケーションに新しいテーブルを追加できます。

```
ALTER PUBLICATION <publication_name> add table <new_table_name>;
```

4. ターゲットデータベースに接続し、ターゲットテーブルを作成する

- ターゲット DB インスタンスのターゲットデータベースに接続します。ソーステーブルと同じ名前でターゲットテーブルを作成します。

```
target=> CREATE TABLE testtab (slno int primary key);
CREATE TABLE
```

- ターゲットテーブルに対して SELECT クエリを実行し、ターゲットテーブルにデータが存在しないことを確認します。

```
target=> SELECT count(*) FROM testtab;
 count
-------
     0
(1 row)
```

5. ターゲットデータベースでサブスクリプションを作成および検証する

- ターゲットデータベースにサブスクリプションを作成します。

```
target=> CREATE SUBSCRIPTION testsub
CONNECTION 'host=<source RDS/host endpoint> port=5432 dbname=<source_db_name>
 user=<user> password=<password>'
PUBLICATION testpub;
NOTICE:  Created replication slot "testsub" on publisher
CREATE SUBSCRIPTION
```

- SELECT クエリを使用して、サブスクリプションが有効になっていることを確認します。

```
target=> SELECT oid, subname, subenabled, subslotname, subpublications FROM
 pg_subscription;
  oid  | subname | subenabled | subslotname | subpublications
-------+---------+------------+-------------+-----------------
 16434 | testsub | t          | testsub     | {testpub}
(1 row)
```

- サブスクリプションが作成されると、ソーステーブルからターゲットテーブルにすべてのデータがロードされます。ターゲットテーブルに対して SELECT クエリを実行し、初期データがロードされていることを確認します。

```
target=> SELECT count(*) FROM testtab;
 count
-------
  1000
(1 row)
```

6. ソースデータベースのレプリケーションスロットを確認する

   ターゲットデータベースにサブスクリプションを作成すると、ソースデータベースにレプリケーションスロットが作成されます。ソースデータベースに対して次の SELECT クエリを実行して、レプリケーションスロットの詳細を確認します。

```
source=> SELECT * FROM pg_replication_slots;

slot_name |  plugin  | slot_type | datoid | database | temporary | active |
 active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn | wal_status
 | safe_wal_size
----------+----------+-----------+--------+----------+-----------+--------
+------------+------+--------------+-------------+---------------------
+------------+---------------
testsub   | pgoutput | logical   | 115048 | source   | f         | t      |
 846 |      |         6945 | 58/B4000568 | 58/B40005A0         | reserved   |
(1 row)
```

7. レプリケーションのテスト

   - ソーステーブルに行を挿入して、ソーステーブルのデータ変更がターゲットテーブルにレプリケートされているかどうかをテストします。

```
source=> INSERT INTO testtab VALUES(generate_series(1001,2000));
INSERT 0 1000

source=> SELECT count(*) FROM testtab;
 count
-------
  2000
(1 row)
```

- ターゲットテーブル内の行数を確認して、新しい挿入がレプリケートされていることを確認します。

```
target=> SELECT count(*) FROM testtab;
 count
-------
  2000
(1 row)
```

8. テーブルを追加した後のサブスクリプションの更新

- 既存のパブリケーションに新しいテーブルを追加する場合、変更を有効にするにはサブスクリプションを更新する必要があります。

```
ALTER SUBSCRIPTION <subscription_name> REFRESH PUBLICATION;
```

- このコマンドは、パブリッシャーから欠落しているテーブル情報を取得し、サブスクリプションの作成または最終更新以降にサブスクライブ先のパブリケーションに追加されたテーブルのレプリケーションを開始します。

論理レプリケーション接続の IAM 認証の設定

RDS for PostgreSQL バージョン 11 以降では、レプリケーション接続に AWS Identity and Access Management (IAM) 認証を使用できます。この機能は、パスワードの代わりに IAM ロールを使用してデータベースアクセスを管理できるようにすることで、セキュリティを強化します。クラスターとインスタンスの両方の粒度で動作し、標準の IAM 認証と同じセキュリティモデルに従います。

レプリケーション接続の IAM 認証はオプトイン機能です。有効にするには、DB クラスターまたは DB パラメータグループの `rds.iam_auth_for_replication` パラメータを 1 に設定します。これは動的パラメータであるため、DB クラスターまたはインスタンスを再起動する必要がなく、ダウンタイムなしで既存のワークロードで IAM 認証を活用できます。この機能を有効にする前に、以下の前提条件を満たす必要があります。

トピック

- 前提条件
- レプリケーション接続の IAM 認証の有効化
- レプリケーション接続の IAM 認証の無効化
- 制約事項と考慮事項

前提条件

レプリケーション接続に IAM 認証を使用するには、以下のすべての要件を満たす必要があります。

- RDS for PostgreSQL DB インスタンスはバージョン 11 以降である必要があります。
- パブリッシャー RDS for PostgreSQL DB インスタンスの場合:
  - IAM データベース認証を有効にする。詳細については、「IAM データベース認証の有効化と無効化」を参照してください。
  - `rds.logical_replication` パラメータを 1 に設定して、論理レプリケーションを有効にします。

論理レプリケーションでは、パブリッシャーはサブスクライバーデータベースにデータを送信するソース RDS for PostgreSQL データベースです。詳細については、「Amazon RDS for PostgreSQL の論理レプリケーションの実行」を参照してください。

> **Note**
>
> パブリッシャー RDS for PostgreSQL DB インスタンスで IAM 認証と論理レプリケーションの両方を有効にする必要があります。どちらかが有効になっていない場合、レプリケーション接続に IAM 認証を使用することはできません。

レプリケーション接続の IAM 認証の有効化

レプリケーション接続の IAM 認証を有効にするには、次の手順を実行します。

レプリケーション接続の IAM 認証を有効にするには

1. RDS for PostgreSQL DB クラスターまたはインスタンスが、レプリケーション接続による IAM 認証のすべての前提条件を満たしていることを確認します。詳細については、「前提条件」を参照してください。

2. RDS for PostgreSQL の設定に基づいて `rds.iam_auth_for_replication` パラメータを設定します。

   - RDS for PostgreSQL DB インスタンスの場合: DB パラメータグループを変更します。
   - マルチ AZ クラスターの場合: DB クラスターパラメータグループを変更します。

   `rds.iam_auth_for_replication` を 1 に設定します。これは、再起動を必要とせずにすぐに有効になる動的パラメータです。

   > **Note**
   >
   > マルチ AZ クラスターは、DB クラスターパラメータグループのみを使用します。マルチ AZ クラスターでは、個々のインスタンスパラメータグループを変更することはできません。

3. データベースに接続し、必要なロールをレプリケーションユーザーに付与します。

   次の SQL コマンドは、レプリケーション接続の IAM 認証を有効にするために必要なロールを付与します。

   ```
   -- Grant IAM authentication role
   GRANT rds_iam TO replication_user_name;

   -- Grant replication privileges
   ALTER USER replication_user_name WITH REPLICATION;
   ```

   これらのステップを完了した後、指定されたユーザーはレプリケーション接続に IAM 認証を使用する必要があります。

   > **Important**
   >
   > この機能を有効にすると、`rds_iam` ロールと `rds_replication` ロールの両方を持つユーザーは、レプリケーション接続に IAM 認証を使用する必要があります。これは、ロールがユーザーに直接割り当てられているか、他のロールを通じて継承されているかにかかわらず適用されます。

レプリケーション接続の IAM 認証の無効化

レプリケーション接続の IAM 認証を無効にするには、次のいずれかの方法を使用します。

- DB インスタンスの DB パラメータグループまたはマルチ AZ クラスターの DB クラスターパラメータグループで、`rds.iam_auth_for_replication` パラメータを 0 に設定します。
- または、RDS for PostgreSQL DB クラスターまたはインスタンスで次のいずれかの機能を無効にすることもできます。
  - `rds.logical_replication` パラメータを 0 に設定して論理レプリケーションを無効にする
  - IAM 認証の無効化

この機能を無効にすると、レプリケーション接続では認証にデータベースパスワードを使用できるようになります。

> **Note**
>
> `rds_iam` ロールを持たないユーザーのレプリケーション接続は、この機能が有効になっている場合でもパスワード認証を使用できます。

制約事項と考慮事項

論理レプリケーション接続に IAM 認証を使用する場合、以下の制限と考慮事項を考慮してください。

- この機能は、RDS for PostgreSQL バージョン 11 以降でのみ使用できます。
- パブリッシャーは、レプリケーション接続の IAM 認証をサポートする必要があります。
- IAM 認証トークンは、デフォルトで 15 分後に期限切れになります。トークンの有効期限が切れる前に、長時間実行されるレプリケーション接続を更新する必要がある場合があります。

stats_temp_directory の RAM ディスク

RDS for PostgreSQL パラメータ `rds.pg_stat_ramdisk_size` を使用して、PostgreSQL `stats_temp_directory` を保存する RAM ディスクに割り当てられたシステムメモリを指定できます。RAM ディスクパラメータは、RDS for PostgreSQL バージョン 14 以下のバージョンでのみ利用できます。

特定のワークロードでは、このパラメータを設定することでパフォーマンスが向上し、I/O 要件を軽減することができます。stats_temp_directory の詳細については、PostgreSQL のドキュメントを参照してください。

stats_temp_directory の RAM ディスクをセットアップにするには、rds.pg_stat_ramdisk_size パラメータを、DB インスタンスで使用されるパラメータグループの整数リテラル値に設定します。このパラメータは MB を表すため、整数値を使用する必要があります。表現、数式、関数が rds.pg_stat_ramdisk_size パラメータに対して有効ではありません。変更が反映されるように、DB インスタンスを再起動してください。パラメータの設定の詳細については、「Amazon RDS のパラメータグループ」を参照してください。

例えば、次の AWS CLI コマンドは、RAM ディスクパラメータを 256 MB に設定します。

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name pg-95-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

再起動後は、次のコマンドを実行して stats_temp_directory のステータスを確認します。

```
postgres=> SHOW stats_temp_directory;
```

コマンドは次の情報を返します。

```
stats_temp_directory
----------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

RDS for PostgreSQL のテーブルスペース

RDS for PostgreSQL では、互換性のためにテーブルスペースがサポートされています。すべてのストレージが 1 つの論理ボリューム上にあるため、I/O 分割または分離にテーブルスペースを使用することはできません。当社のベンチマークと経験は、ほとんどのユースケースで 単一の論理的なボリュームが最適なセットアップであることを示しています。

RDS for PostgreSQL DB インスタンスでテーブルスペースを作成して使用するには、rds_superuser ロール が必要です。RDS for PostgreSQL DB インスタンスのメインユーザーアカウント (デフォルトの名前は postgres) は、このロールのメンバーです。詳細については、「PostgreSQL のロールとアクセス権限について」を参照してください。

表空間の作成時にファイル名を指定する場合、パスの接頭辞は `/rdsdbdata/db/base/tablespace` になります。次の例では、`/rdsdbdata/db/base/tablespace/data`に表空間ファイルを配置しています。この例では、`dbadmin` ユーザー (ロール) が存在し、テーブルスペースの操作に必要な `rds_superuser` ロールが付与されていることを前提としています。

```
postgres=> CREATE TABLESPACE act_data
  OWNER dbadmin
  LOCATION '/data';
CREATE TABLESPACE
```

PostgreSQL テーブルスペースの詳細については、PostgreSQL のドキュメントの「Tablespaces」(テーブル空間) を参照してください。

EBCDIC やその他のメインフレーム移行のための RDS for PostgreSQL 照合順序

RDS for PostgreSQL バージョン 10 以降には ICU バージョン 60.2 が含まれています。これは Unicode 10.0 に基づいており、Unicode Common Locale Data Repositor、CLDR 32 からの照合順序が含まれています。これらのソフトウェア国際化ライブラリにより、オペレーティングシステムやプラットフォームに関係なく、文字エンコーディングが一貫した方法で表示されます。Unicode CLDR-32 の詳細については、Unicode CLDR のウェブサイトの「CLDR 32 リリースノート」を参照してください。Unicode の国際化コンポーネント (ICU) については、ICU 技術委員会 (ICU-TC) のウェブサイトで詳しく説明されています。ICU-60 の詳細については、「ICU 60 のダウンロード」を参照してください。

バージョン 14.3 以降、RDS for PostgreSQL には、EBCDIC ベースのシステムからのデータ統合と変換に役立つ照合順序も含まれています。Extended Binary Coded Decimal Interchange Code (EBCDIC) エンコーディングはメインフレームのオペレーティングシステムで一般的に使用されます。マッピングAmazon RDS が提供するこれらの照合順序は、EBCDIC コードページに直接マッピングされる Unicode 文字のみをソートするように狭義に定義されています。文字は、変換後のデータ検証を可能にするために、EBCDIC コードポイント順にソートされます。これらの照合順序には、非正規化された形式や、ソース EBCDIC コードページの文字に直接マッピングされない Unicode 文字は含まれていません。

EBCDIC コードページと Unicode コードポイント間の文字マッピングは、IBM が公開している表に基づいています。一式は、IBM から圧縮ファイルとしてダウンロード可能です。RDS for PostgreSQL は、ICU から提供されたツールでこれらのマッピングを使用して、このセクションの表にリストされている照合順序を作成しました。照合順序名には、ICU が要求する言語と国が含まれま

す。ただし、EBCDIC コードページでは言語が指定されておらず、一部の EBCDIC コードページは複数の国を対象としています。つまり、テーブル内の照合名の言語と国の部分は任意であり、現在のロケールと一致する必要はありません。つまり、コードページ番号はこの表の照合順序名の最も重要な部分です。次の表に示す照合順序は、どの RDS for PostgreSQL データベースでも使用できます。

- Unicode to EBCDIC collations table — メインフレームのデータ移行ツールの中には、LATIN1 または LATIN9 を内部的に使用してデータのエンコードと処理を行うものがあります。こういったツールは、ラウンドトリップスキームを使用してデータの整合性を維持し、逆変換をサポートします。この表の照合順序は、特別な処理を必要としない LATIN1 エンコーディングを使用してデータを処理するツールで使用できます。
- Unicode to LATIN9 collations table — これらの照合順序は、どの RDS for PostgreSQL データベースでも使用できます。

次の表に、EBCDIC コードページを Unicode コードポイントにマッピングする RDS for PostgreSQL の照合順序を示します。IBM コードページの順序に基づいてソートする必要があるアプリケーション開発には、この表の照合順序を使用することをお勧めします。

PostgreSQL 照合順序名	コードページのマッピングとソート順序の説明
da-DK-cp277-x-icu	IBM EBCDIC コードページ 277 (変換表ごと) に直接マッピングされる Unicode 文字は、IBM CP 277 コードポイント順にソートされます。
de-DE-cp273-x-icu	IBM EBCDIC コードページ 273 (変換表ごと) に直接マッピングされる Unicode 文字は、IBM CP 273 コードポイント順にソートされます。
en-GB-cp285-x-icu	IBM EBCDIC コードページ 285 (変換表ごと) に直接マッピングされる Unicode 文字は、IBM CP 285 コードポイント順にソートされます。
en-US-cp037-x-icu	IBM EBCDIC コードページ 037 (変換表ごと) に直接マッピングされる Unicode 文字は、IBM CP 037 コードポイント順にソートされます。

PostgreSQL 照合順序名	コードページのマッピングとソート順序の説明
es-ES-cp284-x-icu	IBM EBCDIC コードページ 284 (変換表ごと) に直接マッピングされる Unicode 文字は、IBM CP 284 コードポイント順にソートされます。
fi-FI-cp278-x-icu	IBM EBCDIC コードページ 278 (変換表ごと) に直接マップされる Unicode 文字は、IBM CP 278 コードポイント順にソートされます。
fr-fr-cp297-X-ICU	IBM EBCDIC コードページ 297 (変換表ごと) に直接マップされる Unicode 文字は、IBM CP 297 コードポイント順にソートされます
it-IT-cp280-x-icu	IBM EBCDIC コードページ 280 (変換表ごと) に直接マップされる Unicode 文字は、IBM CP 280 コードポイント順にソートされます
nl-BE-cp500-x-icu	IBM EBCDIC コードページ 500 (変換テーブルごと) に直接マップされる Unicode 文字は、IBM CP 500 コードポイント順にソートされます

Amazon RDS には、IBM が公開しているテーブルを使用して、LATIN9 文字にマッピングされる Unicode コードポイントをソースデータの EBCDIC コードページに従って元のコードポイントの順序でソートする追加の照合セットが用意されています。

PostgreSQL 照合順序名	コードページのマッピングとソート順序の説明
da-DK-cp1142m-x-icu	IBM EBCDIC コードページ 1142 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1142 コードポイント順にソートされます。
de-DE-cp1141m-x-icu	IBM EBCDIC コードページ 1141 (変換テーブルごと) から最初に変換された LATIN9 文字に

PostgreSQL 照合順序名	コードページのマッピングとソート順序の説明
	マッピングされる Unicode 文字は、IBM CP 1141 コードポイント順にソートされます。
en-GB-cp1146m-x-icu	IBM EBCDIC コードページ 1146 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1146 コードポイント順にソートされます。
en-US-cp1140m-x-icu	IBM EBCDIC コードページ 1140 (変換テーブルごと) から最初に変換された LATIN9 文字にマップされる Unicode 文字は、IBM CP 1140 コードポイント順にソートされます。
es-ES-cp1145m-x-icu	IBM EBCDIC コードページ 1145 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1145 コードポイント順にソートされます。
fi-FI-cp1143m-x-icu	IBM EBCDIC コードページ 1143 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1143 コードポイント順にソートされます。
fr-FR-cp1147m-x-icu	IBM EBCDIC コードページ 1147 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1147 コードポイント順にソートされます。
it-IT-cp1144m-x-icu	IBM EBCDIC コードページ 1144 (変換テーブルごと) から最初に変換された LATIN9 文字にマッピングされる Unicode 文字は、IBM CP 1144 コードポイント順にソートされます。
nl-BE-cp1148m-x-icu	IBM EBCDIC コードページ 1148 (変換テーブルごと) から最初に変換された LATIN9 文字にマップされる Unicode 文字は、IBM CP 1148 コードポイント順にソートされます。

以下に、RDS for PostgreSQL 照合順序の使用例を示します。

```
db1=> SELECT pg_import_system_collations('pg_catalog');
 pg_import_system_collations
-----------------------------
                          36
db1=> SELECT '¤' < 'a' col1;
 col1
------
 t
db1=> SELECT '¤' < 'a' COLLATE "da-DK-cp277-x-icu" col1;
 col1
------
 f
```

IBM コードページの順序に基づいてソートする必要があるアプリケーション開発には、Unicode to EBCDIC collations table と Unicode to LATIN9 collations table の照合順序を使用することをお勧めします。次の照合順序 (接尾辞に「b」の文字が付いている) は、pg_collation でも表示されますが、特定のコードポイントシフトを持つコードページをマッピングする AWS のメインフレームデータ統合および移行ツールで使用することを目的としており、照合順序で特別な処理が必要です。つまり、以下の照合順序の使用は推奨されません。

- da-DK-277b-x-icu
- da-DK-1142b-x-icu
- de-DE-cp273b-x-icu
- de-DE-cp1141b-x-icu
- en-GB-cp1146b-x-icu
- en-GB-cp285b-x-icu
- en-US-cp037b-x-icu
- en-US-cp1140b-x-icu
- es-ES-cp1145b-x-icu
- es-ES-cp284b-x-icu
- fi-FI-cp1143b-x-icu
- fr-FR-cp1147b-x-icu
- fr-FR-cp297b-x-icu
- it-IT-cp1144b-x-icu

- it-IT-cp280b-x-icu
- nl-BE-cp1148b-x-icu
- nl-BE-cp500b-x-icu

メインフレーム環境から AWS へのアプリケーションの移行に関する詳細については、「AWSMainframe Modernization とは?」を参照してください。

PostgreSQL における照合順序の管理の詳細については、PostgreSQL のドキュメントの「Collation Support」(照合順序のサポート) を参照してください。

RDS for PostgreSQL の論理スロット同期の管理

コミュニティ PostgreSQL 17 以降、プライマリサーバーからスタンバイサーバーに論理レプリケーションスロットを自動的に同期する新機能が、パラメータ `sync_replication_slots` や関連する関数 `pg_sync_replication_slots()` を通じて導入されました。これは、実行時にスロットを手動で同期します。

これらの機能は、RDS for PostgreSQL 17 以降で使用できます。一般的なセットアップには、プライマリインスタンスとそのリードレプリカと、プライマリへの論理レプリケーションサブスクライバーがあります。

サブスクリプションがフェイルオーバーオプションを true に設定して作成されていることを確認します。

```
CREATE SUBSCRIPTION subname CONNECTION 'host=...' PUBLICATION pubname WITH (failover =
 true);
```

これにより、フェイルオーバーを有効にした状態でパブリッシャーに論理スロットが作成されます。

```
postgres=> SELECT slot_name, slot_type, failover FROM pg_catalog.pg_replication_slots;
 slot_name | slot_type | failover
-----------+-----------+----------
 subname   | logical   | t
(1 row)
```

スロット同期を有効にすると、プライマリのすべてのフェイルオーバー論理レプリケーションスロットが物理スタンバイに自動的に作成され、定期的に同期されます。パラメータグループを通じて次の値が設定されていることを確認します。

- 論理レプリケーションを有効にするには rds.logical_replication が 1 である必要があります。
- hot_standby_feedback はスタンバイ時に 1 である必要があります。
- スタンバイの rds.logical_slot_sync_dbname には有効なデータベース名を設定する必要があります。

  パラメータのデフォルト値は postgres です。論理パブリッシングインスタンスに postgres データベースがある場合、デフォルトのパラメータを変更する必要はありません。
- プライマリの synchronized_standby_slots は、同期するスタンバイの物理レプリケーションスロットに設定する必要があります。
- 自動同期を有効にするには sync_replication_slots が 1 である必要があります。

フェイルオーバー対応のサブスクリプションスロットと上記のパラメータ値を使用すると、スタンバイが昇格したとき、サブスクライバーはサブスクリプションをこの新しく昇格されたインスタンスに変更し、論理レプリケーションをシームレスに続行できます。

PostgreSQL データベースエンジンを実行する DB インスタンスへの接続

Amazon RDS によって DB インスタンスがプロビジョニングされると、スタンダードの SQL クライアントアプリケーションを使用してインスタンスに接続できます。接続する前に、DB インスタンスが使用可能でアクセス可能である必要があります。VPC の外部からインスタンスに接続できるかどうかは、Amazon RDS DB インスタンスの作成方法によって異なります。

- DB インスタンスを公開で作成した場合、VPC 外部のデバイスと Amazon EC2 インスタンスからデータベースに接続できます。
- DB インスタンスをプライベートで作成した場合、Amazon VPC 内の Amazon EC2 インスタンスとデバイスのみがデータベースに接続できます。

DB インスタンスがパブリックかプライベートかを確認するには、AWS マネジメントコンソール を使用してインスタンスの [Connectivity & security] (接続とセキュリティ) タブを確認します。[Security] (セキュリティ) では、「パブリックアクセス可能」の値を見つけることができます。プライベートの場合は [No] (いいえ)、パブリックの場合は [Yes] (はい) です。

Amazon RDS および Amazon VPC のさまざまな設定、およびそれらがアクセシビリティに与える影響の詳細については、「VPC の DB インスタンスにアクセスするシナリオ」を参照してください。

目次

- psql クライアントをインストールする
- RDS for PostgreSQL DB インスタンスの接続情報の検索
- pgAdmin を使用して RDS PostgreSQL DB インスタンスに接続する
- psql を使用したRDS for PostgreSQL DB インスタンスへの接続
- Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for PostgreSQL への接続
- Amazon Web Services (AWS) Python ドライバーを使用した RDS for PostgreSQL への接続
- RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング
  - – 致命的エラー: データベース名が存在しません。
  - - サーバー接続失敗エラー: 接続がタイムアウトしました。
  - セキュリティグループのアクセスルールのエラー

psql クライアントをインストールする

EC2 インスタンスから DB インスタンスに接続するには、EC2 インスタンスに PostgreSQL クライアントをインストールします。最新バージョンの psql クライアントを Amazon Linux 2023 にインストールするには、次のコマンドを実行します。

```
sudo dnf install postgresql<version number>
```

最新バージョンの psql クライアントを Amazon Linux 2 にインストールするには、次のコマンドを実行します。

```
sudo yum install -y postgresql
```

最新バージョンの psql クライアントを Ubuntu にインストールするには、次のコマンドを実行します。

```
sudo apt install -y postgresql-client
```

RDS for PostgreSQL DB インスタンスの接続情報の検索

DB インスタンスが利用可能でアクセス可能な場合は、SQL クライアントアプリケーションに次の情報を提供することによって接続できます。

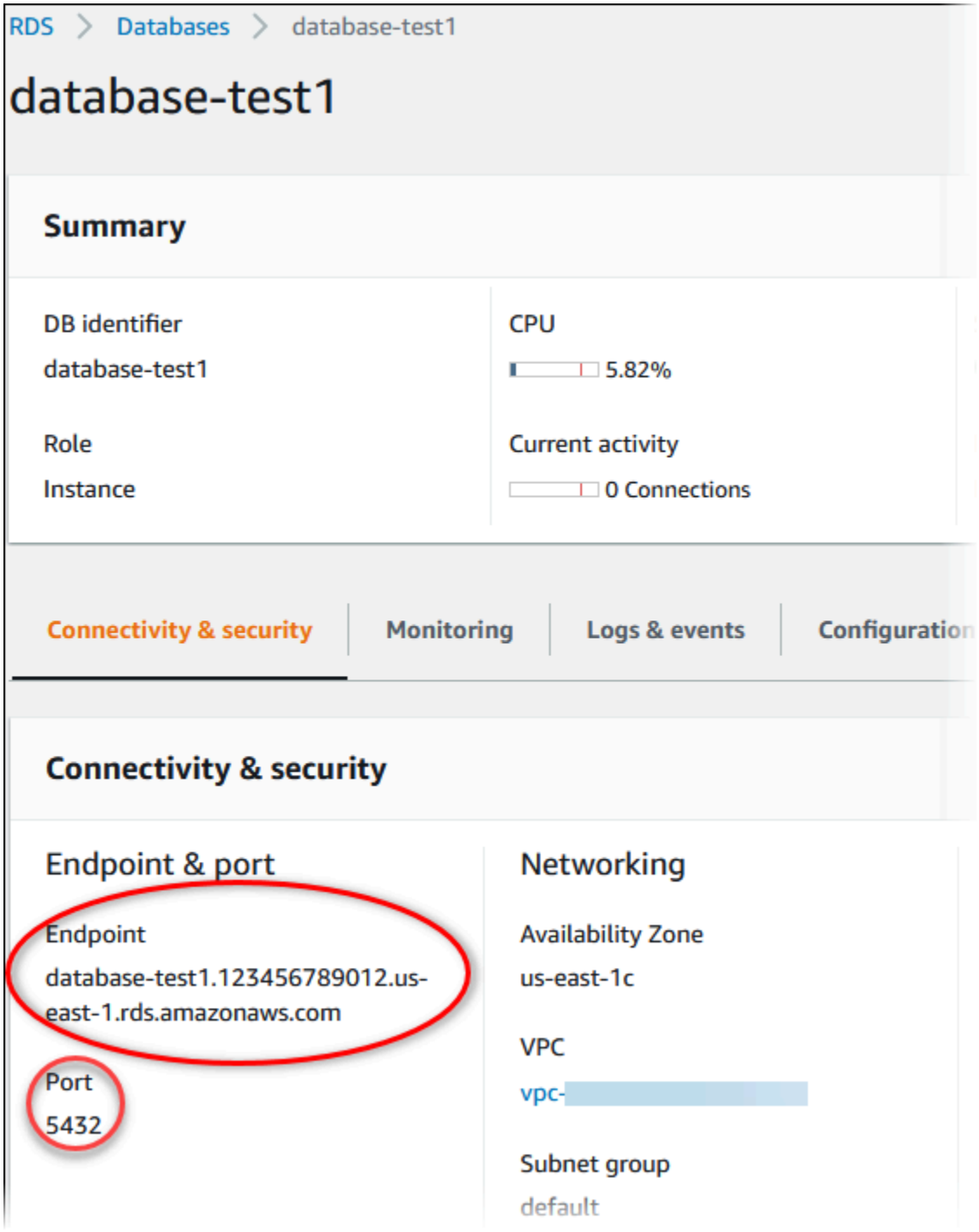
- DB インスタンスのエンドポイントで、インスタンスのホスト名 (DNS 名)として機能します。
- DB インスタンスがリッスンするポート。PostgreSQL のデフォルトポートは 5432 です。
- DB インスタンスのユーザーネームとパスワード。PostgreSQL のデフォルトの「マスターユーザーネーム」は`postgres`です。
- データベースの名前とパスワード (DB 名)です。

これらの詳細は、AWS マネジメントコンソール、AWS CLI`describe-db-instances`コマンド、または Amazon RDS API `DescribeDBInstances`オペレーションを使用して取得できます。

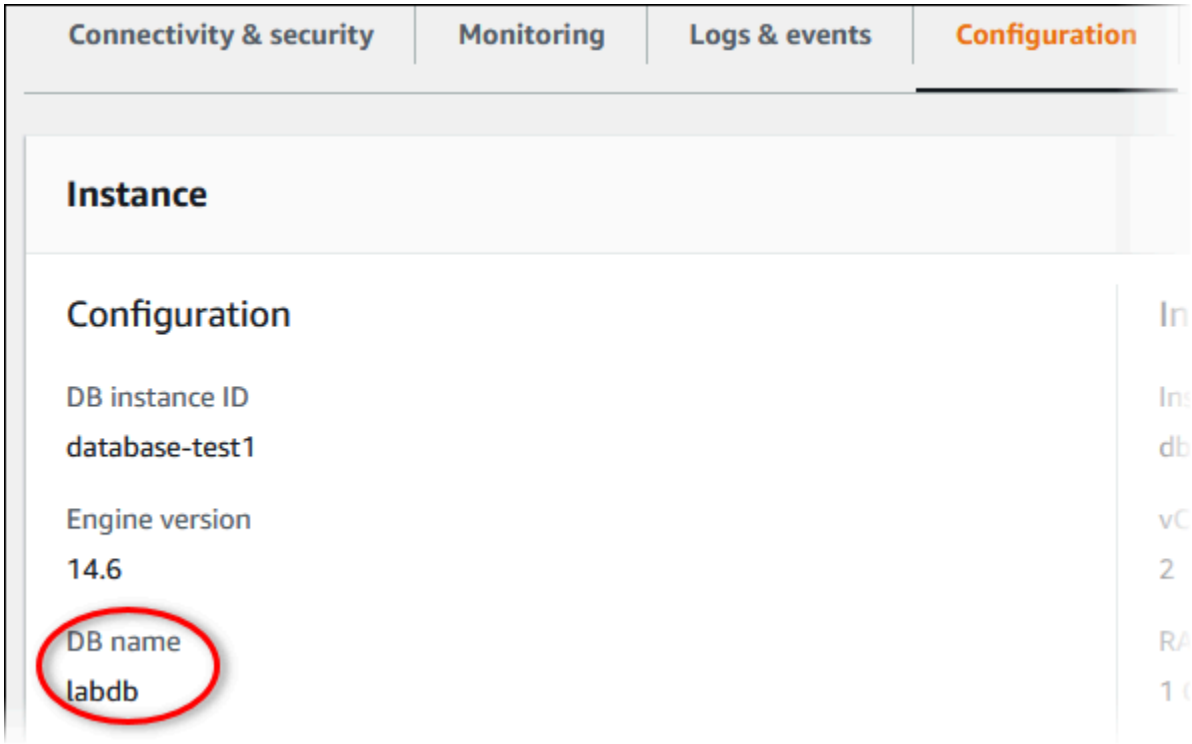
エンドポイント、ポート番号、および DB 名を検索するにはAWS マネジメントコンソールを使用します。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. RDS コンソールを開き、[データベース] を選択して、DB インスタンスを一覧表示します。
3. PstgreSQL DB インスタンス名を選択して、詳細を表示します。
4. 接続とセキュリティ タブで、エンドポイントをコピーします。また、ポート番号を書き留めます。DB インスタンスに接続するには、エンドポイントとポート番号の両方が必要です。



5. 設定タブで、DB 名をメモします。RDS for PostgreSQL インスタンスの作成時にデータベースを作成した場合は、DB 名に名前が表示されます。データベースを作成しなかった場合、DB 名にダッシュ (-) が表示されます。



以下に、PostgreSQL DB インスタンスに接続する 2 つの方法を示します。初期の例では、オープンソースの PostgreSQL 向け管理開発ツールとして人気のある pgAdmin を使用します。2 番目の例では、PostgreSQL に付属するコマンドラインユーティリティである psql を使用します。

pgAdmin を使用して RDS PostgreSQL DB インスタンスに接続する

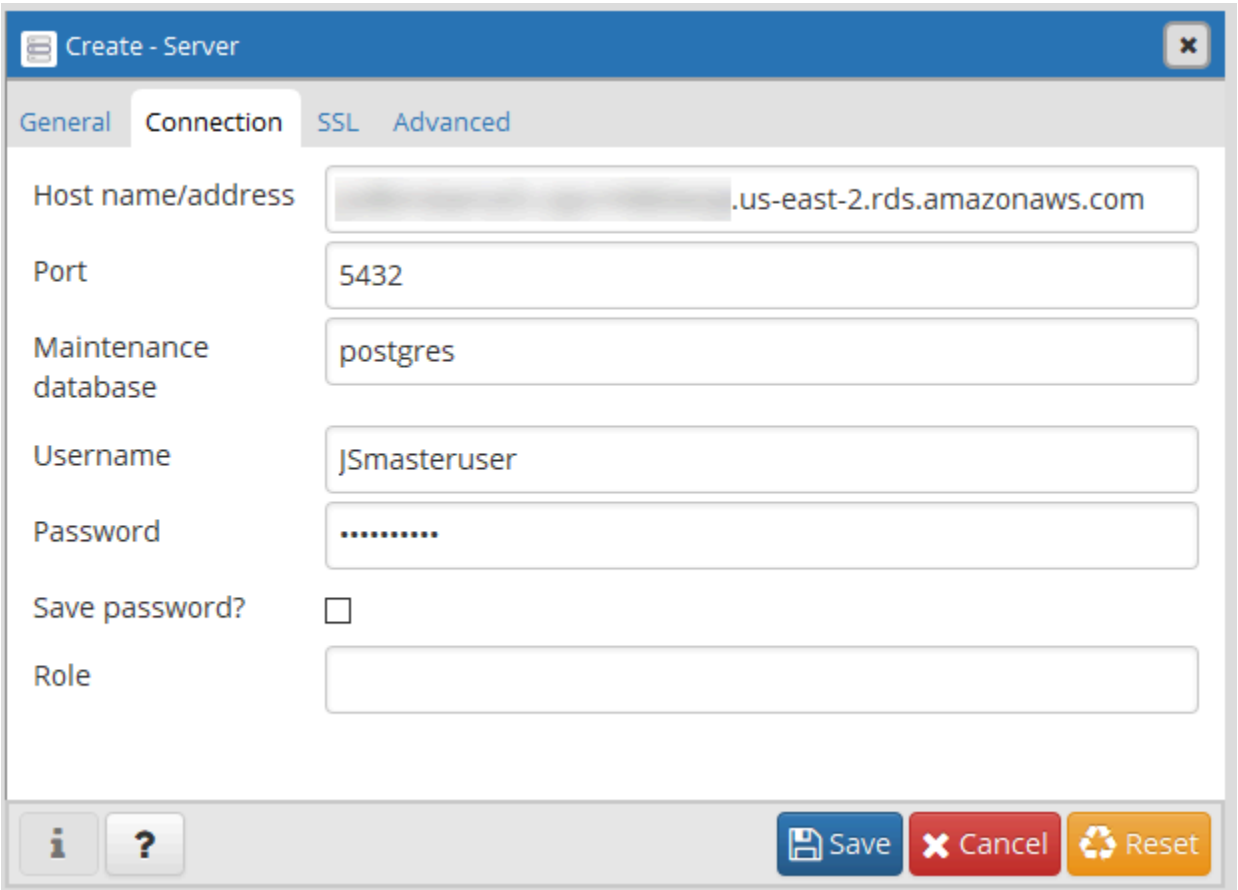
オープンソースのツール pgAdmin を使用して、RDS for PostgreSQL DB インスタンスに接続できます。クライアントコンピュータに PostgreSQL のローカルインスタンスがなくても、pgAdmin をhttp://www.pgadmin.org/ からダウンロードして使用できます。

pgAdmin を使用して、 RDS for PostgreSQL DB インスタンスに接続するには

1. クライアントコンピュータの pgAdmin アプリケーションを起動します。
2. [Dashboard] (ダッシュボード) タブで、[Add New Server] (新しいサーバーの追加) を選択します。
3. [Create - Server] (作成 - サーバー) ダイアログボックスの [General] (全般) タブで名前を入力して、pgAdmin のサーバーを特定します。

4. [Connection] (接続) タブで、DB インスタンスから以下の情報を入力します。

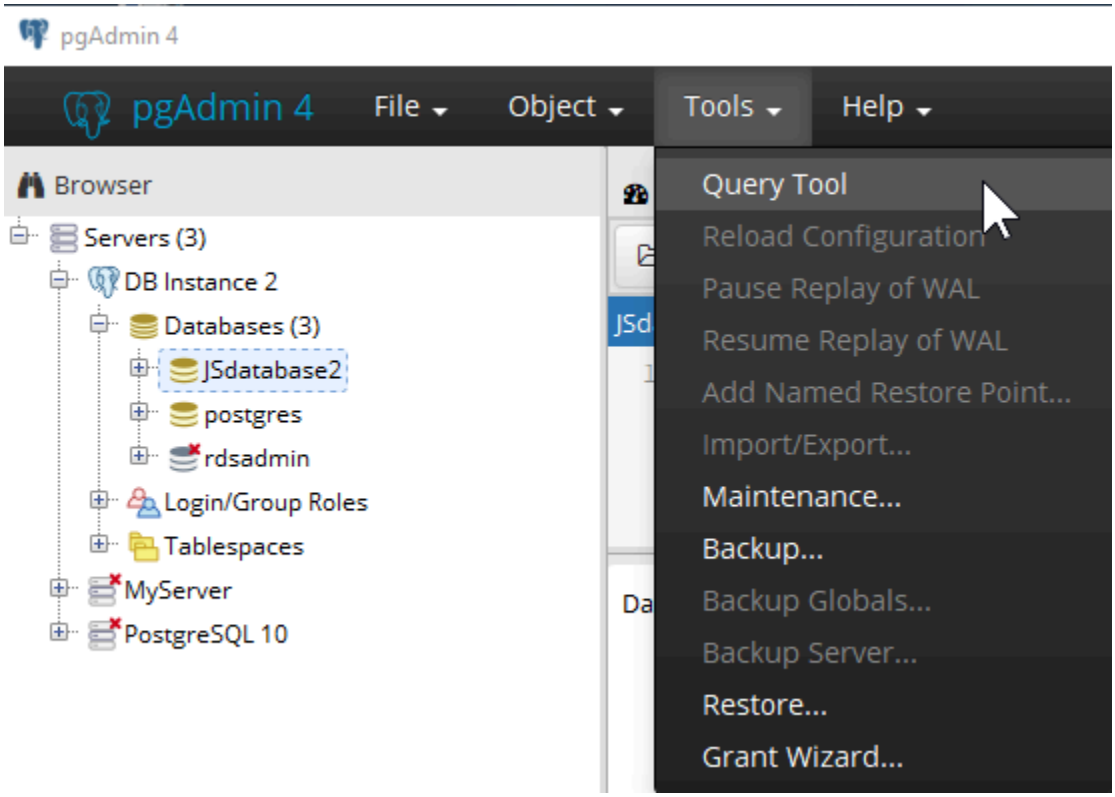
   - [ホスト] に、エンドポイントを入力します (例: `mypostgresql.c6c8dntfzzhgv0.us-east-2.rds.amazonaws.com`)。
   - [Port] (ポート) には、割り当てられているポートを入力します。
   - ユーザーネームでは、DB インスタンスの作成時に入力したユーザーネームを入力します (「マスターユーザーネーム」をデフォルトから変更した場合、`postgres`)。
   - パスワード に、DB インスタンスの作成時に入力したパスワードを入力します。



5. Save を選択します。

   接続に問題がある場合は、RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング を参照してください。

6. pgAdmin ブラウザでデータベースにアクセスするには、[Servers] (サーバー)、DB インスタンス、および [Databases] (データベース) を展開します。DB インスタンスのデータベース名を選択します。



7. SQL コマンドを入力できるパネルを開くには、[Tools] (ツール)、[Query Tool] (クエリツール) の順に選択します。

psql を使用したRDS for PostgreSQL DB インスタンスへの接続

psql コマンドラインユーティリティのローカルインスタンスを使用して、RDS for PostgreSQL DB インスタンスに接続できます。PostgreSQL またはクライアントコンピュータにインストールされた psql クライアントのいずれかが必要です。

PostgreSQL クライアントは、PostgreSQL のウェブサイトからダウンロードできます。オペレーティングシステムのバージョンに対応した手順に従い、psql をインストールします。

psql を使用して RDS for PostgreSQL DB インスタンスに接続するには、ホスト (DNS) 情報とアクセス認証情報、データベース名を指定する必要があります。

RDS for PostgreSQL DB インスタンスに接続するには、以下の形式のいずれかを使用します。接続時にパスワードを求められます。バッチジョブまたはスクリプトには、`--no-password` オプションを使用します。このオプションは、セッション全体に対して設定されます。

> **Note**
>
> サーバーがパスワード認証を要求し、他の出典からパスワードを取得できない場合、--no-password との接続試行に失敗します。詳細については、psql ドキュメントを参照してください。

この DB インスタンスに初めて接続する場合、またはこの RDS for PostgreSQL インスタンスのデータベースをまだ作成していない場合、「マスターユーザーネーム」とパスワードを使用してPostgresのデータベースに接続できます。

Unix の場合、次の形式を使用します。

```
psql \
   --host=<DB instance endpoint> \
   --port=<port> \
   --username=<master username> \
   --password \
   --dbname=<database name>
```

Windows の場合、次の形式を使用します。

```
psql ^
   --host=<DB instance endpoint> ^
   --port=<port> ^
   --username=<master username> ^
   --password ^
   --dbname=<database name>
```

例えば、次のコマンドは、架空の認証情報を使用して、mypgdb という PostgreSQL DB インスタンス上の mypostgresql というデータベースに接続します。

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --
username=awsuser --password --dbname=mypgdb
```

Amazon Web Services (AWS) JDBC ドライバーを使用した RDS for PostgreSQL への接続

Amazon Web Services (AWS) JDBC ドライバーは、高度な JDBC ラッパーとして設計されています。このラッパーは、既存の JDBC ドライバーの機能を補完し、拡張します。ドライバーには、コミュニティ pgJDBC ドライバーとドロップイン互換性があります。

AWS JDBC ドライバーをインストールするには、AWS JDBC ドライバーの .jar ファイル (CLASSPATH アプリケーション内) を追加して、それぞれのコミュニティドライバーへの参照を保持します。対応する接続 URL プレフィックスを次のように更新します。

- `jdbc:postgresql://` ~ `jdbc:aws-wrapper:postgresql://`

AWS JDBC ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) JDBC ドライバー GitHub リポジトリ」を参照してください。

Amazon Web Services (AWS) Python ドライバーを使用した RDS for PostgreSQL への接続

Amazon Web Services (AWS) Python ドライバーは、高度な Python ラッパーとして設計されています。このラッパーは、オープンソースの Psycopg ドライバーの機能を補完し、拡張します。AWS Python ドライバーは Python バージョン 3.8 以降をサポートしています。`aws-advanced-python-wrapper` パッケージは、`pip` コマンドと `psycopg` オープンソースパッケージを使用してインストールできます。

AWS Python ドライバーおよびその使用方法の詳細については、「Amazon Web Services (AWS) Python Driver GitHub repository」を参照してください。

RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング

トピック

- – 致命的エラー: データベース名が存在しません。
- - サーバー接続失敗エラー: 接続がタイムアウトしました。
- セキュリティグループのアクセスルールのエラー

– 致命的エラー: データベース#が存在しません。

接続時にFATAL: database *name* does not existのようなエラーが発生する場合、デフォルトのデータベース名 postgres を --dbname オプションに使用してみてください。

- サーバー接続失敗エラー: 接続がタイムアウトしました。

DB インスタンスに接続できない場合、最も一般的なエラーは Could not connect to server: Connection timed out. です。このエラーを受け取った場合、以下を確認します。

- 使用しているホスト名が DB インスタンスのエンドポイントであること、および使用しているポート番号が正しいことを確認します。
- DB インスタンスのパブリックアクセシビリティがはいに設定され、外部接続が許可されていることを確認します。パブリックアクセス設定を変更するには 、Amazon RDS DB インスタンスを変更する を参照してください。
- データベースに接続するユーザーに CONNECT アクセス権があることを確認してください。次のクエリを使用して、データベースへの接続アクセスを提供できます。

```
GRANT CONNECT ON DATABASE database name TO username;
```

- 接続で経由する可能性のあるファイアウォールを通過するアクセス許可のルールが、DB インスタンスに割り当てられたセキュリティグループに設定されていることを確認します。例えば、DB インスタンスをデフォルトポート 5432 で作成した場合、会社のファイアウォールルールにより、外部企業デバイスから当該ポートへの接続がブロックされる可能性があります。

  この問題を解決するには、別のポートを使用するよう DB インスタンスを変更します。また、DB インスタンスに適用されているセキュリティグループが、その新しいポートへの接続を許可していることも確認します。データベースポートの設定を変更するには 、「Amazon RDS DB インスタンスを変更する」を参照してください 。
- 使用しようとしているポートが PostgreSQL のローカルインスタンスまたはコンピュータで実行している別のサービスによって既に占有されているかどうかを確認します。例えば、同じポート (デフォルトは 5432) でローカル PostgreSQL データベースを実行している場合、RDS for PostgreSQL DB インスタンスに正常に接続できない場合があります。ポートが空いていることを確認するか、可能であれば別のポート番号で接続してみてください。
- 「セキュリティグループのアクセスルールのエラー」も参照してください。

セキュリティグループのアクセスルールのエラー

最も一般的な接続の問題は、DB インスタンスに割り当てられているセキュリティグループのアクセスルールに関係するものです。DB インスタンスの作成時にデフォルトのセキュリティグループを使用した場合、恐らくそのセキュリティグループにはインスタンスへのアクセスを許可するルールはありません。

接続が機能するには、作成時に DB インスタンスに割り当てたセキュリティグループが DB インスタンスへのアクセスを許可する必要があります。例えば、DB インスタンスが VPC で作成された場合、接続を承認する VPC セキュリティグループが必要です。アプリケーションが実行中のデバイスまたはAmazon EC2 インスタンスからの接続を承認しないセキュリティグループを使用して、 DB インスタンスを作成したかどうかを確認します。

セキュリティグループでインバウンドのルールを追加または編集できます。出典 に対して マイ IP を選択すると、ブラウザで検出された IP アドレスから DB インスタンスにアクセスできます。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

または、DB インスタンスが VPC の外部で作成された場合は、それらの接続を承認するデータベースセキュリティグループが必要です。

Amazon RDS セキュリティグループの詳細については、「セキュリティグループによるアクセス制御」を参照してください。

SSL/TLS を使用して RDS for PostgreSQL への接続を保護する

RDS for PostgreSQL では、PostgreSQL DB インスタンスの Secure Socket Layer (SSL) 暗号化がサポートされています。SSL を使用して、アプリケーションと PostgreSQL DB インスタンスとの PostgreSQL 接続を暗号化できます。また、PostgreSQL DB インスタンスへのすべての接続に SSL の使用を強制することができます。RDS for PostgreSQL では、SSL の後継プロトコルである Transport Layer Security (TLS) もサポートされています。

Amazon RDS およびデータ保護 (SSL/TLS を使用した接続の暗号化を含む) の詳細については、「Amazon RDS でのデータ保護」を参照してください。

トピック

- PostgreSQL DB インスタンスで SSL を使用する
- 新しい SSL/TLS 証明書を使用して PostgreSQL DB インスタンスに接続するようにアプリケーションを更新する

PostgreSQL DB インスタンスで SSL を使用する

Amazon RDS は、PostgreSQL DB インスタンスの Secure Socket Layer (SSL) 暗号化をサポートします。SSL を使用して、アプリケーションと PostgreSQL DB インスタンスとの PostgreSQL 接続を暗号化できます。デフォルトでは、RDS for PostgreSQL は SSL/TLS を使用し、すべてのクライアントが SSL/TLS を使用して接続することを想定していますが、必須にすることもできます。RDS for PostgreSQL は、Transport Layer Security (TLS) バージョン 1.1、1.2、および 1.3 をサポートしています。

SSL サポートおよび PostgreSQL データベースの一般情報については、PostgreSQL ドキュメントの「SSL Support」を参照してください。JDBC を介した SSL 接続の使用については、PostgreSQL ドキュメントの「Configuring the Client」を参照してください。

PostgreSQL 用の SSL は、すべての AWS リージョンで利用が可能です。インスタンスの作成時に、PostgreSQL DB インスタンス用の SSL 証明書が、Amazon RDS により作成されます。SSL 証明書認証を有効にした場合、SSL 証明書には、なりすまし攻撃から保護するために、SSL 証明書の共通名 (CN) として DB インスタンスのエンドポイントが含まれます。

トピック

- SSL 経由での PostgreSQL DB インスタンスへの接続
- PostgreSQL DB インスタンスへの SSL 接続を必須にする

- SSL 接続ステータスを確認する
- RDS for PostgreSQL の SSL 暗号スイート

SSL 経由での PostgreSQL DB インスタンスへの接続

SSL を使用して PostgreSQL DB インスタンスに接続するには

1. 証明書をダウンロードします。

   証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

2. SSL 経由の PostgreSQL DB インスタンスへの接続

   SSL を使用して接続するとき、クライアントでは証明書チェーンを検証するかどうかを選択できます。接続パラメータで `sslmode=verify-ca` または `sslmode=verify-full` を指定すると、RDS CA 証明書が信頼ストアに存在するか、または接続 URL で参照されることをクライアントは要求します。この要求は、データベース証明書に署名する証明書チェーンを検証するためのものです。

   psql や JDBC など、クライアントに SSL サポートが設定されている場合、クライアントは初期にデフォルトで SSL を使用してデータベースへの接続を試みます。クライアントが SSL を使用して接続できない場合、SSL を使用しない接続に戻ります。使用されるデフォルトの `sslmode` モードは、libpq ベースのクライアント (psql など) と JDBC では異なります。libpq ベースのクライアントおよび JDBC クライアントはデフォルトで `prefer` に設定されます。

   `sslrootcert` パラメータを使用して証明書を参照します (`sslrootcert=rds-ssl-ca-cert.pem` など)。

以下は、証明書検証で SSL を使用して PostgreSQL DB インスタンスに接続するために `psql` を使用する例です。

```
$ psql "host=db-name.555555555555.ap-southeast-1.rds.amazonaws.com
    port=5432 dbname=testDB user=testuser sslrootcert=rds-ca-rsa2048-g1.pem
 sslmode=verify-full"
```

PostgreSQL DB インスタンスへの SSL 接続を必須にする

rds.force_ssl パラメータを使用することで、PostgreSQL DB インスタンスへの接続に SSL の使用を必須にすることができます。RDS for PostgreSQL バージョン 15 以降では、rds.force_ssl パラメータのデフォルト値は 1 (オン) です。その他すべての RDS for PostgreSQL メジャーバージョン 14 以前では、このパラメータのデフォルト値は 0 (オフ) です。rds.force_ssl パラメータを 1 (オン) に設定すれば、DB クラスターへの接続で SSL/TLS を必須にすることができます。rds.force_ssl パラメータを 1 に設定すれば、DB インスタンスへの接続に SSL を必須にすることができます。

このパラメータの値を変更するには、カスタム DB パラメータグループを作成する必要があります。次に、カスタム DB パラメータグループ内で rds.force_ssl の値を 1 に変更して、この機能をオンにします。RDS for PostgreSQL DB インスタンスを作成する前にカスタム DB パラメータグループを準備する場合は、作成プロセス中に (デフォルトのパラメータグループではなく) そのパラメータグループを選択できます。RDS for PostgreSQL DB インスタンスが既に実行された後でこれを行う場合は、インスタンスがカスタムパラメータグループを使用するようにインスタンスを再起動する必要があります。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

DB インスタンスで rds.force_ssl 機能がアクティブになっている場合、SSL を使用していない接続試行は拒否され、次のメッセージが表示されます。

```
$ psql -h db-name.555555555555.ap-southeast-1.rds.amazonaws.com port=5432 dbname=testDB
 user=testuser
psql: error: FATAL: no pg_hba.conf entry for host "w.x.y.z", user "testuser", database
 "testDB", SSL off
```

SSL 接続ステータスを確認する

接続の暗号化ステータスは、DB インスタンスに接続するときにログオンバナーに表示されます。

```
Password for user master:
psql (10.3)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
postgres=>
```

また、sslinfo エクステンションをロードしてから、ssl_is_used() 関数を呼び出して、SSL が使用されているかどうかを調べることもできます。この関数は、この接続が SSL を使用している場合に t を返し、それ以外の場合に f を返します。

```
postgres=> CREATE EXTENSION sslinfo;
CREATE EXTENSION
postgres=> SELECT ssl_is_used();
ssl_is_used
---------
t
(1 row)
```

詳細については、次のクエリを使用して pg_settings から情報を取得できます。

```
SELECT name as "Parameter name", setting as value, short_desc FROM pg_settings WHERE
 name LIKE '%ssl%';
             Parameter name             |                  value                  |
                  short_desc
----------------------------------------+-----------------------------------------
+-------------------------------------------------------
 ssl                                    | on                                      |
 Enables SSL connections.
 ssl_ca_file                            | /rdsdbdata/rds-metadata/ca-cert.pem     |
 Location of the SSL certificate authority file.
 ssl_cert_file                          | /rdsdbdata/rds-metadata/server-cert.pem |
 Location of the SSL server certificate file.
 ssl_ciphers                            | HIGH:!aNULL:!3DES                       |
 Sets the list of allowed SSL ciphers.
 ssl_crl_file                           |                                         |
 Location of the SSL certificate revocation list file.
 ssl_dh_params_file                     |                                         |
 Location of the SSL DH parameters file.
 ssl_ecdh_curve                         | prime256v1                              |
 Sets the curve to use for ECDH.
 ssl_key_file                           | /rdsdbdata/rds-metadata/server-key.pem  |
 Location of the SSL server private key file.
 ssl_library                            | OpenSSL                                 |
 Name of the SSL library.
 ssl_max_protocol_version               |                                         |
 Sets the maximum SSL/TLS protocol version to use.
 ssl_min_protocol_version               | TLSv1.2                                 |
 Sets the minimum SSL/TLS protocol version to use.
 ssl_passphrase_command                 |                                         |
 Command to obtain passphrases for SSL.
 ssl_passphrase_command_supports_reload | off                                     |
 Also use ssl_passphrase_command during server reload.
```

```
 ssl_prefer_server_ciphers              | on                                      |
 Give priority to server ciphersuite order.
(14 rows)
```

また、次のクエリを使用して、RDS for PostgreSQL DB インスタンスの SSL 使用状況に関するすべての情報をプロセス、クライアント、およびアプリケーション別に収集することもできます。

```
SELECT datname as "Database name", usename as "User name", ssl, client_addr,
 application_name, backend_type
   FROM pg_stat_ssl
   JOIN pg_stat_activity
   ON pg_stat_ssl.pid = pg_stat_activity.pid
   ORDER BY ssl;
 Database name | User name | ssl |  client_addr   |    application_name    |
 backend_type
---------------+-----------+-----+----------------+------------------------
+------------------------------
               |           | f   |                |                        | autovacuum
 launcher
               | rdsadmin  | f   |                |                        | logical
 replication launcher
               |           | f   |                |                        | background
 writer
               |           | f   |                |                        |
 checkpointer
               |           | f   |                |                        | walwriter
 rdsadmin      | rdsadmin  | t   | 127.0.0.1      |                        | client
 backend
 rdsadmin      | rdsadmin  | t   | 127.0.0.1      | PostgreSQL JDBC Driver | client
 backend
 postgres      | postgres  | t   | 204.246.162.36 | psql                   | client
 backend
(8 rows)
```

SSL 接続に使用される暗号を識別するには、次のようにクエリを実行します。

```
postgres=> SELECT ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

sslmode オプションの詳細については、PostgreSQL ドキュメントの「Database connection control functions」を参照してください。

RDS for PostgreSQL の SSL 暗号スイート

PostgreSQL 設定パラメータ ssl_ciphers は、TLS 1.2 以下のバージョンを使用する際、SSL 接続で許可される暗号スイートのカテゴリを指定します。

RDS for PostgreSQL 16 以降では、許可リストに登録された暗号スイートから特定の値を使用するように ssl_ciphers パラメータを変更できます。これは、データベースインスタンスの再起動を必要としない動的パラメータです。許可リストに登録された暗号スイートを表示するには、Amazon RDS コンソールまたは次の AWS CLI コマンドを使用します。

```
aws rds describe-db-parameters --db-parameter-group-name <your-parameter-group> --
region <region> --endpoint-url <endpoint-url> --output json | jq '.Parameters[] |
 select(.ParameterName == "ssl_ciphers")'
```

次の表に、デフォルトの暗号スイートと、カスタム設定をサポートするバージョンの許可された暗号スイートの両方を示します。

PostgreSQL エンジンのバージョン	デフォルトの ssl_cipher スイート値	許可リストに登録されたカスタム ssl_cipher スイート値
18	HIGH:!aNULL:!3DES	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

PostgreSQL エンジンのバージョン	デフォルトの ssl_cipher スイート値	許可リストに登録されたカスタム ssl_cipher スイート値
17	HIGH:!aNULL:!3DES	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
16	HIGH:!aNULL:!3DES	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
15	HIGH:!aNULL:!3DES	カスタム ssl_ciphers はサポートされていません

PostgreSQL エンジンのバージョン	デフォルトの ssl_cipher スイート値	許可リストに登録されたカスタム ssl_cipher スイート値
14	HIGH:!aNULL:!3DES	カスタム ssl_ciphers はサポートされていません
13	HIGH:!aNULL:!3DES	カスタム ssl_ciphers はサポートされていません
12	HIGH:!aNULL:!3DES	カスタム ssl_ciphers はサポートされていません
11.4 以降のマイナーバージョン	HIGH:MEDIUM:+3DES:!aNULL:!RC4	カスタム ssl_ciphers はサポートされていません
11.1、11.2	HIGH:MEDIUM:+3DES:!aNULL	カスタム ssl_ciphers はサポートされていません
10.9 以降のマイナーバージョン	HIGH:MEDIUM:+3DES:!aNULL:!RC4	カスタム ssl_ciphers はサポートされていません
10.7 以前のマイナーバージョン	HIGH:MEDIUM:+3DES:!aNULL	カスタム ssl_ciphers はサポートされていません

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 暗号スイートを使用するようにすべてのインスタンス接続を設定するには、次の例に示すようにパラメータグループを変更します。

```
aws rds modify-db-parameter-group --db-parameter-group-name <your-parameter-group> --
parameters
 "ParameterName='ssl_ciphers',ParameterValue='TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384',ApplyMet
```

この例では ECDSA 暗号を使用しています。この方式では、接続を確立するために、インスタンスが楕円曲線暗号 (ECC) に対応した認証機関を使用する必要があります。Amazon RDS が提供する認証機関の詳細については、「認証機関」を参照してください。

デフォルトでは、RDS for PostgreSQL 16 以降の ssl_max_protocol_version の値は TLS v1.3 です。TLS v1.3 は、ssl_ciphers パラメータで指定された暗号設定を使用しません。「ssl_ciphers」で定義されている暗号のみを使用するには、ssl_max_protocol_version を TLS v1.2 に設定する必要があります。

```
aws rds modify-db-parameter-group --db-parameter-group-name <your-parameter-group> --
parameters
 "ParameterName='ssl_max_protocol_version',ParameterValue='TLSv1.2',ApplyMethod=immediate"
```

PostgreSQL 設定パラメータ ssl_tls13_ciphers は、TLS v1.3 を使用する際、データベースへの SSL 接続で許可される暗号スイートを指定します。

RDS for PostgreSQL 18 以降では、許可リストに登録された暗号スイートから特定の値を使用するように ssl_tls13_ciphers パラメータを変更できます。これは、データベースインスタンスの再起動を必要としない動的パラメータです。

次の表に、エンジンメジャーバージョン別の ssl_tls13_ciphers のデフォルト値、および許可リストに登録されたカスタム値を示します。

PostgreSQL エンジンのバージョン	**ssl_tls13_ciphers** のデフォルト値	許可リストに登録された **ssl_tls13_ciphers** のカスタム値
18	なし。PostgreSQL は、TLS ライブラリ (AWS-LC) のデフォルトの暗号スイートのセットを使用します。	TLS_AES_128_GCM_SHA256 TLS_AES_256_GCM_SHA384

使用中の暗号は、「SSL 接続ステータスの決定」で説明されている方法で確認できます。

暗号の名前はコンテキストによって異なる場合があります。

- パラメータグループで設定できる許可リストに登録された暗号は、その IANA 名で参照されます。
- sslinfo および psql ログオンバナーは、OpenSSL 名を使用した暗号を参照します。

TLS v1.3 の場合、OpenSSL 名と IANA 名は同じです。

データベース接続で SSL が使用されるようにするには、パラメータグループで rds.force_ssl parameter を 1 に設定します。パラメータとパラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

新しい SSL/TLS 証明書を使用して PostgreSQL DB インスタンスに接続するようにアプリケーションを更新する

Secure Socket Layer または Transport Layer Security (SSL/TLS) に使用される証明書には、通常、設定されたライフタイムがあります。サービスプロバイダーが認証局 (CA) 証明書を更新する際、クライアントでは、新しい証明書を使用するようにアプリケーションを更新する必要があります。以下で、クライアントアプリケーションが SSL/TLS を使用して、Amazon RDS for PostgreSQL DB インスタンスに接続されているかどうかの判別方法についての情報を参照できます。また、これらのアプリケーションが接続時にサーバーの証明書を検証しているかどうかを、確認する方法についても説明します。

> **Note**
>
> SSL/TLS 接続の前にサーバー証明書を検証するように構成されたクライアントアプリケーションでは、クライアントのトラストストアに有効な CA 証明書を持つ必要があります。新しい証明書が必要となった場合は、クライアントトラストストアも更新を行います。

クライアントアプリケーションの信頼ストアで CA 証明書を更新した後、DB インスタンスで証明書をローテーションできます。これらの手順は、非本番環境でテストしてから、本番環境で実装することを強くお勧めします。

証明書のローテーションの詳細については、「SSL/TLS 証明書のローテーション」を参照してください。証明書のダウンロードの詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。PostgreSQL DB インスタンスで SSL/TLS を使用する方法については、「PostgreSQL DB インスタンスで SSL を使用する」を参照してください。

トピック

- アプリケーションが SSL を使用して PostgreSQL DB インスタンスに接続しているかどうかの確認
- クライアントが接続するために証明書の検証を必要とするかどうかの確認
- アプリケーション信頼ストアの更新
- 各種アプリケーションでの SSL/TLS 接続の使用

アプリケーションが SSL を使用して PostgreSQL DB インスタンスに接続しているかどうかの確認

DB インスタンスの設定で rds.force_ssl パラメータの値を確認します。デフォルトでは、バージョン 15 より前のバージョンの PostgreSQL を使用する DB インスタンスの rds.force_ssl パラメータは 0 (オフ) に設定されています。デフォルトでは、rds.force_ssl は、PostgreSQL バージョン 15 以降のメジャーバージョンを使用する DB インスタンスでは1 (オン) に設定されています。rds.force_ssl パラメータが 1 (オン) に設定されている場合、クライアントは接続に SSL/TLS を使用する必要があります。パラメータグループの詳細については、「Amazon RDS のパラメータグループ」を参照してください。

RDS PostgreSQL バージョン 9.5 以降のメジャーバージョンを使用していて、rds.force_ssl が 1 (オン) に設定されていない場合、pg_stat_ssl ビューに対してクエリを実行して SSL を使用している接続をチェックします。例えば次のクエリは、SSL 接続、および SSL を使用するクライアントに関する情報のみを返します。

```
SELECT datname, usename, ssl, client_addr
  FROM pg_stat_ssl INNER JOIN pg_stat_activity ON pg_stat_ssl.pid =
 pg_stat_activity.pid
  WHERE ssl is true and usename<>'rdsadmin';
```

SSL/TLS 接続を使用する行のみが、接続に関する情報とともに表示されます。以下は出力例です。

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

このクエリでは、クエリ実行時点の接続のみが表示されます。結果が表示されなくても、SSL 接続を使用しているアプリケーションが存在しないわけではありません。それ以降に別の SSL 接続が確立される場合もあります。

クライアントが接続するために証明書の検証を必要とするかどうかの確認

psql や JDBC など、クライアントに SSL サポートが設定されている場合、クライアントは初期にデフォルトで SSL を使用してデータベースへの接続を試みます。クライアントが SSL を使用して接続できない場合、SSL を使用しない接続に戻ります。libpq ベースのクライアント (psql など)

と JDBC の両方で使用されるデフォルトの sslmode モードは、prefer に設定されています。サーバー上の証明書が検証されるのは、sslmode が verify-ca または verify-full に設定されて、sslrootcert が指定されている場合のみです。証明書が無効な場合は、エラーがスローされます。

PGSSLROOTCERT を使用して、PGSSLMODE 環境変数で証明書を検証します (PGSSLMODE は verify-ca または verify-full に設定)。

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U masteruser -d postgres
```

sslrootcert 引数を使用して、接続文字列形式で sslmode を使用して証明書を検証します (証明書を検証するには、sslmode は verify-ca または verify-full に設定します)。

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=masteruser dbname=postgres"
```

例えば前述の例の場合、無効なルート証明書を使用していると、クライアントで次のようなエラーが表示されます。

```
psql: SSL error: certificate verify failed
```

アプリケーション信頼ストアの更新

PostgreSQL アプリケーション信頼ストアの更新については、PostgreSQL ドキュメントの「Secure TCP/IP Connections with SSL」を参照してください。

ルート証明書のダウンロードについては、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。

証明書をインポートするサンプルスクリプトについては、証明書を信頼ストアにインポートするためのサンプルスクリプト を参照してください。

> **Note**
>
> 信頼ストアを更新するとき、新しい証明書を追加できるだけでなく、古い証明書を保持できます。

各種アプリケーションでの SSL/TLS 接続の使用

以下に、各種アプリケーションでの SSL/TLS 接続の使用に関する情報を示します。

- psql

  接続文字列または環境可変としてオプションを指定することで、クライアントがコマンドラインから呼び出されます。SSL/TLS 接続の場合、関連するオプションは `sslmode` (環境可変 `PGSSLMODE`)、`sslrootcert` (環境可変 `PGSSLROOTCERT`) です。

  オプションの詳細なリストについては、PostgreSQL ドキュメントの「Parameter Key Words」を参照してください。環境可変の詳細なリストについては、PostgreSQL ドキュメントの「Environment Variables」を参照してください。

- pgAdmin

  このブラウザベースのクライアントは、PostgreSQL データベースに接続するために使用できる使いやすいインターフェイスです。

  接続の設定については、pgAdmin のドキュメントを参照してください。

- JDBC

  JDBC は、Java アプリケーションとのデータベース接続を可能にします。

  JDBC を使用した PostgreSQL データベースへの接続に関する一般情報については、PostgreSQL JDBC ドライバードキュメントの「データベースへの接続」を参照してください。SSL/TLS を使用した接続については、PostgreSQL JDBC ドライバードキュメントの「クライアントの設定」を参照してください。

- Python:

  PostgreSQL データベースに接続するために一般的に使用される Python ライブラリは、`psycopg2` です。

  `psycopg2` の使用については、psycopg2 のドキュメントを参照してください。PostgreSQL データベースへの接続方法に関する簡単なチュートリアルについては、「Psycopg2 Tutorial」を参照してください。connect コマンドで受け入れられるオプションについては、「The psycopg2 module content」を参照してください。

Important

データベース接続で SSL/TLS を使用することを決定し、アプリケーションの信頼ストアを更新したら、rds-ca-rsa2048-g1 証明書を使用するようにデータベースを更新できます。ステップについては、「DB インスタンスまたはクラスターの変更による CA 証明書の更新」のステップ 3 を参照してください。

Amazon RDS for PostgreSQL で Kerberos 認証を使用する

ユーザーが PostgreSQL が実行されている DB インスタンスに接続する場合、Kerberos を使用してそのユーザーを認証できます。そのためには、Kerberos 認証に AWS Directory Service for Microsoft Active Directory を使用するように DB インスタンスを設定します。AWS Directory Service for Microsoft Active Directory は AWS Managed Microsoft AD とも呼ばれます。これは、Directory Service で利用できる機能です。詳細については、「AWS Directory Service 管理ガイド」の「Directory Service とは」を参照してください。

まず、ユーザー認証情報を格納する AWS Managed Microsoft AD ディレクトリを作成します。次に、Active Directory のドメインおよびその他の情報を PostgreSQL DB インスタンスに提供します。ユーザーが PostgreSQL DB インスタンスを使用して認証を実行すると、認証要求は AWS Managed Microsoft AD ディレクトリに転送されます。

同じディレクトリにすべての認証情報を保持することで時間と労力を節約できます。複数の DB インスタンスの認証情報を一元的に保存および管理できます。また、ディレクトリを使用することで、セキュリティプロファイル全体を向上できます。

また、独自のオンプレミスの Microsoft Active Directory から認証情報にアクセスできます。そのためには、信頼するドメイン関係を作成して、AWS Managed Microsoft AD ディレクトリがオンプレミスの Microsoft Active Directory を信頼するようにします。これにより、ユーザーは、オンプレミスネットワークのワークロードにアクセスするときと同じ Windows シングルサインオン (SSO) の使い方で、PostgreSQL インスタンスにアクセスできます。

データベースは、パスワード認証または、Kerberos 認証または AWS Identity and Access Management (IAM) 認証のいずれかによるパスワード認証を使用できます。IAM 認証の詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。

> **Note**
>
> RDS for PostgreSQL は、Active Directory グループの Kerberos 認証をサポートしていません。

トピック

- 利用可能なリージョンとバージョン
- PostgreSQL DB インスタンスの Kerberos 認証の概要
- PostgreSQL DB インスタンスの Kerberos 認証のセットアップ

- Active Directory ドメインの RDS for PostgreSQL DB インスタンスの管理
- PostgreSQL を Kerberos 認証と接続する

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。Kerberos 認証を使用した RDS for PostgreSQL のバージョンとリージョンの可用性の詳細については、「Amazon RDS での Kerberos データベース認証でサポートされているリージョンと DB エンジン」を参照してください。

PostgreSQL DB インスタンスの Kerberos 認証の概要

PostgreSQL DB インスタンスに Kerberos 認証を設定するには、以下で説明するステップを実行します。

1. AWS Managed Microsoft AD を使用して AWS Managed Microsoft AD ディレクトリを作成します。AWS マネジメントコンソール、AWS CLI、Directory Service API を使用して、ディレクトリを作成できます。ディレクトリがインスタンスと通信できるように、ディレクトリセキュリティグループで関連するアウトバウンドポートを必ず開いてください。
2. AWS Managed Microsoft AD ディレクトリを呼び出すためのアクセスを Amazon RDS に許可するロールを作成します。これにより、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する AWS Identity and Access Management (IAM) ロールが作成されます。

   IAM ロールによるアクセスを許可するには、AWS Security Token Service (AWS STS) エンドポイントを AWS アカウントの AWS リージョンでアクティベートする必要があります。AWS STS エンドポイントはすべての AWS リージョン でデフォルトでアクティブになっているため、他のアクションを実行せずに、エンドポイントを使用することができます。詳細については、IAM ユーザーガイドの「AWS STS リージョンでの AWS のアクティブ化と非アクティブ化」を参照してください。
3. Microsoft Active Directory のツールを使用して、AWS Managed Microsoft AD ディレクトリでユーザーとグループを作成し、設定します。Active Directory にユーザーを作成する方法の詳細については、AWS 管理ガイドの「Directory Service マネージド Microsoft AD でユーザーとグループを管理する」を参照してください。
4. 異なる AWS アカウントまたは Virtual Private Cloud (VPC) 内にディレクトリおよび DB インスタンスを配置する場合は、VPC ピア接続を設定します。詳細については、Amazon VPC Peering Guideの「VPC ピア機能とは」を参照してください。

5. 以下のいずれかの方法を使用して、コンソール、CLI、RDS API から PostgreSQL DB インスタンスを作成または変更します。
   - Amazon RDS DB インスタンスの作成
   - Amazon RDS DB インスタンスを変更する
   - DB インスタンスへの復元
   - Amazon RDS の DB インスタンスを特定の時点に復元する

   インスタンスは、ディレクトリと同じ Amazon Virtual Private Cloud (VPC)、または別の AWS アカウントまたは VPC にあります。PostgreSQL DB インスタンスの作成または変更時に、次のステップを行います。
   - ディレクトリの作成時に、生成されたドメイン識別子 (d-* 識別子) を指定します。
   - 作成した IAM ロール名を指定します。
   - DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからインバウンドトラフィックを受信できることを確認します。
6. RDS マスターユーザー認証情報を使用して、PostgreSQL DB インスタンスインスタンスに接続します。外部で識別されるように PostgreSQL でユーザーを作成します。外部で識別されたユーザーは、Kerberos 認証を使用して PostgreSQL DB インスタンスにログインできます。

PostgreSQL DB インスタンスの Kerberos 認証のセットアップ

AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) を使用して、PostgreSQL DB インスタンスに Kerberos 認証をセットアップします。Kerberos 認証をセットアップするには、次のステップに従います。

トピック

- ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する
- ステップ 2: (オプション) オンプレミスの Active Directory と Directory Service との間の信頼関係を作成する
- ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する
- ステップ 4: ユーザーを作成して設定する
- ステップ 5: ディレクトリと DB インスタンスの間のクロス VPC トラフィックを有効にする
- ステップ 6: PostgreSQL DB インスタンスを作成または変更する
- ステップ 7: Kerberos プリンシパル用の PostgreSQL ユーザーを作成する
- ステップ 8: PostgreSQL クライアントを設定する

ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する

Directory Service はフルマネージド型の Active Directory を AWS クラウド内に作成します。AWS Managed Microsoft AD ディレクトリを作成すると、Directory Service が 2 つのドメインコントローラーと DNS サーバーを作成します。ディレクトリサーバーは、VPC 内の異なるサブネットで作成されます。この冗長性によって、障害が発生してもディレクトリにアクセス可能な状態を維持できます。

AWS Managed Microsoft AD ディレクトリを作成すると、AWS Directory Service がユーザーに代わって次のタスクを実行します。

- VPC 内に Active Directory を設定します。
- ユーザー名 `Admin` と指定されたパスワードを使用してディレクトリ管理者アカウントを作成します。このアカウントを使用してディレクトリを管理します。

  > **Important**
  >
  > このパスワードは必ず保管してください。Directory Service にはこのパスワードは保存されず、復元やリセットもできません。

- ディレクトリコントローラー用セキュリティグループを作成します。セキュリティグループは、PostgreSQL DB インスタンスとの通信を許可する必要があります。

AWS Directory Service for Microsoft Active Directory を起動すると、AWS は組織単位 (OU) を作成します。OU にはディレクトリのオブジェクトがすべて含まれています。この OU はドメインルートにあります。OU にはディレクトリを作成する際に入力した NetBIOS 名があります。ドメインルートは AWS が所有し、管理します。

`Admin` ディレクトリに作成された AWS Managed Microsoft AD アカウントには、OU に対して頻繁に実行される管理行為の権限が含まれています。

- ユーザーを作成、更新、削除する
- ファイルやプリントサーバーなどのドメインにリソースを追加して、追加したリソースへのアクセス許可を OU のユーザーとグループに割り当てる
- 追加の OU やコンテナを作成する
- 権限を委譲する
- 削除されたオブジェクトを Active Directory のごみ箱から元に戻す

- Active Directory Web Service で Windows PowerShell 用の Active Directory と Domain Name Service (DNS) モジュールを実行する。

Admin アカウントには、ドメイン全体に関係するアクティビティを実行する権限もあります。

- DNS 設定 (レコード、ゾーン、フォワーダーの追加、削除、更新) を管理する
- DNS イベントログを参照する
- セキュリティイベントログを参照する

AWS Managed Microsoft AD でディレクトリを作成するには

1. Directory Serviceコンソールのナビゲーションペインで、[ディレクトリ]、[ディレクトリのセットアップ] の順に選択します。
2. AWS Managed Microsoft AD を選択します。現在、 Amazon RDS での使用では AWS Managed Microsoft AD のオプションのみがサポートされています。
3. [次へ] を選択します。
4. [ディレクトリ情報の入力] ページに、以下の情報を指定します。

   エディション

   目的の要件を満たすエディションを選択します。

   ディレクトリの DNS 名

   ディレクトリの完全修飾名 (例: **corp.example.com**)。

   ディレクトリの NetBIOS 名

   ディレクトリの短縮名 (例: CORP)。

   ディレクトリの説明

   必要に応じて、ディレクトリの説明。

   Admin パスワード

   ディレクトリ管理者のパスワードです。ディレクトリの作成プロセスでは、ユーザー名 Admin とこのパスワードを使用して管理者アカウントが作成されます。

ディレクトリ管理者のパスワードには、「admin」の単語を含めることはできません。パスワードは大文字と小文字を区別し、8-64 文字にします。また、以下の 4 つのカテゴリうち 3 つから少なくとも 1 文字を含める必要があります。

- 小文字 (a ~ z)
- 大文字 (A ~ Z)
- 数字 (0 ~ 9)
- 英数字以外の文字 (~!@#$%^&*_-+=`|\(){}[]:;'"<>,.?/)

パスワードを確認

管理者のパスワードをもう一度入力します。

> **Important**
>
> このパスワードは必ず保管してください。Directory Service にはこのパスワードは保存されず、復元やリセットもできません。

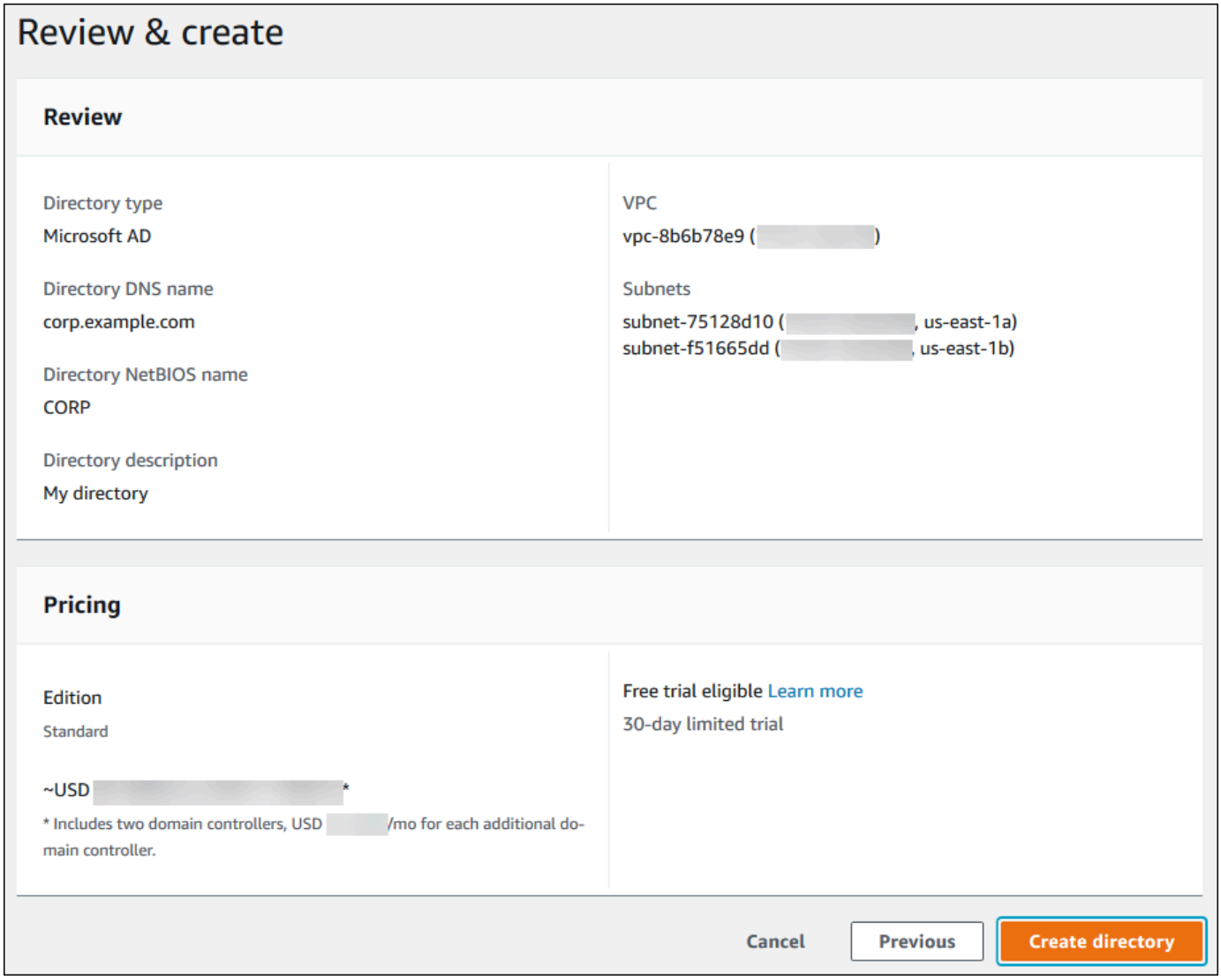
5. [Next (次へ)] を選択します。
6. [VPC とサブネットの選択] ページで、以下の情報を指定します。

   VPC

   ディレクトリ用の VPC を選択します。PostgreSQL DB インスタンスは、この同じ VPC または異なる VPC で作成できます。

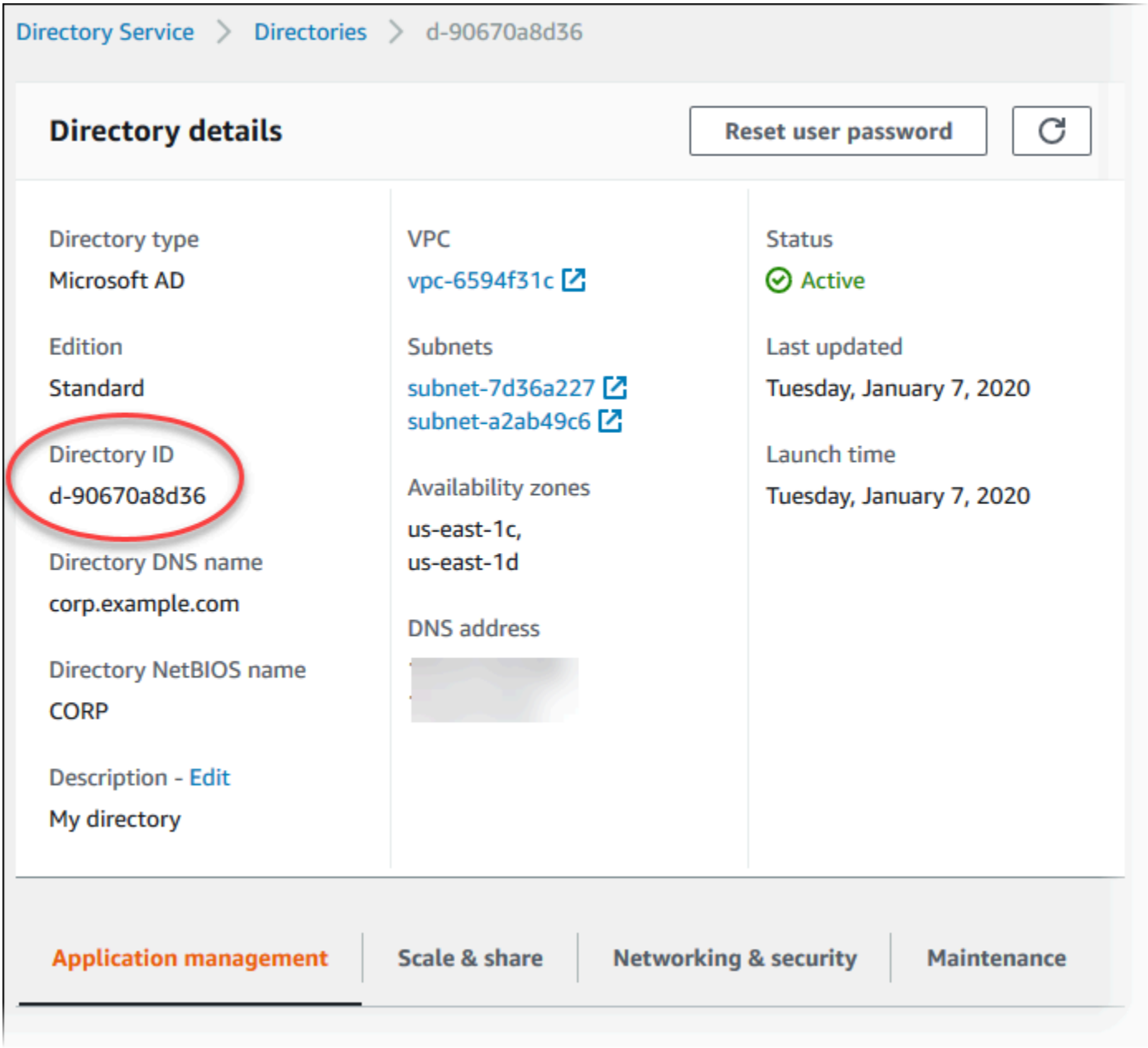
   Subnets

   ディレクトリサーバーのサブネットを選択します。2 つのサブネットは、異なるアベイラビリティーゾーンに存在している必要があります。
7. [Next (次へ)] を選択します。
8. ディレクトリの情報を確認します。変更が必要な場合は、[戻る] を選択し、変更を行います。情報が正しい場合は、[Create directory (ディレクトリの作成)] を選択します。



ディレクトリが作成されるまで、数分かかります。正常に作成されると、[Status] 値が [Active] に変わります。

ディレクトリに関する情報を表示するには、ディレクトリの一覧で、そのディレクトリ ID を選択します。ディレクトリ ID 値を書き留めます。PostgreSQL DB インスタンスを作成または変更する場合は、この値が必要です。



ステップ 2: (オプション) オンプレミスの Active Directory と Directory Service との間の信頼関係を作成する

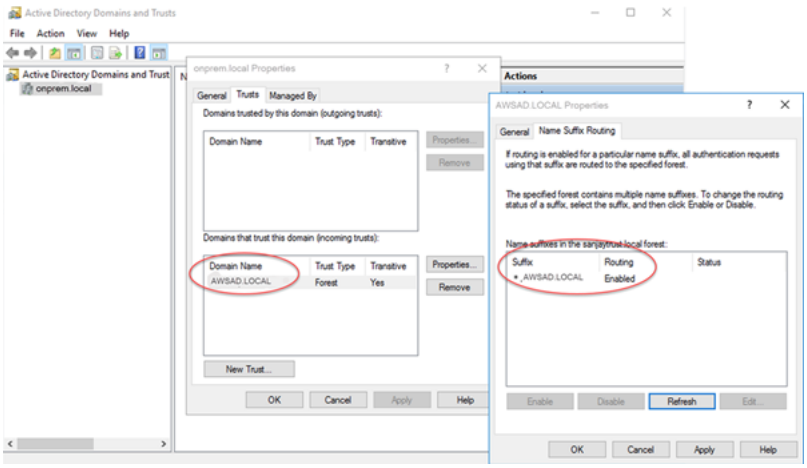
独自のオンプレミスの Microsoft Active Directory を使用する予定がない場合は、ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する に進みます。

オンプレミスの Active Directory を使用して Kerberos 認証を取得するには、オンプレミスの Microsoft Active Directory と (AWS Managed Microsoft AD で作成された) ステップ 1: AWS Managed Microsoft AD を使用してディレクトリを作成する ディレクトリとの間に、フォレストの信頼を使用して、信頼するドメイン関係を作成する必要があります。信頼は一方向にすることができます。この

場合、AWS Managed Microsoft AD ディレクトリはオンプレミスの Microsoft Active Directory を信頼します。信頼は、両方の Active Directory が相互に信頼する双方向にすることもできます。Directory Service を使用して信頼関係を設定する方法の詳細については、「AWS Directory Service 管理ガイド」の「信頼関係を作成する場合」を参照してください。

> **Note**
>
> オンプレミスの Microsoft Active Directory を使用している場合、Windows クライアントは rds.amazonaws.com の代わりにエンドポイント内の Directory Service のドメイン名を使用して接続します。詳細については PostgreSQL を Kerberos 認証と接続するを参照してください。

オンプレミスの Microsoft Active Directory ドメイン名に、新しく作成された信頼関係に対応する DNS サフィックスルーティングが含まれていることを確認してください。次のスクリーンショットは、例を示しています。



ステップ 3: Amazon RDS が Directory Service にアクセスするための IAM ロールを作成する

Amazon RDS が Directory Service を呼び出すには、AWS アカウントにマネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する IAM ロールが必要です。このロールにより、Amazon RDS は Directory Service を呼び出すことが可能になります。

AWS マネジメントコンソール を使用して DB インスタンスを作成し、コンソールユーザーが `iam:CreateRole` アクセス許可を持っている場合、コンソールは必要な IAM ロールを自動的に作成します。この場合、ロール名は `rds-directoryservice-kerberos-access-role` です。それ以外の場合は、IAM ロールを手動で作成する必要があります。IAM ロー

ルを作成する場合、[Directory Service] を選択し、それに AWS マネージドポリシー AmazonRDSDirectoryServiceAccess をアタッチします。

サービス用の IAM ロールを作成する方法の詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Note**
>
> RDS for Microsoft SQL Server の Windows 認証に使用される IAM ロールは、Amazon RDS for PostgreSQL に使用できません。

AmazonRDSDirectoryServiceAccess マネージドポリシーを使用する代わりに、必要なアクセス許可を使用してポリシーを作成することもできます。これを行うには、IAM ロールに次の IAM 信頼ポリシーが必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

また、ロールには、以下の IAM ロールポリシーも必要です。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

オプトインの AWS リージョンの場合は、IAM ロールの信頼ポリシーでリージョン固有のサービスプリンシパルを使用します。これらのリージョンでサービスの信頼ポリシーを作成するときは、サービスプリンシパルでリージョンコードを指定します。

次の例では、リージョン固有のサービスプリンシパルを含む信頼ポリシーを示します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
```

REGION-CODE を、特定のリージョンのコードに置き換えます。例えば、アジアパシフィック (メルボルン) リージョンでは、次のサービスプリンシパルを使用します。

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

イスラエル (テルアビブ) より後に開設された AWS リージョン (カナダ西部 (カルガリー) 以降) では、エンジン固有のサービスプリンシパル directoryservice-*engine*.rds.*region-code*.amazonaws.com を使用します。ここで、*engine* は、postgresql です。例えば、カナダ西部 (カルガリー) リージョンの場合。

```
"Service": [
  "directoryservice-postgresql.rds.ca-west-1.amazonaws.com",
  "rds.ca-west-1.amazonaws.com"
]
```

ステップ 4: ユーザーを作成して設定する

Active Directory ユーザーとコンピューターツールを使用してユーザーを作成できます。これは Active Directory Domain Services ツールおよび Active Directory Lightweight Directory Services ツールの 1 つです。詳細については、Microsoft のドキュメントの「ユーザーとコンピュータを Active Directory ドメインに追加する」を参照してください。この場合、ユーザーは個人またはその他のエンティティです。例えば、ドメインの一部であり、その ID がディレクトリで管理されているコンピュータなどです。

Directory Service ディレクトリにユーザーを作成するには、Directory Service ディレクトリのメンバーである Windows ベースの Amazon EC2 インスタンスに接続している必要があります。同時に、ユーザーを作成する権限を持つユーザーとしてログインしていなければなりません。詳細については、AWS Directory Service 管理ガイドの「ユーザーの作成」を参照してください。

ステップ 5: ディレクトリと DB インスタンスの間のクロス VPC トラフィックを有効にする

同じ VPC 内にディレクトリおよび DB インスタンスを配置する場合は、このステップをスキップして ステップ 6: PostgreSQL DB インスタンスを作成または変更する に進みます。

ディレクトリと DB インスタンスを別の VPC に配置する場合は、VPC ピア接続または AWS Transit Gateway を使用してクロス VPC トラフィックを設定します。

次の手順では、VPC ピア接続を使用して VPC 間のトラフィックを有効にします。Amazon Virtual Private Cloud ピアリング接続ガイドの「VPC ピア機能とは」の手順に従います。

VPC ピア接続を使用してクロス VPC トラフィックを有効にするには

1. 適切な VPC ルーティングを設定し、ネットワークトラフィックが双方向にフローするようにします。
2. DB インスタンスのセキュリティグループが、ディレクトリのセキュリティグループからインバウンドトラフィックを受信できることを確認します。
3. トラフィックをブロックするネットワークのアクセス制御リスト (ACL) ルールがないことを確認します。

別の AWS アカウントがディレクトリを所有している場合は、ディレクトリを共有する必要があります。

AWS アカウント間でディレクトリを共有するには

1. DB インスタンスを作成する AWS アカウントとの間でディレクトリの共有をスタートするには、AWS 管理ガイドの「チュートリアル: Directory Service マネージド Microsoft AD ディレクトリを共有して、シームレスに EC2 ドメインを結合する」の手順を実行します。
2. DB インスタンスのアカウントを使用して、Directory Service コンソールにサインインし、続行する前にドメインが必ず SHARED ステータスであることを確認します。
3. DB インスタンスのアカウントを使用して Directory Service コンソールにサインインしている間に、[ディレクトリ ID] の値を書き留めておきます。このディレクトリ ID は、DB インスタンスをドメインに結合するために使用します。

ステップ 6: PostgreSQL DB インスタンスを作成または変更する

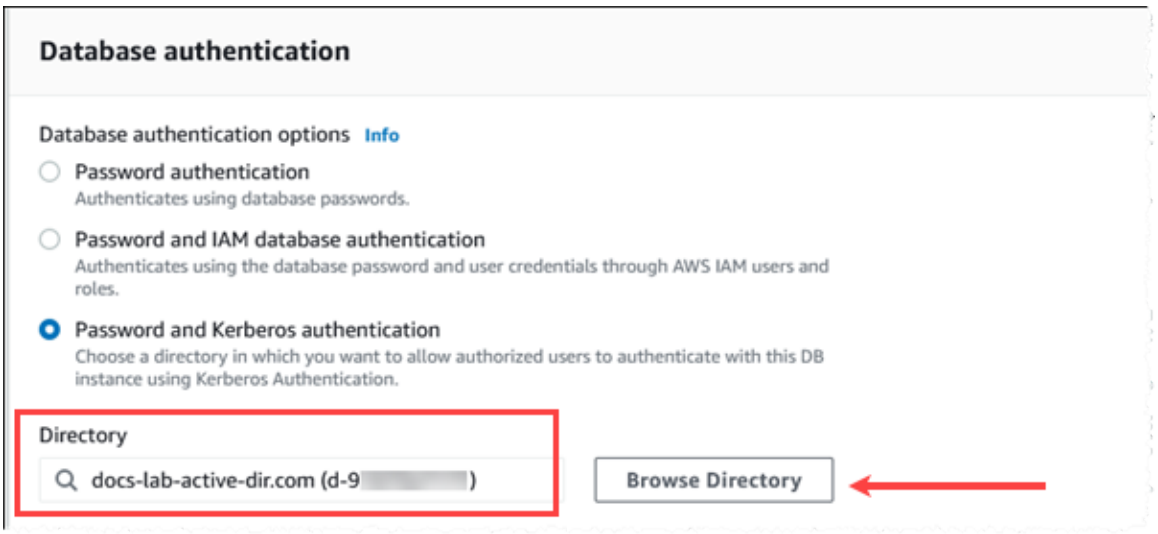
ディレクトリで使用する PostgreSQL DB インスタンスを作成または変更します。コンソール、CLI、RDS API を使用して DB インスタンスとディレクトリを関連付けることができます。これには以下の 2 つの方法があります。

- コンソール、create-db-instance CLI コマンド、または CreateDBInstance RDS API オペレーションを使用して新しい PostgreSQL DB インスタンスを作成します。手順については、「Amazon RDS DB インスタンスの作成」を参照してください。
- コンソール、modify-db-instance CLI コマンド、または ModifyDBInstance RDS API オペレーションを使用して、既存の PostgreSQL DB インスタンスを変更します。手順については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- コンソール、DB スナップショットから DB インスタンスを復元する CLI コマンド、または RestoreDBInstanceFromDBSnapshot RDS API オペレーションを使用して、DB スナップショットから PostgreSQL DB インスタンスを復元します。手順については、「DB インスタンスへの復元」を参照してください。
- コンソール、restore-db-instance-to-point-in-time CLI コマンド、または RestoreDBInstanceToPointInTime RDS API オペレーションを使用して、PostgreSQL インスタンスをポイントインタイムに復元します。手順については、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

Kerberos 認証は、VPC 内の PostgreSQL DB インスタンスでのみサポートされています。DB インスタンスは、ディレクトリと同じ VPC または異なる VPC 内にあります。DB インスタンスがディレクトリと通信する場合、そのインスタンスは、ディレクトリの VPC 内での送受信を許可するセキュリティグループを使用する必要があります。

コンソール

DB インスタンスを作成、変更、復元するためにコンソールを使用する場合は、データベースの認証セクションの [パスワードと Kerberos 認証] を選択します。次に、[ディレクトリのブラウジング] を選択します。Directory Service を使用するには、ディレクトリを選択するか、[新しいディレクトリの作成] を選択します。



AWS CLI

AWS CLI を使用する場合は、DB インスタンスが、作成したディレクトリを使用できるように、以下のパラメータが必要です。

- `--domain` パラメータには、ディレクトリの作成時に生成されたドメイン識別子 ("d-*" 識別子) を使用します。
- `--domain-iam-role-name` パラメータには、マネージド IAM ポリシー `AmazonRDSDirectoryServiceAccess` を使用する作成済みのロールを使用します。

例えば、以下の CLI コマンドはディレクトリを使用するように DB インスタンスを変更します。

```
aws rds modify-db-instance --db-instance-identifier mydbinstance --domain d-Directory-ID --domain-iam-role-name role-name
```

> **Important**
>
> DB インスタンスを変更して Kerberos 認証を有効にした場合、変更後、その DB インスタンスを再起動します。

ステップ 7: Kerberos プリンシパル用の PostgreSQL ユーザーを作成する

この時点で、 RDS for PostgreSQL DB インスタンスが AWS Managed Microsoft AD ドメインに参加しました。ステップ 4: ユーザーを作成して設定する のディレクトリに作成したユーザーを PostgreSQL データベースユーザーとして設定し、データベースにログインする権限を付与する必要

があります。そのためには、rds_superuser 権限を持つデータベースユーザーとしてサインインします。例えば、RDS for PostgreSQL DB インスタンス、の作成時にデフォルト値を受け入れた場合は、次のステップに示すように postgres を使用します。

Kerberos プリンシパル用の PostgreSQL データベースユーザーを作成するには

1. psql を使用して、の RDS for PostgreSQL DB インスタンスのエンドポイントに psql を使用して接続します。次の例では、postgres ロールにデフォルトの rds_superuser アカウントを使用しています。

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. データベースにアクセスする Kerberos プリンシパル (Active Directory ユーザー名) ごとにデータベースユーザー名を作成します。Active Directory インスタンスで定義されている正規のユーザー名 (ID) を使用します。つまり、そのユーザー名には、小文字 alias (Active Directory 内のユーザー名) と Active Directory ドメインの大文字の名前を使用します。Active Directory ユーザー名は外部認証されたユーザーなので、次に示すように名前を引用符で囲んでください。

```
postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
CREATE ROLE
```

3. データベースユーザーに rds_ad ロールを付与します。

```
postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
GRANT ROLE
```

Active Directory ユーザー ID のすべての PostgreSQL ユーザーの作成が完了すると、ユーザーは Kerberos 認証情報を使用して RDS for PostgreSQL DB インスタンス にアクセスできます。

Kerberos を使用して認証するデータベースユーザーは、Active Directory ドメインのメンバーであるクライアントマシンから認証を行う必要があります。

rds_ad ロールを付与されたデータベースユーザーもその rds_iam ロールを持つことはできません。これは、ネストされたメンバーシップにも適用されます。詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。

ステップ 8: PostgreSQL クライアントを設定する

PostgreSQL クライアントを設定するには、次のステップを実行します。

- ドメインを指す krb5.conf ファイル (または同等) を作成します。
- クライアントホストと Directory Service 間でトラフィックが流れることを確認します。次の目的で Netcat などのネットワークユーティリティを使用します。
  - ポート 53 の DNS 経由のトラフィックを確認します。
  - ポート 53 および Kerberos の TCP/UDP 上のトラフィックを確認します。これには、Directory Service の場合ポート 88 および 464 が含まれます。
- データベースポートを介してクライアントホストと DB インスタンス間でトラフィックが流れることを確認します。例えば、psql を使用してデータベースに接続し、アクセスします。

以下は、AWS Managed Microsoft AD 向けの krb5.conf の内容のサンプルです 。

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

以下は、オンプレミスの Microsoft Active Directory 向けの krb5.conf の内容のサンプルです。

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
```

```
.rds.amazonaws.com = EXAMPLE.COM
.amazonaws.com.cn = EXAMPLE.COM
.amazon.com = EXAMPLE.COM
```

Active Directory ドメインの RDS for PostgreSQL DB インスタンスの管理

コンソール、CLI、RDS API を使用して、 DB インスタンスと Microsoft Active Directory との関係を管理できます。例えば、Kerberos 認証を有効化するために、Microsoft Active Directory を関連付けることができます。また、Microsoft Active Directory の関連付けを解除して、Kerberos 認証を無効化することもできます。さらに、1 つの Microsoft Active Directory によって外部に認証される DB インスタンスをもう 1 つの Microsoft Active Directory に移動することもできます。

例えば、CLI を使用して次を実行できます。

- メンバーシップが失敗した Kerberos 認証を再度有効化するには、modify-db-instance CLI コマンドを使用します。`--domain` オプションに現在のメンバーシップのディレクトリ ID を指定します。
- DB インスタンスの Kerberos 認証を無効にするには、modify-db-instance CLI コマンドを使用します。`none` オプションで、`--domain` を指定します。
- DB インスタンスをあるドメインから別のドメインに移動するには、modify-db-instance CLI コマンドを使用します。`--domain` オプションの新しいドメインのドメイン識別子を指定します。

ドメインのメンバーシップを理解する

DB インスタンスを作成または変更すると、ドメインのそれがメンバーになります。DB インスタンスのドメインメンバーシップのステータスは、コンソールで表示したり、describe-db-instances CLI コマンドを実行して表示したりすることができます。DB インスタンスのステータスは、以下のいずれかです。

- `kerberos-enabled` - DB インスタンスは Kerberos 認証を有効化しました。
- `enabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を有効化中です。
- `pending-enable-kerberos` - この DB インスタンスでは、Kerberos 認証の有効化が保留中になっています。
- `pending-maintenance-enable-kerberos` - AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の有効化を試みます。
- `pending-disable-kerberos` - この DB インスタンスでは、Kerberos 認証の無効化が保留中になっています。

- `pending-maintenance-disable-kerberos` - AWS は、次にスケジュールされたメンテナンスウィンドウで、DB インスタンスでの Kerberos 認証の無効化を試みます。
- `enable-kerberos-failed` - 設定の問題により、AWS が DB インスタンスで Kerberos 認証を有効化できませんでした。DB インスタンスを変更するコマンドを再発行する前に、設定の問題を修正します。
- `disabling-kerberos` - AWS は、この DB インスタンスで Kerberos 認証を無効化中です。

ネットワーク接続の問題や正しくない IAM ロールのために、Kerberos 認証を有効化するリクエストは失敗する可能性があります。場合によっては、DB インスタンスを作成または変更するときに、Kerberos 認証を有効にしようとすると失敗する可能性があります。その場合、正しい IAM ロールを使用していることを確認してから、DB インスタンスを変更し、ドメインに接続します。

> **Note**
>
> PostgreSQL で RDS を使用する Kerberos 認証でのみ、ドメインの DNS サーバーにトラフィックが送信されます。他のすべての DNS リクエストは、PostgreSQL を実行している DB インスタンスでアウトバウンドのネットワークアクセスとして扱われます。RDS for PostgreSQL を使用したアウトバウンドネットワークアクセスの詳細については、「アウトバウンドネットワークアクセスでカスタム DNS サーバーを使用する」を参照してください。

PostgreSQL を Kerberos 認証と接続する

pgAdmin インターフェイスまたは psql などのコマンドラインインターフェイスを使用して、Kerberos 認証で PostgreSQL に接続できます。接続の詳細については、「PostgreSQL データベースエンジンを実行する DB インスタンスへの接続 」を参照してください。エンドポイント、ポート番号、および接続に必要なその他の詳細情報を取得する方法について詳細は、「PostgreSQL DB インスタンスへの接続」を参照してください。

> **Note**
>
> PostgreSQL の GSSAPI 認証と暗号化は、Kerberos ライブラリ `libkrb5.so` によって実装されます。`postgres_fdw` や `dblink` などの機能は、Kerberos 認証または暗号化によるアウトバウンド接続についても同じライブラリに依存します。

pgAdmin

pgAdmin を使用して、PostgreSQL を Kerberos 認証に接続するには、以下のステップを実行します。

1. クライアントコンピュータの pgAdmin アプリケーションを起動します。
2. [Dashboard] (ダッシュボード) タブで、[Add New Server] (新しいサーバーの追加) を選択します。
3. [Create - Server (作成 - サーバー)] ダイアログボックスで、[General (全般)] タブに名前を入力し、pgAdmin のサーバーを特定します。
4. [Connection] (接続) タブで、 RDS for PostgreSQL データベースから次の情報を入力します。
   - [Host] (ホスト) では、のエンドポイントを入力します。RDS for PostgreSQL DB インスタンス。エンドポイントは次のようになります。

     ```
     RDS-DB-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Windows クライアントからオンプレミスの Microsoft Active Directory に接続するには、ホストエンドポイントで `rds.amazonaws.com` の代わりに AWS Managed Active Directory のドメイン名を使用します。例えば、AWS Managed Active Directory のドメイン名が `corp.example.com` であるとします。この場合、[Host] (ホスト) では、エンドポイントは次のように指定されます。

     ```
     RDS-DB-instance.111122223333.aws-region.corp.example.com
     ```

   - [Port (ポート)] に、割り当てられたポートを入力します。
   - [Maintenance database (メンテナンスデータベース)] に、クライアントが接続する初期データベースの名前を入力します。
   - [ユーザーネーム] に、「ステップ 7: Kerberos プリンシパル用の PostgreSQL ユーザーを作成する」で Kerberos 認証用に入力したユーザーネームを入力します。
5. [保存] を選択します。

Psql

psql を使用して、PostgreSQL を Kerberos 認証に接続するには、以下のステップを実行します。

1. コマンドプロンプトで、次のコマンドを実行します。

   ```
   kinit username
   ```

username をユーザー名で置き換えます。プロンプトで Microsoft Active Directory に保存されているユーザーのパスワードを入力します。

2. PostgreSQL DB インスタンスがパブリックにアクセス可能な VPC を使用している場合は、EC2 クライアントの /etc/hosts ファイルに、DB インスタンスエンドポイントの IP アドレスを記述します。例えば、次のコマンドは IP アドレスを取得し、それを /etc/hosts ファイルに入れます。

```
% dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
;; Truncated, retrying in TCP mode.
ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
34.210.197.118

% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
hosts
```

   Windows クライアントからオンプレミスの Microsoft Active Directory を使用している場合は、特別なエンドポイントを使用して接続する必要があります。ホストエンドポイントで Amazon ドメイン rds.amazonaws.com を使用する代わりに、AWS Managed Active Directory のドメイン名を使用します。

   例えば、AWS Managed Active Directory のドメイン名が corp.example.com であるとします。次に、エンドポイントの形式 *PostgreSQL-endpoint*.*AWS-Region*.*corp.example.com* を使用して、これを /etc/hosts ファイルに配置します。

```
% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
hosts
```

3. 次の psql コマンドを使用して、Active Directory と統合されている PostgreSQL DB インスタンスにログインします。

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

   オンプレミスの Active Directory を使用して、Windows クライアントから PostgreSQL DB クラスターにログインするには、前のステップ (corp.example.com) のドメイン名を指定して次の psql コマンドを使用します。

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

アウトバウンドネットワークアクセスでカスタム DNS サーバーを使用する

RDS for PostgreSQL では、DB インスタンスでのアウトバウンドネットワークアクセスをサポートし、お客様が所有するカスタム DNS サーバーからのドメインネームサービス (DNS) 解決を許可します。カスタム DNS サーバーを介して、RDS for PostgreSQL DB インスタンスの完全修飾ドメイン名のみを解決できます。

トピック

- カスタム DNS 解決をオンにする
- カスタム DNS 解決をオフにする
- カスタム DNS サーバーのセットアップ

カスタム DNS 解決をオンにする

ユーザーの VPC で DNS 解決をオンにするには、まずカスタム DB パラメータグループを RDS for PostgreSQL インスタンスに関連付けます。次に、`rds.custom_dns_resolution` パラメータを 1 に設定してオンにしてから、変更を反映させるために DB インスタンスを再起動します。

カスタム DNS 解決をオフにする

ユーザーの VPC で DNS 解決をオフにするには、まずカスタム DB パラメータグループの `rds.custom_dns_resolution` パラメータを 0 に設定してオフにします。その後、変更を反映させるために DB インスタンスを再起動します。

カスタム DNS サーバーのセットアップ

カスタム DNS ネームサーバーを設定後、変更を DB インスタンスに反映させるまで約 30 分ほどかかります。DB インスタンスへの変更が反映されたら、すべてのアウトバウンドネットワークトラフィックのポート 53 の DNS サーバーにおいて DNS ルックアップクエリを行う必要があります。

> **Note**
>
> カスタム DNS サーバーを設定せず、`rds.custom_dns_resolution` が 1 に設定されている場合、ホストは Amazon Route 53 プライベートゾーンを使用して解決されます。詳細については、「プライベートホストゾーンの使用」を参照してください。

RDS for PostgreSQL DB インスタンスのカスタム DNS サーバーをセットアップするには

1. VPC にアタッチされた Dynamic Host Configuration Protocol (DHCP) オプションセットで、domain-name-servers オプションを DNS ネームサーバーの IP アドレスに設定します。詳細については、「DHCP オプションセット」を参照してください。

   > **Note**
   >
   > domain-name-servers オプションが許可する値は 4 つまでになりますが、Amazon RDS DB インスタンスが使用するのは初期の値のみです。

2. DNS サーバーが、パブリック DNS 名、Amazon EC2 プライベート DNS 名、ユーザー固有の DNS 名を含むすべてのルックアップクエリを解決できることを確認します。DNS サーバーが処理できない DNS ルックアップがアウトバウンドネットワークトラフィックにある場合は、状況に適したアップストリーミング DNS プロバイダを必ず設定してください。
3. 512 バイト以下の User Datagram Protocol (UDP) レスポンスを生成するように DNS サーバーを設定します。
4. 1,024 バイト以下の Transmission Control Protocol (TCP) レスポンスを生成するように DNS サーバーを設定します。
5. ポート 53 で Amazon RDS DB インスタンスからのインバウンドトラフィックを許可するように DNS サーバーを設定します。DNS サーバーが Amazon VPC にある場合、VPC にはポート 53 で UDP と TCP トラフィックを許可するインバウンドルールを含むセキュリティグループが必要になります。DNS サーバーが Amazon VPC にない場合は、ポート 53 で UDP と TCP インバウンドトラフィックを許可できるように、適切なファイアウォール設定が必要になります。

   詳細については、「VPC のセキュリティグループ」と「ルールの追加と削除」を参照してください。
6. ポート 53 でアウトバウンドトラフィックを許可するため、Amazon RDS DB インスタンスの VPC を設定します。VPC には、ポート 53 で UDP および TCP トラフィックを許可するアウトバウンドルールを含むセキュリティグループが必要になります。

   詳細については、「Amazon VPC ユーザーガイド」の「Security groups for your VPC」(VPC のセキュリティグループ) と「Adding and removing rules」(ルールの追加および削除) を参照してください。
7. Amazon RDS DB インスタンスと DNS サーバー間のルーティングパスが、DNS トラフィックを許可するように適切に設定されていることを確認してください。

Amazon RDS DB インスタンスと DNS サーバーが同じ VPC に存在しない場合は、それらの間でピアリング接続がセットアップされていることを確認してください。詳細については、「Amazon VPC ピアリングガイド」の「VPC ピアリングとは」を参照してください。

RDS for PostgreSQL DB エンジンのアップグレード

PostgreSQL データベースで管理できる 2 つのタイプのアップグレードがあります。

- OS の更新 - では、Amazon RDS は、セキュリティの修正や OS の変更を適用するために、データベースの基になるオペレーティングシステムの更新が必要になる場合があります。RDS コンソール、AWS Command Line Interface (AWS CLI)、または RDS API を使用して、Amazon RDS に OS の更新を適用するタイミングを指定できます。OS 更新接続の詳細については、「DB インスタンスへの更新の適用」を参照してください
- データベースエンジンの更新 - Amazon RDS が新バージョンのデータベースエンジンをサポートすると、データベースをその新バージョンにアップグレードできます。

このコンテキストのデータベースとは、RDS for PostgreSQL DB インスタンスまたはマルチ AZ DB クラスターです。

PostgreSQL データベースのエンジンアップグレードには、メジャーバージョンアップグレードとマイナーバージョンアップグレードの 2 種類があります。

メジャーバージョンのアップグレード

メジャーバージョンのアップグレードには、既存のアプリケーションとの下位互換性のないデータベースの変更が含まれる場合があります。そのため、データベースのメジャーバージョンアップグレードは手動で実行する必要があります。メジャーバージョンアップグレードをスタートするには、DB インスタンスまたはマルチ AZ DB クラスターを変更します。メジャーバージョンアップグレードを行う前に、「RDS for PostgreSQL のメジャーバージョンアップグレードの選択」で説明されているステップを実行することをお勧めします。

Amazon RDS は、次の方法でマルチ AZ メジャーバージョンのアップグレードを処理します。

- マルチ AZ DB インスタンスデプロイ – Amazon RDS はプライマリインスタンスとスタンバイインスタンスを同時にアップグレードします。アップグレードが完了するまで、データベースが数分間使用できない場合があります。
- マルチ AZ DB クラスターデプロイ – Amazon RDS はリーダーインスタンスとライターインスタンスを同時にアップグレードします。アップグレードが完了するまで、データベースが数分間使用できない場合があります。

リージョン内リードレプリカがある DB インスタンスをアップグレードする場合、Amazon RDS はプライマリ DB インスタンスと共にレプリカもアップグレードします。

Amazon RDS は、マルチ AZ DB クラスターのリードレプリカをアップグレードしません。マルチ AZ DB クラスターのメジャーバージョンアップグレードを実行すると、リードレプリカのレプリケーション状態が終了に変わります。アップグレードの完了後、リードレプリカを手動で削除し、再作成する必要があります。

> **Tip**
>
> ブルー/グリーンデプロイを使用することで、メジャーバージョンアップグレードに必要なダウンタイムを最小限に抑えることができます。詳細については、「データベースの更新にブルー/グリーンデプロイを使用する」を参照してください。

マイナーバージョンのアップグレード

それに対して、マイナーバージョンのアップグレードに含まれるのは、既存のアプリケーションとの下位互換性がある変更のみです。マイナーバージョンのアップグレードを手動で行うには、データベースを変更します。または、データベースの作成時または変更時に、[マイナーバージョン自動アップグレード] を有効にすることができます。これにより、Amazon RDS は、新しいバージョンがテストおよび承認されると、データベースを自動的にアップグレードします。

Amazon RDS は、マルチ AZ マイナーバージョンアップグレードを次の方法で処理します。

- マルチ AZ DB インスタンスデプロイ – Amazon RDS はプライマリインスタンスとスタンバイインスタンスを同時にアップグレードします。アップグレードが完了するまで、データベースが数分間使用できない場合があります。
- マルチ AZ DB クラスターデプロイ – Amazon RDS はリーダー DB インスタンスを一度に 1 つずつアップグレードします。その後、リーダー DB インスタンスの 1 つが新しいライター DB インスタンスに切り替わります。次に、Amazon RDS は、古いライターインスタンス (今ではリーダーインスタンス) をアップグレードします。マルチ AZ DB クラスターは、通常、マイナーバージョンアップグレードのダウンタイムを約 35 秒に短縮します。RDS Proxy と併用すると、ダウンタイムをさらに 1 秒以下に短縮できます。詳細については、「Amazon RDS Proxy 」を参照してください。または、ProxySQL、PgBouncer、AWS Advanced JDBC Wrapper Driver などのオープンソースデータベースプロキシを使用することもできます。

データベースに複数のリードレプリカがある場合は、ソースインスタンスまたはクラスターをアップグレードする前に、すべてのリードレプリカをアップグレードする必要があります。

詳細については、「RDS for PostgreSQL のマイナーバージョンの自動アップグレード」を参照してください。マイナーバージョンアップグレードの手動での実行に関する詳細は、「エンジンバージョンの手動アップグレード」を参照してください。

データベースエンジンのバージョン、およびデータベースエンジンのバージョンを廃止するためのポリシーの詳細については、Amazon RDS FAQ の「データベースエンジンのバージョン」を参照してください。

トピック

- PostgreSQL アップグレードに関する考慮事項
- 有効なアップグレードターゲットの検索
- PostgreSQL のバージョン番号
- RDS for PostgreSQL の RDS バージョン番号
- RDS for PostgreSQL のメジャーバージョンアップグレードの選択
- RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法
- RDS for PostgreSQL のマイナーバージョンの自動アップグレード
- RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード
- イベントによる RDS for PostgreSQL エンジンのアップグレードのモニタリング

PostgreSQL アップグレードに関する考慮事項

データベースを安全にアップグレードするために、Amazon RDS では、「PostgreSQL ドキュメント」で示されている pg_upgrade ユーティリティを使用します。

バックアップ保存期間が 0 を超える値の場合、Amazon RDS はアップグレードプロセス中に 2 つの DB スナップショットを収得します。初期の DB スナップショットは、アップグレードの変更が行われる前のデータベースから作成されます。アップグレードがデータベースに対して失敗した場合は、このスナップショットを復元して、以前のバージョンを実行するデータベースを作成できます。アップグレードの完了後に 2 番目の DB スナップショットが作成されます。これらの DB スナップショットは、バックアップ保持期間が終了すると自動的に削除されます。

> **Note**
>
> データベースのバックアップ保持期間を 0 より大きく設定した場合にのみ、Amazon RDS は、アップグレード中に DB スナップショットを作成します。DB インスタンスのバックアップ保持期間を変更するには、「the section called “DB インスタンスを変更する”」を参照してください。マルチ AZ DB クラスターのカスタムバックアップ保持期間を設定することはできません。

DB インスタンスのメジャーバージョンアップグレードを実行すると、任意のリージョン内リードレプリカも自動的にアップグレードされます。アップグレードワークフローがスタートされると、リードレプリカは、pg_upgrade がプライマリ DB インスタンスで正常に完了するまで待ちます。次に、プライマリ DB インスタンスのアップグレードは、リードレプリカのアップグレードが完了するまで待ちます。アップグレードが完了するまで停止が発生します。マルチ AZ DB クラスターののメジャーバージョンを実行すると、すべてのリードレプリカのレプリケーション状態が終了に変わります。

アップグレードが完了したら、DB エンジンの前のバージョンに戻すことはできません。前のバージョンに戻す必要がある場合は、アップグレードの前に作成された DB スナップショットを復元して、新しいデータベースを作成します。

有効なアップグレードターゲットの検索

AWS マネジメントコンソール を使用してデータベースをアップグレードする場合、データベースの有効なアップグレードターゲットが表示されます。次の AWS CLI コマンドを使用して、データベースの有効なアップグレードターゲットを特定することもできます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
  --engine postgres \
  --engine-version version-number \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine postgres ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

例えば、PostgreSQL バージョン 16.1 データベースの有効なアップグレードターゲットを特定するには、次の AWS CLI コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
  --engine postgres \
```

```
  --engine-version 16.1 \
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
  --engine postgres ^
  --engine-version 16.1 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

PostgreSQL のバージョン番号

PostgreSQL データベースエンジンのバージョン番号は、次のように付けられています。

- PostgreSQL バージョン 10 以降では、エンジンのバージョン番号はメジャー.マイナーの形式になります。メジャーバージョンの番号は、バージョン番号の整数の部分です。マイナーバージョンの番号は、バージョン番号の小数の部分です。

  メジャーバージョンのアップグレードでは、10.マイナーから 11.マイナーのように、バージョン番号の整数の部分が大きくなります。
- 10 より前のバージョンの PostgreSQL では、エンジンのバージョン番号はメジャー.メジャー.マイナーの形式になります。エンジンのメジャーバージョンの番号は、バージョン番号の整数と 1 つ目の小数の部分の両方です。例えば、9.6 はメジャーバージョンです。マイナーバージョンの番号は、バージョン番号の 3 つ目の部分です。例えば、バージョン 9.6.12 では、12 がマイナーバージョンの番号です。

  メジャーバージョンのアップグレードでは、バージョン番号の主要な部分が大きくなります。例えば、9.6.12 から 11.14 へのアップグレードはメジャーバージョンのアップグレードであり、9.6 と 11 はメジャーバージョン番号です。

RDS 延長サポートのバージョン番号付けの詳細については、「Amazon RDS 延長サポートバージョンの命名規則」を参照してください。

RDS for PostgreSQL の RDS バージョン番号

RDS バージョン番号は *major*.*minor*.*patch* 命名規則を使用します。RDS パッチバージョンには、リリース後にマイナーバージョンに追加された重要なバグ修正が含まれています。RDS 延長サ

ポートのバージョン番号付けの詳細については、「Amazon RDS 延長サポートバージョンの命名規則」を参照してください。

データベースの Amazon RDS バージョン番号を識別するには、まず次のコマンドを使用して rds_tools 拡張機能を作成する必要があります。

```
CREATE EXTENSION rds_tools;
```

PostgreSQL バージョン 15.2-R2 のリリース以降、次の SQL クエリを使用して RDS for PostgreSQL データベースの RDS バージョン番号を確認できるようになりました。

```
postgres=> SELECT rds_tools.rds_version();
```

例えば、RDS for PostgreSQL 15.2 データベースをクエリすると、次が返されます。

```
rds_version
----------------
 15.2.R2
(1 row)
```

RDS for PostgreSQL のメジャーバージョンアップグレードの選択

メジャーバージョンのアップグレードには、以前のバージョンのデータベースと下位互換性のない変更が含まれる場合があります。新しい機能により、既存のアプリケーションが適切に動作しなくなることがあります。このため、Amazon RDS では、メジャーバージョンアップグレードは自動的に適用されません。メジャーバージョンのアップグレードを行うには、データベースを手動で変更します。本稼働データベースにアップグレードを適用する前に、アップグレードを徹底的にテストしてアプリケーションが正常に動作することを確認してください。PostgreSQL メジャーバージョンアップグレードを行うには、「RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法」に記載されているステップを実施することをお勧めします。

PostgreSQL シングル AZ DB インスタンスまたはマルチ AZ DB インスタンス配置を次のメジャーバージョンにアップグレードすると、データベースに関連付けられている任意のリードレプリカも次のメジャーバージョンにアップグレードされます。場合によっては、アップグレード時に上位のメジャーバージョンにスキップできます。アップグレードでメジャーバージョンがスキップされると、リードレプリカもターゲットのメジャーバージョンにアップグレードされます。他のメジャーバージョンをスキップするバージョン 11 へのアップグレードには、特定の制限があります。詳細については、RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法 で説明する手順を参照してください。

PostgreSQL のほとんどの拡張機能は、PostgreSQL エンジンのアップグレード時にアップグレードされません。拡張機能は、個別にアップグレードする必要があります。詳細については、「RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード」を参照してください。

次の AWS CLI クエリを実行すると、RDS for PostgreSQL データベースで利用できるメジャーバージョンを確認できます。

```
aws rds describe-db-engine-versions --engine postgres  --engine-version your-version
 --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" --
output text
```

使用可能なすべてのバージョンに対するこのクエリの結果の概要を次の表に示します。バージョン番号のアスタリスク (*) は、そのバージョンのサポートが終了したことを示します。現在のバージョンがサポートされていない場合は、最新のマイナーバージョンのアップグレードターゲットにアップグレードするか、そのバージョンで利用可能な他のアップグレードターゲットにアップグレードすることをお勧めします。

現在のソースバージョン	アップグレードターゲット
18.3	なし
18.2*	18.3
18.1	18.3
17.9	18.3、18.1
17.8	18.3、18.1 17.9
17.7	18.3、18.1 17.9
17.6	18.3、18.1 17.9、17.7
17.5	18.3、18.1

現在のソースバージョン	アップグレードターゲット
	17.9、17.7、17.6
17.4	18.3、18.1 17.9、17.7、17.6、17.5
17.3*、17.2	18.3、18.1 17.9、17.7、17.6、17.5、17.417.4
17.1*	17.6、17.5、17.4、17.2
16.12*、16.13	18.3 17.9
16.11	18.3、18.1 17.9、17.7 16.13
16.10	18.3、18.1 17.9、17.7 16.13、16.11
16.9	18.3、18.1 17.9、17.7、17.6、17.5 16.13、16.11、16.10
16.8	18.3、18.1 17.9、17.7、17.6、17.5、17.417.4 16.13、16.11、16.10、16.9

現在のソースバージョン	アップグレードターゲット
16.7*	17.6、17.5 、17.4 16.10、16.9、16.8
16.6	18.3、18.1 17.9、17.7、17.6、17.5、17.4、17.2 16.13、16.11、16.10、16.9、16.816.8
16.5*、16.4*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6
16.3*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.416.4
16.2*、16.1*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3
15.17、15.16*	18.3 17.9 16.13
15.15	18.3、18.1 17.9、17.7 16.13、16.11 15.17

現在のソースバージョン	アップグレードターゲット
15.14	18.3、18.1 17.9、17.7、17.6 16.13、16.11、16.10 15.17、15.15
15.13	18.3、18.1 17.9、17.7、17.6、17.5 16.13、16.11、16.10、16.9 15.17、15.15、15.14
15.12、15.11*	18.3、18.1 17.9、17.7、17.6、17.5、17.417.4 16.13、16.11、16.10、16.9、16.816.8 15.17、15.15、15.14、15.13
15.10	18.3、18.1 17.9、17.7、17.6、17.5、17.4、17.2 16.13、16.11、16.10、16.9、16.8、16.616.6 15.17、15.15、15.14、15.13、15.12
15.9*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6 15.14、15.13、15.12、15.10

現在のソースバージョン	アップグレードターゲット
15.8*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.416.4 15.14、15.13、15.12、15.10
15.7*	16.9、16.8、16.7、16.6、16.5、16.4、16.3, 16.3 15.13、15.12、15.11、15.10、15.9、15.815.8
15.6*、15.5*、15.4*、15.3*、15.2*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7
14.22、14.21*	18.3 17.9 16.13 15.17
14.20	18.3、18.1 17.9、17.7 16.13、16.11 15.17、15.15 14.22

現在のソースバージョン	アップグレードターゲット
14.19	18.3、18.1 17.9、17.7、17.6 16.13、16.11、16.10 15.17、15.15、15.14 14.22、14.20
14.18	18.3、18.1 17.9、17.7、17.6、17.5 16.13、16.11、16.10、16.9 15.17、15.15、15.14、15.13 14.22、14.20、14.19
14.17、14.16*	18.3、18.1 17.9、17.7、17.6、17.5、17.417.4 16.13、16.11、16.10、16.9、16.816.8 15.17、15.15、15.14、15.13、15.12 14.22、14.20、14.19、14.18
14.15	18.3、18.1 17.9、17.7、17.6、17.5、17.4、17.2 16.13、16.11、16.10、16.9、16.8、16.616.6 15.17、15.15、15.14、15.13、15.12、15.1015.10 14.22、14.20、14.19、14.18、14.17

現在のソースバージョン	アップグレードターゲット
14.14*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6 15.14、15.13、15.12、15.10 14.19、14.18、14.17、14.15
14.13*	16.4 15.13、15.12、15.11、15.10、15.9、15.815.8 14.18、14.17、14.16、14.15、14.14
14.12*	16.3 15.13、15.12、15.11、15.10、15.9、15.8、15.7, 15.7 14.18、14.17、14.16、14.15、14.14、14.1314.13
14.11*、14.10*、14.9*、14.8*、14.7*、14.6*、14.5*、14.4*、14.3*、14.2*、14.1*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7 14.19、14.18、14.17、14.15、14.13、14.1214.12
13.22	17.6 16.10 15.14 14.19

現在のソースバージョン	アップグレードターゲット
13.21	17.6、17.5 16.10、16.9 15.14、15.13 14.19、14.18 13.22
13.20、13.19*	17.6、17.5、17.4 16.10、16.9、16.8 15.14、15.13、15.12 14.19、14.18、14.17 13.22、13.21
13.18、13.17*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6 15.14、15.13、15.12、15.10 14.19、14.18、14.17、14.15 13.22、13.21、13.20
13.16	16.4 15.8 14.18、14.17、14.15、14.14、14.13 13.21、13.20、13.19、13.18、13.17

現在のソースバージョン	アップグレードターゲット
13.15	16.3 15.8、15.7 14.18、14.17、14.15、14.14、14.13、14.1214.12 13.21、13.20、13.19、13.18、13.17、13.1613.16
13.14*、13.13*、13.12*、13.11*、13.10*、13.9*、13.8*、13.7*、13.6*、13.5*、13.4*、13.3*、13.2*、13.1*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7 14.19、14.18、14.17、14.15、14.13、14.1214.12 13.22、13.21、13.20、13.18、13.16、13.1513.15
12.22-rds.20250508	17.6、17.5 16.10、16.9 15.14、15.13 14.19、14.18 13.22、13.21
12.22-rds.20250220	17.6、17.5、17.4 16.10、16.9、16.8 15.14、15.13、15.12 14.19、14.18、14.17 13.22、13.21、13.20 12.22-rds.20250508

現在のソースバージョン	アップグレードターゲット
12.22、12.21*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6 15.14、15.13、15.12、15.10 14.19、14.18、14.17、14.15 13.22、13.21、13.20、13.18 12.22-rds.20250220、12.22-rds.20250508
12.20*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.416.4 15.14、15.13、15.12、15.10、15.815.8 14.19、14.18、14.17、14.15、14.13 13.22、13.21、13.20、13.18、13.16 12.22、12.22-rds.20250220、12.22-rds.20250508
12.19*、12.18*、12.17*、12.16*、12.15*、12.14*、12.13*、12.12*、12.11*、12.10*、12.9*、12.8*、12.7*、12.6*、12.5*、12.4*、12.3*、12.2*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7 14.19、14.18、14.17、14.15、14.13、14.1214.12 13.22、13.21、13.20、13.18、13.16、13.1513.15 12.22、12.22-rds.20250220、12.22-rds.20250508

現在のソースバージョン	アップグレードターゲット
11.22-rds.20250508	17.6、17.5 16.10、16.9 15.14、15.13 14.19、14.18 13.22、13.21 12.22-rds.20250508
11.22-rds.20250220	17.6、17.5、17.4 16.10、16.9、16.8 15.14、15.13、15.12 14.19、14.18、14.17 13.22、13.21、13.20 12.22-rds.20250220、12.22-rds.20250508 11.22-rds.20250508

現在のソースバージョン	アップグレードターゲット
11.22-rds.20240509	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7 14.19、14.18、14.17、14.15、14.13、14.1214.12 13.22、13.21、13.20、13.18、13.16、13.1513.15 12.22、12.22-rds.20250220、12.22-rds.20250508 11.22-rds.20240808、11.22-rds.20241121、11.22-rds.20250220、11.22-rds.20250508
11.22、11.21*、11.20*、11.19*、11.18*、11.17*、11.16*、11.15*、11.14*、11.13*、11.12*、11.11*、11.10*、11.9*、11.8*、11.7*、11.6*、11.5*、11.4*、11.2*、11.1*	17.6、17.5、17.4、17.2 16.10、16.9、16.8、16.6、16.4、16.316.3 15.14、15.13、15.12、15.10、15.8、15.715.7 14.19、14.18、14.17、14.15、14.13、14.1214.12 13.22、13.21、13.20、13.18、13.16、13.1513.15 12.22、12.22-rds.20250220、12.22-rds.20250508 11.22-rds.20240418、11.22-rds.20240509、11.22-rds.20240808、11.22-rds.20241121、11.22-rds.20250220、11.22-rds.20250508

* このバージョンは現在サポートされていません。

RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法

Amazon RDS for PostgreSQL データベースでメジャーバージョンのアップグレードを実行する場合は、以下のプロセスをお勧めします。

1. バージョンとの互換性を持つパラメータグループの準備 - カスタムパラメータグループを使用している場合は、2 つのオプションがあります。新しい DB エンジンバージョンのデフォルトのパラメータグループを指定します。または、新しい DB エンジンバージョンの独自のカスタムパラメータグループを作成します。詳細については、「the section called “パラメータグループ”」および「the section called “DB クラスターのパラメータグループ”」を参照してください。
2. サポートされていないデータベースクラスを確認する - データベースのインスタンスクラスが、アップグレード先の PostgreSQL バージョンと互換性があることを確認します。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。
3. サポートされていない使用の確認
   - 準備済みのトランザクション - アップグレードを実行する前に、すべての準備済みのトランザクションをコミットまたはロールバックします。

     次のクエリを使用して、開いている準備済みのトランザクションがデータベースにないことを確認します。

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

   - Reg* データ型 - アップグレードの実施前に reg* データ型の使用をすべて削除します。regtype と regclass を除き、reg* データ型をアップグレードすることはできません。このデータ型はアップグレードで使用されているため、 pg_upgrade ユーティリティで維持することはできません。

     サポートされていない reg* データ型が使用されていないことを確認するには、データベースごとに次のクエリを使用します。

     ```
     SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
      pg_catalog.pg_attribute a
       WHERE c.oid = a.attrelid
           AND NOT a.attisdropped
           AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                              'pg_catalog.regprocedure'::pg_catalog.regtype,
                              'pg_catalog.regoper'::pg_catalog.regtype,
                              'pg_catalog.regoperator'::pg_catalog.regtype,
                              'pg_catalog.regconfig'::pg_catalog.regtype,
                              'pg_catalog.regdictionary'::pg_catalog.regtype)
           AND c.relnamespace = n.oid
           AND n.nspname NOT IN ('pg_catalog', 'information_schema');
     ```

4. 無効なデータベースがないかチェックします。

   - 無効なデータベースがないことを確認します。pg_database カタログの datconnlimit 列には、DROP DATABASE オペレーション中に中断されたデータベースを無効としてマークする -2 という値が含まれています。

     次のクエリを使用して、無効なデータベースをチェックします。

     ```
     SELECT datname FROM pg_database WHERE datconnlimit = - 2;
     ```

   - 前のクエリは無効なデータベース名を返します。DROP DATABASE *invalid_db_name*; を使用して無効なデータベースを削除できます。また、次のコマンドを使用して、無効なデータベースを削除することもできます。

     ```
     SELECT 'DROP DATABASE ' || quote_ident(datname) || ';' FROM pg_database WHERE
      datconnlimit = -2 \gexec
     ```

   無効なデータベースの詳細については、「無効なデータベースでの自動バキュームの動作を理解する」を参照してください。

5. 論理レプリケーションスロットの処理 — データベースに論理レプリケーションスロットがある場合、アップグレードは実行できません。論理レプリケーションスロットは通常、データベースからデータレイク、BI ツール、およびその他のターゲットへのテーブルのレプリケートおよび AWS DMS に使用されます。アップグレードする前に、使用中の論理レプリケーションスロットの目的を確認し、削除しても問題ないことを確認してください。論理レプリケーションスロットがまだ使用されている場合は、それらを削除しないでください。また、その場合、アップグレードを続行することはできません。

   論理レプリケーションスロットが不要な場合は、次の SQL を使用して削除できます。

   ```
   SELECT * FROM pg_replication_slots WHERE slot_type NOT LIKE 'physical';
   SELECT pg_drop_replication_slot(slot_name);
   ```

   pglogical 拡張機能を使用する論理レプリケーション設定でも、メジャーバージョンアップグレードを正常に行うには、スロットを削除する必要があります。pglogical 拡張機能を使用して作成されたスロットを識別して削除する方法については、「RDS for PostgreSQL 用ロジカルレプリケーションスロットの管理」を参照してください。

ソースバージョン 17 以降では、non-read-replicas の論理レプリケーションスロットはアップグレードを通じて保持されます。リードレプリカで作成された論理レプリケーションスロットは、アップグレード中は保持されません。

アップグレードを開始する前に、すべてのトランザクションと論理デコードメッセージがスロットから消費されていることを確認します。論理レプリケーションスロットによって保持されている消費されていない先行書き込みログファイル (WAL) がある場合、アップグレードは失敗し、問題のスロットを識別するメッセージが表示されます。詳細については、「PostgreSQL ドキュメント」を参照してください。

ソースバージョンが 17.8 または 18.2 より前のマルチ AZ クラスターでは、`flow_control` が無効になっていることを確認します。詳細については、「マルチ AZ DB クラスターのフロー制御のオンとオフ」を参照してください。フロー制御を無効にするには、`shared_preload_libraries` から拡張機能を削除するか、DB インスタンスを再起動します。

6. リードレプリカの処理 - シングル AZ DB インスタンスまたはマルチ AZ DB インスタンス配置のアップグレードでは、プライマリ DB インスタンスと共にリージョン内リードレプリカもアップグレードされます。Amazon RDS は、マルチ AZ DB クラスターのリードレプリカをアップグレードしません。

   リードレプリカを個別にアップグレードすることはできません。個別にアップグレードできるとすると、プライマリインスタンスとレプリカデータベースの PostgreSQL メジャーバージョンが一致しないという状況が生じる可能性があります。ただし、リードレプリカをアップグレードすると、プライマリ DB インスタンスのダウンタイムが増加する場合があります。リードレプリカのアップグレードを防ぐには、レプリカをスタンドアロンインスタンスに昇格させるか、アップグレードプロセスをスタートする前にレプリカを削除します。

   アップグレードプロセスでは、リードレプリカの現在のパラメータグループに基づいて、リードレプリカのパラメータグループが再作成されます。アップグレードの完了後にのみ、リードレプリカを変更してカスタムパラメータグループをリードレプリカに適用できます。リードレプリカの詳細については、「Amazon RDS for PostgreSQL でのリードレプリカの使用」を参照してください。

7. ラージオブジェクトの処理 – PostgreSQL では、通常の列データ型で許可される最大サイズを超える大きなバイナリオブジェクト (ファイル、画像、動画など) を保存および管理するために、ラージオブジェクト (BLOB とも呼ばれます) が使用されます。詳細については、PostgreSQL ラージオブジェクトのドキュメントを参照してください。

数百万のラージオブジェクトがあり、アップグレード中にインスタンスがそれらを処理できない場合、アップグレードはメモリ不足になり、失敗する可能性があります。PostgreSQL メジャーバージョンのアップグレードプロセスは、大きく分けて 2 つのフェーズで構成されます。pg_dump によるスキーマダンプと、pg_restore による復元です。データベースに数百万のラージオブジェクトがある場合は、アップグレード中に pg_dump と pg_restore を処理するために十分なメモリがインスタンスにあることを確認し、より大きなインスタンスタイプにスケールする必要があります。

アップグレードを開始する前に、データベースにラージオブジェクトがあるかどうかを確認します。カタログには、ラージオブジェクトに関連付けられたメタデータ `pg_largeobject_metadata` が格納されます。実際のラージオブジェクトデータは `pg_largeobject` に保存されます。次のクエリを使用して、ラージオブジェクトの数を確認します。

```
SELECT count(*) FROM pg_largeobject_metadata;
```

既存のラージオブジェクトまたは孤立したラージオブジェクトをクリーンアップするには、「lo モジュールを使用したラージオブジェクトの管理」を参照してください。

メジャーバージョンのアップグレードを計画する際、データベースに 2,500 ~ 3,000 万個のラージオブジェクトが含まれている場合は、少なくとも 32 GB のメモリを備えたインスタンスタイプを使用することをお勧めします。この推奨事項は当社のテストに基づいており、特定のワークロードとデータベース設定によって異なる場合があります。データベースに追加のオブジェクト (テーブル、インデックス、マテリアライズドビューなど) が含まれている場合は、アップグレードプロセス中に最適なパフォーマンスを確保するために、より大きなインスタンスタイプを選択することをお勧めします。

8. ゼロ ETL 統合の処理 – 既存のゼロ ETL 統合がある場合は、これを削除してから、メジャーバージョンアップグレードを実行してください。次に、アップグレードが完了したら、統合を再作成します。

   ソースのメジャーバージョン 17 以降では、ゼロ ETL 統合はアップグレードを通じて保持できます。

9. バックアップの実行 - データベースの復元ポイントを認識できるように、メジャーバージョンアップグレードを実行する前にはバックアップを実行しておくことをお勧めします。バックアップ保持期間を 0 より大きい値に設定すると、アップグレードプロセスにおいて、アップグレード前後にデータベースの DB スナップショットが作成されます。バックアップ保持期間を変更する

には、「Amazon RDS DB インスタンスを変更する」と「the section called “マルチ AZ DB クラスターの変更”」を参照してください。

手動でバックアップを実行するには、「the section called “シングル AZ DB インスタンスの DB スナップショットの作成”」と「the section called “マルチ AZ DB クラスターのスナップショットの作成”」を参照してください。

10. メジャーバージョンのアップグレード前に特定の エクステンション を更新する - アップグレードでメジャーバージョンをスキップする場合、メジャーバージョンのアップグレード前に特定の エクステンション を更新する必要があります。例えば、バージョン 9.5.x または 9.6.x からバージョン 11.x 以降へのアップグレードでは、メジャーバージョンがスキップされます。更新する拡張機能には、空間データを処理するための PostGIS および関連する拡張機能が含まれます。

    - `address_standardizer`
    - `address_standardizer_data_us`
    - `postgis_raster`
    - `postgis_tiger_geocoder`
    - `postgis_topology`

    `rdkit` バージョン 4.6.0 以前、および PostgreSQL バージョン 16 以前を使用している場合は、`rdkit` に互換性がないため、PostgreSQL バージョン 17 に直接アップグレードすることはできません。アップグレードオプションは次のとおりです。

    - PostgreSQL バージョン 13 以前を使用している場合は、まずバージョン 14.14 以降のバージョン 14、15.9 以降のバージョン 15、または 16.5 以降のバージョン 16 へのメジャーバージョンアップグレードを実行してから、PostgreSQL 17 にアップグレードする必要があります。
    - PostgreSQL バージョン 14、15、または 16 を使用している場合は、バージョン 14.14 以降のバージョン 14、15.9 以降のバージョン 15、または 16.5 以降のバージョン 16 へのマイナーバージョンアップグレードを実行してから、PostgreSQL 17 にアップグレードする必要があります。

    使用している拡張機能ごとに以下のコマンドを実行します。

    ```
    ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
    ```

    詳細については、「RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード」を参照してください。PostGIS のアップグレードの詳細については、「ステップ 6: PostGIS 拡張機能を更新する」を参照してください。

11. メジャーバージョンアップグレードの前に特定の拡張機能を削除する - ターゲットバージョンでサポートされていない拡張機能は削除する必要があります。削除しない場合、アップグレードは失敗します。

    `plrust` 拡張機能は RDS PostgreSQL 18 から削除されます。既知の問題のため、RDS PostgreSQL バージョン 18.1 および 18.2 では `postgis_topology` 拡張機能を使用できません [1]、[2]。これらの拡張機能は、アップグレードする前に削除する必要があります。

    メジャーバージョンをバージョン 11.x にスキップするアップグレードは、pgRouting 拡張機能の更新をサポートしていません。バージョン 9.4.x、9.5.x、または 9.6.x からバージョン 11.x へのアップグレードでは、メジャーバージョンがスキップされます。pgRouting エクステンションを削除し、アップグレード後に互換性のあるバージョンに再インストールできます。更新できるエクステンションのバージョンについては、サポートされている PostgreSQL 拡張機能バージョンをご覧ください。

    `tsearch2` および `chkpass` のエクステンションは、PostgreSQL バージョン 11 以降では現在サポートされていません。

    拡張機能がインストールされているかどうかは、次のクエリで確認できます。

    ```
    SELECT * FROM pg_extension WHERE extname in ('extension_name');
    ```

12. unknown データ型の削除 - ターゲットのバージョンに応じて、unknown データ型を削除します。

    PostgreSQL バージョン 10 では、unknown データ型のサポートは終了しています。バージョン 9.6 のデータベースで unknown データ型を使用している場合、バージョン 10 にアップグレードすると次のようなエラーメッセージが表示されます。

    ```
    Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
    The instance could not be upgraded because the 'unknown' data type is used in user
     tables.
    Please remove all usages of the 'unknown' data type and try again."
    ```

    データベース内の unknown データ型を検索して、問題の列を削除したり、サポートされているデータ型に変更したりするには、次の SQL を使用します。

    ```
    SELECT DISTINCT data_type FROM information_schema.columns WHERE data_type ILIKE
     'unknown';
    ```

13.リハーサル更新の実行 - プロダクションデータベースのアップグレードを行う前に、プロダクションデータベースの複製でメジャーバージョンアップグレードをテストすることを強くお勧めします。複製されたテストデータベースの実行計画を監視して、実行計画のリグレッションが発生していないかどうかを確認し、そのパフォーマンスを評価できます。テストインスタンスの複製を作成するには、データベースを最新スナップショットから復元するか、ポイントインタイムの復元を実行して復元可能な直近の時間でデータベースを復元します。

詳細については、「the section called “スナップショットからの復元”」または「the section called “ポイントインタイムリカバリ”」を参照してください。マルチ AZ DB クラスターについては、「the section called “スナップショットからマルチ AZ DB クラスターへの復元”」または「the section called “マルチ AZ DB クラスターを指定の時点の状態に復元する”」を参照してください。

アップグレードの実施の詳細については、「the section called “エンジンバージョンの手動アップグレード”」を参照してください。

バージョン 9.6 のデータベースをバージョン 10 にアップグレードする場合、PostgreSQL 10 では、デフォルトで並列クエリが有効になることに注意してください。テストデータベースの `max_parallel_workers_per_gather` パラメータを 2 に変更することで、アップグレードする前に並列処理による影響をテストできます。

> **Note**
>
> `default.postgresql10` DB パラメータグループで は、`max_parallel_workers_per_gather` パラメータのデフォルト値は 2 です。

詳細については、「PostgreSQL ドキュメント」の「Parallel Query」(並列クエリ) を参照してください。バージョン 10 で並列処理を無効にするには、`max_parallel_workers_per_gather` パラメータを 0 に設定します。

メジャーバージョンのアップグレード中、`public` データベースと `template1` データベース、およびすべてのデータベースの `public` スキーマは、一時的に名前が変更されます。これらのオブジェクトは、元の名前とランダム文字列を組み合わせた名前でログに表示されます。この文字列は、メジャーバージョンアップグレード時に `locale` や `owner` などのカスタム設定が保持されるように追加されます。アップグレードが完了したら、これらのオブジェクト名は元の名前に戻ります。

> **Note**
>
> メジャーアップグレードのプロセス中に、DB インスタンスまたはマルチ AZ DB クラスターのポイントインタイムの復元を実行することはできません。Amazon RDS でアップグレードが完了すると、データベースの自動バックアップが実施されます。ポイントインタイムの復元を実行できるのは、アップグレードのスタート前およびデータベースの自動バックアップ完了後です。

14. 事前チェック手順エラーでアップグレードが失敗した場合、問題を解決します - メジャーバージョンのアップグレードプロセス中に、Amazon RDS for PostgreSQL は初期に事前チェック手順を実行して、アップグレードの失敗の原因となる可能性のある問題を特定します。事前チェック手順は、インスタンス内のすべてのデータベースにわたって潜在的な互換性のない条件をすべてチェックします。

    事前チェックで問題が発生した場合、アップグレード事前チェックが失敗したことを示すログイベントが作成されます。事前確認プロセスの詳細は、データベースのすべてのデータベースの `pg_upgrade_precheck.log` と名前が付けられたアップグレードログにあります。Amazon RDS では、ファイル名にタイムスタンプが追加されます。ログの表示の詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

    リードレプリカのアップグレードが事前チェックで失敗した場合、失敗したリードレプリカのレプリケーションは中断され、リードレプリカは終了状態になります。リードレプリカを削除し、アップグレードしたプライマリ DB インスタンスに基づいて、新しいリードレプリカを再作成します。

    事前チェックログで特定されたすべての問題を解決してから、メジャーバージョンのアップグレードを再試行します。事前チェックログの例は次のようになります。

```
-----------------------------------------------------------------------
Upgrade could not be run on Wed Apr 4 18:30:52 2018
-----------------------------------------------------------------------
The instance could not be upgraded from 9.6.11 to 10.6 for the following reasons.
Please take appropriate action on databases that have usage incompatible with the
 requested major engine version upgrade and try the upgrade again.

* There are uncommitted prepared transactions. Please commit or rollback all prepared
 transactions.* One or more role names start with 'pg_'. Rename all role names that
 start with 'pg_'.
```

```
* The following issues in the database 'my"million$"db' need to be corrected before
 upgrading:** The ["line","reg*"] data types are used in user tables. Remove all
 usage of these data types.
** The database name contains characters that are not supported by RDS for
 PostgreSQL. Rename the database.
** The database has extensions installed that are not supported on the target
 database version. Drop the following extensions from your database: ["tsearch2"].

* The following issues in the database 'mydb' need to be corrected before
 upgrading:** The database has views or materialized views that depend on
 'pg_stat_activity'. Drop the views.
```

15.データベースのアップグレード中にリードレプリカのアップグレードが失敗した場合は、問題を解決します - 失敗したリードレプリカは `incompatible-restore` 状態になり、レプリケーションはデータベースで終了します。リードレプリカを削除し、アップグレードしたプライマリ DB インスタンスに基づいて、新しいリードレプリカを再作成します。

> **Note**
>
> Amazon RDS は、マルチ AZ DB クラスターのリードレプリカをアップグレードしません。マルチ AZ DB クラスターのメジャーバージョンアップグレードを実行すると、リードレプリカのレプリケーション状態が終了に変わります。

リードレプリカのアップグレードは、次の理由で失敗することがあります。

- 待機時間が経過してもプライマリ DB インスタンスにキャッチアップできなかった。
- ストレージ不足、互換性のない復元など、最終状態または互換性のないライフサイクル状態であった。
- プライマリ DB インスタンスのアップグレードのスタート時に、リードレプリカで別のマイナーバージョンのアップグレードが実行されていた。
- リードレプリカで互換性のないパラメータを使用していた。
- リードレプリカがプライマリ DB インスタンスと通信できず、データフォルダを同期できなかった。

16.本番稼働用データベースのアップグレード - リハーサルのメジャーバージョンアップグレードに成功すれば、自信を持って本番稼働用のプロダクションデータベースをアップグレードできます。詳細については、「エンジンバージョンの手動アップグレード」を参照してください。

17.ANALYZE 操作を実行して `pg_statistic` テーブルを更新します。これは、すべての PostgreSQL データベースのすべてのデータベースに対して行う必要があります。Optimizer の統

計情報はメジャーバージョンのアップグレード中には転送されないため、パフォーマンスの問題を回避するためにすべての統計情報を再生成する必要があります。次のようにパラメータを指定せずにコマンドを実行して、現在のデータベース内のすべての標準テーブルの統計情報を生成します。

```
ANALYZE VERBOSE;
```

VERBOSE フラグはオプションですが、使用することで進行状況を表示できます。詳細については、「PostgreSQL ドキュメント」の「ANALYZE」を参照してください。

ANALYZE VERBOSE を使用する代わりに特定のテーブルを分析する場合は、次のようにテーブルごとに ANALYZE コマンドを実行します。

```
ANALYZE table_name;
```

パーティションテーブルの場合は、常に親テーブルを分析します。このプロセスでは、以下の操作を行います。

- すべてのパーティションにわたって行を自動的にサンプリングする
- パーティションごとに統計を再帰的に更新する
- 重要な計画統計を親レベルで維持する

親テーブルには実際のデータは保存されませんが、クエリの最適化には親テーブルの分析が不可欠です。個々のパーティションに対してのみ ANALYZE を実行すると、効率的なパーティション間計画に必要な包括的な統計をオプティマイザで利用できないため、クエリパフォーマンスが低下する可能性があります。

> **Note**
>
> パフォーマンスの問題を回避するため、アップグレード後にシステムで ANALYZE を実行してください。

メジャーバージョンのアップグレードが完了したら、次のことをお勧めします。

- PostgreSQL アップグレードでは、PostgreSQL エクステンションはアップグレードされません。エクステンションをアップグレードするには、「RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード」を参照してください。

- オプションで、Amazon RDS を使用して、pg_upgrade ユーティリティによって作成される 2 つのログを表示できます。表示できるのは pg_upgrade_internal.log および pg_upgrade_server.log です。Amazon RDS では、これらのログのファイル名にタイムスタンプが追加されます。これらのログも、他のログと同様、表示できます。詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

  Amazon CloudWatch Logs にアップグレードログをアップロードすることもできます。詳細については、「Amazon CloudWatch Logs への PostgreSQL ログの発行」を参照してください。
- すべてが期待どおりに機能することを確認するには、同様のワークロードでアップグレードされたデータベースでアプリケーションをテストします。アップグレードが確認されたら、このテストインスタンスを削除できます。

RDS for PostgreSQL のマイナーバージョンの自動アップグレード

DB インスタンスまたはマルチ AZ DB クラスターの作成時または変更時に、[マイナーバージョン自動アップグレード] を有効にした場合、データベースを自動的にアップグレードできます。

Amazon RDS は、複数のデータベースリソースと AWS アカウントにわたるマイナーバージョンの自動アップグレードを管理するためのアップグレードロールアウトポリシーもサポートしています。詳細については、「自動マイナーバージョンアップグレードの AWS Organizations アップグレードロールアウトポリシーの使用」を参照してください。

RDS for PostgreSQL の各メジャーバージョンでは、RDS によって 1 つのマイナーバージョンが自動アップグレードバージョンとして指定されます。Amazon RDS でマイナーバージョンのテストと承認が完了すると、メンテナンスウィンドウの間にマイナーバージョンアップグレードが自動的に行われます。RDS では、新しくリリースされたマイナーバージョンが自動アップグレードバージョンとして自動的に設定されることはありません。RDS によって新しい自動アップグレードバージョンが指定される前に、以下のような複数の基準が考慮されます。

- 既知のセキュリティの問題
- PostgreSQL コミュニティバージョンのバグ
- マイナーバージョンがリリースされてからのフリート全体の安定性

次の AWS CLI コマンドを使用して、特定の AWS リージョン で指定された PostgreSQL マイナーバージョンの現在の自動マイナーアップグレードターゲットバージョンを確認できます。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine postgres \
--engine-version minor-version \
--region region \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output text
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine postgres ^
--engine-version minor-version ^
--region region ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output text
```

例えば、次の AWS CLI コマンドは、米国東部 (オハイオ) AWS リージョン (us-east-2) の PostgreSQL マイナーバージョン 16.1 の自動マイナーアップグレードターゲットを決定します。

Linux、macOS、Unix の場合:

```
aws rds describe-db-engine-versions \
--engine postgres \
--engine-version 16.1 \
--region us-east-2 \
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" \
--output table
```

Windows の場合:

```
aws rds describe-db-engine-versions ^
--engine postgres ^
--engine-version 16.1 ^
--region us-east-2 ^
--query "DBEngineVersions[*].ValidUpgradeTarget[*].
{AutoUpgrade:AutoUpgrade,EngineVersion:EngineVersion}" ^
--output table
```

以下のような出力が生成されます。

```
---------------------------------
|    DescribeDBEngineVersions    |
+--------------+-----------------+
|  AutoUpgrade |  EngineVersion  |
+--------------+-----------------+
|  False       |  16.2           |
|  True        |  16.3           |
|  False       |  16.4           |
|  False       |  16.5           |
|  False       |  16.6           |
|  False       |  17.1           |
|  False       |  17.2           |
+--------------+-----------------+
```

この例では、AutoUpgrade 値は、True for PostgreSQL バージョン 16.3 です。したがって、自動マイナーアップグレードターゲットは PostgreSQL バージョン 16.3 であり、出力で強調表示されています。

PostgreSQL データベースは、以下の基準を満たしている場合、メンテナンスウィンドウの間に自動的にアップグレードされます。

- データベースは、[マイナーバージョン自動アップグレード] オプションを有効にしています。
- データベースでは、現在の自動アップグレードマイナーバージョン未満の DB エンジンのマイナーバージョンが実行されています。

詳細については、「マイナーエンジンバージョンの自動アップグレード」を参照してください。

> **Note**
>
> PostgreSQL のアップグレードでは、PostgreSQL のエクステンションはアップグレードされません。エクステンションをアップグレードするには、「RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード」を参照してください。

RDS for PostgreSQL データベースでの PostgreSQL 拡張機能のアップグレード

PostgreSQL エンジンのアップグレードでは、PostgreSQL のほとんどのエクステンションはアップグレードされません。バージョンアップグレードの後にエクステンションを更新するには、ALTER EXTENSION UPDATE のコマンドを使用します。

> **Note**
>
> PostGIS 拡張機能の更新については、「PostGIS 拡張機能を使用した空間データの管理 (ステップ 6: PostGIS 拡張機能を更新する)」を参照してください。
> pg_repack 拡張機能を更新する場合、拡張機能をドロップしてアップグレードされたデータベースに新しいバージョンを作成します。詳細については、「pg_repack ドキュメント」の「pg_repack installation」(pg_repack のインストール) を参照してください。

エクステンションをアップグレードするには、次のコマンドを使用します。

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

PostgreSQL エクステンションのサポートされているバージョンのリストについては、「サポートされている PostgreSQL 拡張機能バージョン」を参照してください。

現在インストールされているエクステンションを一覧表示するには、次のコマンドで PostgreSQL の pg_extension カタログを使用します。

```
SELECT * FROM pg_extension;
```

インストールで使用可能な特定の拡張機能バージョンのリストを表示するには、次のコマンドで PostgreSQL の pg_available_extension_versions ビューを使用します。

```
SELECT * FROM pg_available_extension_versions;
```

イベントによる RDS for PostgreSQL エンジンのアップグレードのモニタリング

RDS for PostgreSQL データベースのエンジンバージョンをアップグレードすると、Amazon RDS はプロセスの各フェーズで特定のイベントを発行します。アップグレードの進行状況を追跡するには、これらのイベントを表示またはサブスクライブします。

RDS イベントの詳細については、「Amazon RDS イベントのモニタリング」を参照してください。

エンジンのアップグレード中に発生する特定の Amazon RDS イベントの詳細については、「Amazon RDS イベントカテゴリとイベントメッセージ」を参照してください。

PostgreSQL DB スナップショットエンジンのバージョンのアップグレード

Amazon RDS を使用すると、PostgreSQL DB インスタンスのストレージボリュームの DB スナップショットを作成できます。作成した DB スナップショットは Amazon RDS インスタンスが使用するエンジンバージョンに基づいています。DB スナップショットのエンジンバージョンをアップグレードできます。

新しいエンジンバージョンにアップグレードした DB スナップショットをリストアしたら、アップグレードに問題がないか必ずテストしてください。主なバージョンアップグレードの詳細は、RDS for PostgreSQL DB エンジンのアップグレード を参照してください。DB スナップショットをリストアする方法については DB インスタンスへの復元 をご覧ください。

暗号化されている場合またはされていない場合でも、マニュアル DB スナップショットをアップグレードすることができます。

RDS for PostgreSQL DB スナップショットで使用できるエンジンバージョンを表示するには、次の AWS CLI 例を使用します。

```
aws rds describe-db-engine-versions --engine postgres  --engine-version example-
engine-version --query "DBEngineVersions[*].ValidUpgradeTarget[*].
{EngineVersion:EngineVersion}" --output text --include-all
```

RDS for PostgreSQL DB スナップショットで使用できるエンジンバージョンの詳細については、「RDS for PostgreSQL のメジャーバージョンアップグレードの選択」を参照してください。

> **Note**
>
> 自動バックアップ処理中に作成された自動 DB スナップショットをアップグレードすることはできません。

コンソール

DB スナップショットをアップグレードするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、「Snapshots」 を選択します。
3. アップグレードしたいスナップショットを選択します。
4. [アクション] は、[スナップショットのアップグレード] を選択します。[スナップショットのアップグレード] ページが表示されます。
5. アップグレードする [新しいエンジンバージョン] を選択します。
6. [変更を保存] を選択してスナップショットをアップグレードします。

   アップグレード中は、この DB スナップショットに関するスナップショット操作が、すべて無効になります。また、DB スナップショットのステータスは利用可能からアップグレード中に変わり、プロセスが完了すると有効になります。スナップショットの破損問題で DB スナップショットをアップグレードできない場合、ステータスは使用不可になります。この状態からスナップショットを復元することはできません。

> **Note**
>
> DB スナップショットアップグレードが失敗した場合、スナップショットは元の状態にロールバックします。

AWS CLI

DB スナップショットを新しいバージョンのデータベースエンジンにアップグレードするには、AWS CLI の modify-db-snapshot コマンドを使用します。

パラメータ

- `--db-snapshot-identifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。
- `--engine-version` – DB スナップショットをこのエンジンバージョンにアップグレードする。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier my_db_snapshot \
```

```
    --engine-version new_version
```

Windows の場合:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier my_db_snapshot ^
    --engine-version new_version
```

RDS API

DB スナップショットを新しいバージョンのデータベースエンジンにアップグレードするには、Amazon RDS API の ModifyDBSnapshot オペレーションを呼び出します。

- `DBSnapshotIdentifier` – アップグレードする DB スナップショットの識別子です。識別子は独自の Amazon リソースネーム (ARN) にしてください。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。
- `EngineVersion` – DB スナップショットをこのエンジンバージョンにアップグレードする。

Amazon RDS for PostgreSQL でのリードレプリカの使用

リードレプリカをインスタンスに追加することによって、Amazon RDS for PostgreSQL DB インスタンスの読み取りをスケーリングできます。他の Amazon RDS データベースエンジンと同様、RDS for PostgreSQL は PostgreSQL のネイティブレプリケーションメカニズムを使用して、ソース DB の更新がリードレプリカに反映されるようにします。リードレプリカと Amazon RDS の概要については、「DB インスタンスのリードレプリカの操作」を参照してください。

RDS for PostgreSQL でのリードレプリカの使用に関する特定の情報については、以下を参照してください。

PostgreSQL でのリードレプリカの制限

PostgreSQL リードレプリカの制限は以下のとおりです。

- PostgreSQL リードレプリカは読み取り専用です。リードレプリカは書き込み可能な DB インスタンスではありませんが、リードレプリカをスタンドアロンの RDS for PostgreSQL DB インスタンスに昇格させることができます。ただし、このプロセスを元に戻すことはできません。
- RDS for PostgreSQL DB インスタンスで、14.1 より前のバージョンの PostgreSQL が実行されている場合、別のリードレプリカからリードレプリカを作成することはできません。RDS for PostgreSQL でカスケードリードレプリカがサポートされているのは、RDS for PostgreSQL バージョン 14.1 以降のリリースのみです。詳細については、「RDS for PostgreSQL でのカスケードリードレプリカの使用」を参照してください。
- PostgreSQL リードレプリカを昇格させると、書き込み可能な DB インスタンスになります。これにより、ソース DB インスタンスからの先書き口グ (WAL) ファイルの受信が停止し、読み取り専用インスタンスではなくなります。RDS for PostgreSQL DB インスタンスと同様に、昇格した DB インスタンスから新しいリードレプリカを作成できます。詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。
- レプリケーションチェーン内 (一連のカスケードリードレプリカ) から PostgreSQL リードレプリカを昇格させると、既存のダウンストリームリードレプリカは、昇格したインスタンスから WAL ファイルを自動的に受信し続けます。詳細については、「RDS for PostgreSQL でのカスケードリードレプリカの使用」を参照してください。
- ソース DB インスタンスでユーザートランザクションが実行されていない場合、関連付けられた PostgreSQL リードレプリカは、最長 5 分のレプリケーションの遅延を報告します。レプリカラグは `currentTime - lastCommitedTransactionTimestamp` として計算され、トランザクションが処理されていない場合、先書き口グ (WAL) セグメントが切り替わるまでの一定期間、レ

プリカラグの値が増加することを意味します。デフォルトでは、RDS for PostgreSQL は 5 分ごとに WAL セグメントを切り替えます。これにより、トランザクションレコードが生成され、報告されるラグが減少します。

- RDS for PostgreSQL 14.1 より前のバージョンの PostgreSQL リードレプリカでは、自動バックアップをオンにすることはできません。リードレプリカの自動バックアップは、RDS for PostgreSQL 14.1 以降のバージョンでのみサポートされます。RDS for PostgreSQL 13 以前のバージョンでバックアップが必要な場合は、リードレプリカのスナップショットを作成します。
- リードレプリカでは、ポイントインタイムリカバリ (PITR) はサポートされません。PITR は、リードレプリカではなく、プライマリ (ライター) インスタンスでのみ使用できます。詳細についてはAmazon RDS の DB インスタンスを特定の時点に復元するを参照してください。
- PostgreSQL バージョン 12 以前のリードレプリカは、60～90 日間のメンテナンスウィンドウ中に自動的に再起動し、パスワードのローテーションを適用します。スケジュールされた再起動前にレプリカからソースへの接続が切断された場合も、再起動してレプリケーションを再開します。PostgreSQL バージョン 13 以降では、パスワードのローテーションプロセス中にリードレプリカでレプリケーションの切断と再接続が短時間発生する場合があります。

PostgreSQL でのリードレプリカの設定

RDS for PostgreSQL では、PostgreSQL ネイティブストリーミングレプリケーションを使用して、ソース DB インスタンスの読み取り専用コピーを作成します。このリードレプリカ DB インスタンスは、非同期的に作成されたソース DB インスタンスの物理レプリカです。これは、先書き口グ (WAL) のデータをソース DB インスタンスからリードレプリカに送信する特別な接続によって作成されます。詳細については、PostgreSQL ドキュメントの「Streaming Replication」(ストリーミングレプリケーション) を参照してください。

ソース DB インスタンスでデータベースの変更が行われた場合、PostgreSQL は、非同期的に安全な接続に変更をストリーミングします。クライアントアプリケーションからソース DB インスタンスまたはリードレプリカへの通信を暗号化するには、`ssl` パラメータを `1` に設定します。詳細については、「PostgreSQL DB インスタンスで SSL を使用する」を参照してください。

PostgreSQL はレプリケーションロールを使用して、ストリーミングレプリケーションを実行します。このロールには特権がありますが、データの変更には使用できません。PostgreSQL ではレプリケーション処理に 1 つのプロセスを使用します。

ソース DB インスタンスのオペレーションとユーザーに影響を与えることなく、PostgreSQL リードレプリカを作成できます。Amazon RDS では、サービスに影響を与えることなく、ソース DB インスタンスとリードレプリカに必要なパラメータとアクセス許可を設定できます。ソース DB インスタ

ンスのスナップショットが取得され、このスナップショットを使用してリードレプリカが作成されます。将来のある時点でリードレプリカを削除しても、停止は発生しません。

同一リージョン内の 1 つの ソース DB インスタンスから、最大 15 個のリードレプリカを作成できます。さらに、RDS for PostgreSQL 14.1 以降では、ソース DB インスタンスから最大 3 層のリードレプリカのチェーン (カスケード) を作成することもできます。詳細については、「RDS for PostgreSQL でのカスケードリードレプリカの使用」を参照してください。いずれの場合も、ソース DB インスタンスで自動バックアップを設定する必要があります。これを行うには、DB インスタンスのバックアップ保持期間を 0 以外の値に設定します。詳細については、「リードレプリカの作成」を参照してください。

RDS for PostgreSQL DB インスタンス用のリードレプリカをソース DB インスタンスとして同じ AWS リージョン 内に作成できます。これは、リージョン内レプリケーションと呼ばれます。また、ソース DB インスタンスと異なる AWS リージョン にリードレプリカを作成することもできます。これは、クロスリージョンレプリケーションと呼ばれます。クロスリージョンリードレプリカの設定に関する詳細は、「別の でのリードレプリカの作成AWS リージョン」を参照してください。「RDS for PostgreSQL のバージョンが異なる場合のストリーミングレプリケーションの仕組み」で説明されているように、リージョン内およびクロスリージョンのレプリケーションプロセスをサポートするさまざまなメカニズムは、RDS for PostgreSQL のバージョンによって若干異なります。

レプリケーションを効率的に実行するには、各リードレプリカにソース DB インスタンスと同程度のコンピューティングリソースとストレージリソースが必要です。ソース DB インスタンスをスケールした場合は、リードレプリカもスケールする必要があります。

互換性のないパラメータがあり、それが原因でリードレプリカを起動できない場合、Amazon RDS によってリードレプリカのパラメータ値が上書きされます。例えば、リードレプリカよりもソース DB インスタンスの方が `max_connections` パラメータ値が高いとします。この場合は、Amazon RDS によって、リードレプリカのパラメータがソース DB インスタンスのパラメータと同じ値になるように更新されます。

RDS for PostgreSQL リードレプリカは、ソース DB インスタンスの外部データラッパー (FDW) を介して利用可能な外部データベースにアクセスできます。例えば、RDS for PostgreSQL DB インスタンスで、`mysql_fdw` ラッパーを使用して RDS for MySQL のデータにアクセスするとします。その場合、リードレプリカはそのデータにもアクセスできます。サポートされているその他の FDW には、`oracle_fdw`、`postgres_fdw`、および `tds_fdw` があります。詳細については、「Amazon RDS for PostgreSQL でサポートされている外部データラッパーを使用する」を参照してください。

マルチ AZ 構成で RDS for PostgreSQL リードレプリカを使用する

リードレプリカは、シングル AZ DB インスタンスから、またはマルチ AZ DB インスタンスから作成できます。重要なデータの耐久性と可用性を高めるために、1 つのスタンバイレプリカを備えたマルチ AZ 配置を使用できます。スタンバイレプリカは、ソース DB がフェイルオーバーした場合にワークロードを引き受けることができる専用のリードレプリカです。スタンバイレプリカを使用して読み取りトラフィックを処理することはできません。ただし、トラフィックの多いマルチ AZ DB インスタンスのリードレプリカを作成して、読み取り専用クエリをオフロードできます。マルチ AZ 配置についての詳細は、「Amazon RDS のマルチ AZ DB インスタンスデプロイ」を参照してください。

マルチ AZ 配置のソース DB インスタンスがスタンバイにフェイルオーバーすると、関連付けられているリードレプリカがスタンバイ (現在はプライマリ) をレプリケーションのソースとして使用するよう切り替わります。以下のように、RDS for PostgreSQL のバージョンによっては、リードレプリカの再起動が必要になる場合があります。

- PostgreSQL 13 以降のバージョン – 再起動は必要ありません。リードレプリカは、新しいプライマリと自動的に同期されます。ただし、場合によっては、クライアントアプリケーションがリードレプリカのドメインネームサービス (DNS) の詳細をキャッシュすることがあります。その場合は、有効期限 (TTL) 値を 30 秒未満に設定します。これにより、リードレプリカは古い IP アドレスを保持できなくなります (したがって、新しいプライマリと同期されません)。このベストプラクティスおよびその他のベストプラクティスの詳細については、「Amazon RDS の基本的な操作のガイドライン」を参照してください。
- PostgreSQL 12 およびそれ以前のすべてのバージョン – スタンバイレプリカにフェイルオーバーした後にリードレプリカが自動的に再起動します。スタンバイ (現在のプライマリ) の IP アドレスとインスタンス名が異なるためです。再起動すると、リードレプリカが新しいプライマリと同期されます。

フェイルオーバーの詳細については、「Amazon RDS 用のマルチ AZ DB インスタンスのフェイルオーバー」を参照してください。リードレプリカがマルチ AZ 配置でどのように機能するかについての詳細は、「DB インスタンスのリードレプリカの操作」を参照してください。

リードレプリカのフェイルオーバーをサポートするため、リードレプリカをマルチ AZ DB インスタンスとして作成できます。これにより、Amazon RDS は、別のアベイラビリティーゾーン (AZ) にレプリカのスタンバイを作成できます。リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。

リードレプリカの論理デコード

RDS for PostgreSQL は、PostgreSQL 16.1 を使用したスタンバイからの論理レプリケーションをサポートしています。これにより、読み取り専用スタンバイから論理デコードを作成し、プライマリ DB インスタンスの負荷を軽減できます。複数のシステム間でデータを同期する必要があるアプリケーションで可用性を高めることができます。この機能により、データウェアハウスとデータ分析のパフォーマンスが向上します。

また、特定のスタンバイのレプリケーションスロットは、そのスタンバイのプライマリへの昇格を保持します。つまり、プライマリ DB インスタンスのフェイルオーバーやスタンバイから新しいプライマリへの昇格の際にも、レプリケーションスロットは保持され、以前のスタンバイサブスクライバーには影響しません。

リードレプリカに論理デコードを作成するには

1. 論理レプリケーションを有効にする – スタンバイで論理デコードを作成するには、ソース DB インスタンスとその物理レプリカで論理レプリケーションを有効にする必要があります。詳細については、「PostgreSQL でのリードレプリカの設定」を参照してください。

   - 新しく作成された RDS for PostgreSQL DB インスタンスの論理レプリケーションを有効にするには – 新しい DB カスタムパラメータグループを作成し、静的パラメータ `rds.logical_replication` を 1 に設定します。次に、この DB パラメータグループをソース DB インスタンスとその物理リードレプリカに関連付けます。詳細については、「Amazon RDS で DB パラメータグループを DB インスタンスに関連付ける」を参照してください。
   - 既存の RDS for PostgreSQL DB インスタンスの論理レプリケーションを有効にするには — ソース DB インスタンスとその物理リードレプリカの DB カスタムパラメータグループを変更して、静的パラメータ `rds.logical_replication` を 1 に設定します。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

   > **Note**
   >
   > これらのパラメータの変更を適用するには、DB インスタンスを再起動する必要があります。

   次のクエリを使用して、ソース DB インスタンスとその物理リードレプリカの `wal_level` および `rds.logical_replication` の値を確認できます。

```
Postgres=>SELECT name,setting FROM pg_settings WHERE name IN
 ('wal_level','rds.logical_replication');

 name                      | setting
---------------------------+---------
 rds.logical_replication | on
 wal_level               | logical
(2 rows)
```

2. ソースデータベースにテーブルを作成する – ソース DB インスタンスのデータベースに接続します。詳細については、「PostgreSQL データベースエンジンを実行する DB インスタンスへの接続」を参照してください。

   次のクエリを使用して、ソースデータベースにテーブルを作成し、値を挿入します。

```
Postgres=>CREATE TABLE LR_test (a int PRIMARY KEY);
CREATE TABLE
```

```
Postgres=>INSERT INTO LR_test VALUES (generate_series(1,10000));
INSERT 0 10000
```

3. ソーステーブルのパブリケーションを作成する – 次のクエリを使用して、ソース DB インスタンスにテーブルのパブリケーションを作成します。

```
Postgres=>CREATE PUBLICATION testpub FOR TABLE LR_test;
CREATE PUBLICATION
```

   SELECT クエリを使用して、ソース DB インスタンスと物理リードレプリカインスタンスの両方で作成されたパブリケーションの詳細を確認します。

```
Postgres=>SELECT * from pg_publication;

oid    | pubname | pubowner | puballtables | pubinsert | pubupdate | pubdelete |
 pubtruncate | pubviaroot
-------+---------+----------+--------------+-----------+-----------+-----------
+-------------+------------
 16429 | testpub |    16413 | f            | t         | t         | t         | t
          | f
```

```
(1 row)
```

4. 論理レプリカインスタンスからサブスクリプションを作成する – 論理レプリカインスタンスとして別の RDS for PostgreSQL DB インスタンスを作成します。この論理レプリカインスタンスが物理リードレプリカインスタンスにアクセスできるように、VPC が正しく設定されていることを確認します。詳細については、「Amazon VPC と Amazon RDS」を参照してください。ソース DB インスタンスがアイドル状態の場合、接続の問題が発生し、プライマリがデータをスタンバイに送信しない可能性があります。

```
Postgres=>CREATE SUBSCRIPTION testsub CONNECTION 'host=Physical replica host name
 port=port
                dbname=source_db_name user=user password=password'
 PUBLICATION testpub;
NOTICE:  created replication slot "testsub" on publisher
CREATE SUBSCRIPTION
```

```
Postgres=>CREATE TABLE LR_test (a int PRIMARY KEY);
CREATE TABLE
```

SELECT クエリを使用して、論理レプリカインスタンスのサブスクリプションの詳細を確認します。

```
Postgres=>SELECT oid,subname,subenabled,subslotname,subpublications FROM
 pg_subscription;

oid    | subname | subenabled | subslotname | subpublications
-------+---------+------------+-------------+-----------------
 16429 | testsub | t          | testsub     | {testpub}
(1 row)
postgres=> select count(*) from LR_test;
 count
-------
 10000
(1 row)
```

5. 論理レプリケーションスロットの状態を確認する – ソース DB インスタンスの物理レプリケーションスロットのみを表示できます。

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;
```

```
slot_name                                      | slot_type | confirmed_flush_lsn
-----------------------------------------------+-----------+--------------------
 rds_us_west_2_db_dhqfsmo5wbbjqrn3m6b6ivdhu4 | physical  |
(1 row)
```

ただし、リードレプリカインスタンスでは、アプリケーションが論理変更をアクティブに消費すると、論理レプリケーションスロットと confirmed_flush_lsn 値の変化を確認できます。

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;

slot_name | slot_type | confirmed_flush_lsn
-----------+-----------+--------------------
 testsub   | logical   | 0/500002F0
(1 row)
```

```
Postgres=>select slot_name, slot_type, confirmed_flush_lsn from
 pg_replication_slots;

slot_name | slot_type | confirmed_flush_lsn
-----------+-----------+--------------------
 testsub   | logical   | 0/5413F5C0
(1 row)
```

RDS for PostgreSQL でのカスケードリードレプリカの使用

バージョン 14.1 以降の RDS for PostgreSQL では、カスケードリードレプリカがサポートされています。カスケードリードレプリカにより、ソースの RDS for PostgreSQL DB インスタンスにオーバーヘッドを追加せずに読み取りをスケーリングできます。ソース DB インスタンスでは、WAL ログへの更新が各リードレプリカに送信されません。代わりに、カスケード層内の各リードレプリカは、その層内の次のリードレプリカに WAL ログの更新を送信します。これにより、ソース DB インスタンスへの負荷が軽減されます。

カスケードリードレプリカを使用すると、RDS for PostgreSQL DB インスタンスは、チェーン内の最初のリードレプリカに WAL データを送信します。その後、そのリードレプリカは、チェーン内の 2 番目のレプリカに WAL データを送信し、その動作が順に続いていきます。その結果、チェーン内のすべてのリードレプリカに RDS for PostgreSQL DB インスタンスの更新が送信されますが、ソース DB インスタンスでのオーバーヘッドは発生しません。

ソースの RDS for PostgreSQL DB インスタンスから、チェーン内にリードレプリカを 3 層まで作成することができます。例えば、RDS for PostgreSQL 14.1 DB インスタンス、rpg-db-main があるとします。以下の操作を行うことができます。

- rpg-db-main で開始し、チェーン内に最初のリードレプリカ、read-replica-1 を作成します。
- 次に、read-replica-1 で、チェーン内に次のリードレプリカ、read-replica-2 を作成します。
- 最後に、read-replica-2 で、チェーン内に 3 番目のリードレプリカ、read-replica-3 を作成します。

rpg-db-main の層では、この 3 番目のカスケードリードレプリカに続く、別のリードレプリカを作成することはできません。一連の完全なインスタンス (RDS for PostgreSQL のソース DB インスタンスから、この層の最後のカスケードリードレプリカまで) は、最大 4 つの DB インスタンスで構成できます。

カスケードリードレプリカを設定するには、RDS for PostgreSQL で自動バックアップを有効にします。まずリードレプリカを作成し、RDS for PostgreSQL DB インスタンスで自動バックアップを有効にします。このプロセスは、他の Amazon RDS DB エンジンと同じです。詳細については、「リードレプリカの作成」を参照してください。

他のリードレプリカと同様に、カスケードの一部となっているリードレプリカを昇格できます。リードレプリカのチェーン内でリードレプリカを昇格させると、そのレプリカはチェーンから削除されます。例えば、rpg-db-main DB インスタンスのワークロードの一部を新しいインスタンスに移動するとします。この新しいインスタンスは経理部でのみ使用します。この例では、3 つのリードレプリカから成るチェーンがある仮定し、read-replica-2 を昇格させることにします。チェーンは以下のような影響を受けます。

- 昇格する read-replica-2 は、レプリケーションチェーンから削除されます。
  - このリードレプリカは、完全な読み取り/書き込み DB インスタンスになります。
  - 昇格前と同じように、read-replica-3 へのレプリケーションを継続します。
- rpg-db-main は、read-replica-1 へのレプリケーションを継続します。

リードレプリカの昇格についての詳細は、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。

Note

- RDS for PostgreSQL は、カスケードレプリカのメジャーバージョンアップグレードをサポートしていません。メジャーバージョンアップグレードを実行する前に、カスケードレプリカを削除する必要があります。ソース DB インスタンスと第 1 レベルのレプリカのアップグレードが完了したら、再作成できます。
- カスケードリードレプリカの場合、RDS for PostgreSQL は、レプリケーションの第 1 レベルではソース DB インスタンスごとに 15 個のリードレプリカを、レプリケーションの第 2 レベルと 第 3 レベルではソース DB インスタンスごとに 5 個のリードレプリカをサポートします。

RDS for PostgreSQL を使用したクロスリージョンカスケーディングリードレプリカの作成

RDS for PostgreSQL は、クロスリージョンカスケーディングリードレプリカをサポートします。ソース DB インスタンスからクロスリージョンレプリカを作成し、そこから同じリージョンレプリカを作成できます。ソース DB インスタンスから同じリージョンレプリカを作成し、そこからクロスリージョンレプリカを作成することもできます。

クロスリージョンレプリカを作成してから、同じリージョンレプリカを作成する

バージョン 14.1 以降 (`rpg-db-main`) の RDS for PostgreSQL DB インスタンス を使用して、以下を実行できます。

1. `rpg-db-main` (US-EAST-1) で開始し、チェーンに最初のクロスリージョンリードレプリカ `read-replica-1` (US-WEST-2) を作成します。
2. 最初のクロスリージョン `read-replica-1` (US-WEST-2) を使用して、チェーンに 2 番目のリードレプリカ `read-replica-2` (US-WEST-2) を作成します。
3. `read-replica-2` を使用して、チェーンに 3 番目のリードレプリカ `read-replica-3` (US-WEST-2) を作成します。

同じリージョンレプリカを作成してから、クロスリージョンレプリカを作成する

バージョン 14.1 以降 (`rpg-db-main`) の RDS for PostgreSQL DB インスタンス を使用して、以下を実行できます。

1. rpg-db-main (US-EAST-1) で開始し、チェーンに最初のリードレプリカ read-replica-1 (US-EAST-1) を作成します。
2. read-replica-1 (US-EAST-1) を使用して、チェーンに最初のクロスリージョンリードレプリカ read-replica-2 (US-WEST-2) を作成します。
3. read-replica-2 (US-WEST-2) を使用して、チェーンに 3 番目のリードレプリカ read-replica-3 (US-WEST-2) を作成します。

クロスリージョンリードレプリカの作成に関する制限事項

- データベースレプリカのクロスリージョンカスケーディングチェーンは、最大 2 つのリージョンにまたがることができ、最大 4 つのレベルがあります。4 つのレベルには、1 つのデータベースソースと 3 つのリードレプリカが含まれます。

カスケーディングリードレプリカを使用する利点

- 読み取りスケーラビリティの向上 – 読み取りクエリを複数のレプリカに分散することで、カスケーディングレプリケーションは負荷均衡に役立ちます。これにより、ライターデータベースの負荷を軽減することで、特に読み取り負荷の高いアプリケーションのパフォーマンスが向上します。
- 地理的分散 — カスケーディングレプリカは、異なる地理的場所に配置できます。これにより、プライマリデータベースから離れた場所にあるユーザーのレイテンシーが軽減され、ローカルリードレプリカが提供され、パフォーマンスとユーザーエクスペリエンスが向上します。
- 高可用性とディザスタリカバリ — プライマリサーバーに障害が発生した場合、レプリカをプライマリに昇格させ、継続性を確保できます。カスケーディングレプリケーションは、複数のフェイルオーバーオプションを提供し、システムの全体的な耐障害性を向上させることで、これをさらに強化します。
- 柔軟性とモジュール式の成長 — システムが成長するにつれて、プライマリデータベースの大規模な再設定を行わずに、新しいレプリカをさまざまなレベルで追加できます。このモジュール方式により、レプリケーション設定をスケーラブルかつ管理しやすい方法で拡張できます。

クロスリージョンリードレプリカを使用するためのベストプラクティス

- レプリカを昇格させる前に、追加のレプリカを作成します。これにより、時間を節約し、ワークロードを効率的に処理できます。

RDS for PostgreSQL のバージョンが異なる場合のストリーミングレプリケーションの仕組み

「PostgreSQL でのリードレプリカの設定」で説明したとおり、RDS for PostgreSQL は PostgreSQL のネイティブストリーミングレプリケーションプロトコルを使用して、ソース DB インスタンスから WAL データを送信します。リージョン内とクロスリージョンの両方のリードレプリカにソース WAL データを送信します。バージョン 9.4 では、レプリケーションプロセスのサポートメカニズムとして、PostgreSQL に物理レプリケーションスロットが導入されました。

物理レプリケーションスロットは、WAL データがすべてのリードレプリカで消費される前に、ソース DB インスタンスによってデータが削除されることを防ぎます。各リードレプリカは、ソース DB インスタンスに独自の物理スロットを持ちます。このスロットは、レプリカが必要とする可能性のある最も古い WAL (論理シーケンス番号、LSN) を追跡します。すべてのスロットと DB との接続が指定した WAL (LSN) を超過して進行すると、その LSN は次のチェックポイントで削除候補になります。

Amazon RDS は、Amazon S3 を使用して WAL データをアーカイブします。リージョン内リードレプリカの場合は、必要に応じて、このアーカイブされたデータを使用してリードレプリカを復旧できます。その例として、何らかの理由でソース DB とリードレプリカ間の接続が中断された場合が挙げられます。

次の表は、PostgreSQL のバージョンの相違点と、RDS for PostgreSQL で使用されるリージョン内およびクロスリージョンのサポートメカニズムの相違点の概要です。

バージョン	リージョン内	クロスリージョン
PostgreSQL 14.1 以降のバージョン	• レプリケーションスロット • Amazon S3 アーカイブ	• レプリケーションスロット
PostgreSQL 13 以前のバージョン	• Amazon S3 アーカイブ	• レプリケーションスロット

詳細については、「レプリケーションプロセスのモニタリングとチューニング」を参照してください。

PostgreSQL レプリケーションを制御するパラメータについて

以下のパラメータはレプリケーションプロセスに影響を与えます。また、リードレプリカがソース DB インスタンスの更新をどの程度反映するかを決定します。

max_wal_senders

`max_wal_senders` パラメータは、ストリーミングレプリケーションプロトコルに対して、ソース DB インスタンスが同時にサポートできる接続の最大数を指定します。

デフォルト値は RDS for PostgreSQL バージョンによって異なります。

- バージョン 13、14、15 の場合、デフォルト値は 20 です。
- バージョン 16 以降では、デフォルト値は 35 です。

このパラメータは、実際のリードレプリカの数よりわずかに大きな値に設定する必要があります。リードレプリカの数に対してこのパラメータの設定が低すぎる場合は、レプリケーションが停止します。

詳細については、PostgreSQL のドキュメントの「max_wal_senders」セクションを参照してください。

> **Note**
>
> `max_wal_senders` パラメータは静的であるため、変更を有効にするには、DB インスタンスを再起動する必要があります。

wal_keep_segments

`wal_keep_segments` パラメータは、ソース DB インスタンスが `pg_wal` ディレクトリで保持する先書きログ (WAL) ファイルの数を指定します。デフォルトの設定は 32 です。

`wal_keep_segments` がデプロイで十分な大きさの値に設定されていない場合、リードレプリカでのストリーミングに大幅な遅延が発生し、レプリケーションが停止します。その場合、Amazon RDS でレプリケーションエラーが報告され、リードレプリカで復旧が開始されます。これは、ソース DB インスタンスのアーカイブされた WAL データを Amazon S3 で再生することによって行われます。この復旧プロセスは、レプリケーションのストリーミングを続行するのに十分なだけリードレプリカの遅延が解消されるまで続きます。このプロセスを PostgreSQL

ログがキャプチャしている様子については、「例: リードレプリカがレプリケーションの中断から復旧する方法」で確認できます。

> **Note**
>
> PostgreSQL バージョン 13 では、`wal_keep_segments` パラメータの名前は `wal_keep_size` です。このパラメータも `wal_keep_segments` と同じ目的を果たしますが、デフォルト値は、ファイル数ではなくメガバイト (MB) (2048 MB) です。詳細については、PostgreSQL ドキュメントの「wal_keep_segments」と「wal_keep_size」を参照してください。

max_slot_wal_keep_size

`max_slot_wal_keep_size` パラメータは、RDS for PostgreSQL DB インスタンスが `pg_wal` ディレクトリで保持する WAL データの量を制御して、スロットを提供します。このパラメータは、レプリケーションスロットを使用する構成に使用されます。このパラメータのデフォルト値は `-1` です。つまり、ソース DB インスタンスに保持される WAL データの量は無制限です。レプリケーションスロットのモニタリングについては、「RDS for PostgreSQL DB インスタンスのレプリケーションスロットのモニタリング」を参照してください。

このパラメータの詳細については、PostgreSQL ドキュメントの「max_slot_wal_keep_size」を参照してください。

リードレプリカに WAL データを提供するストリームが中断した場合、PostgreSQL は復旧モードに切り替わります。リードレプリカの復元は、Amazon S3 のアーカイブ済み WAL データを使用するか、レプリケーションスロットに関連付けられている WAL データを使用して行われます。このプロセスが完了すると、PostgreSQL でストリーミングレプリケーションが再構築されます。

例: リードレプリカがレプリケーションの中断から復旧する方法

次の例では、リードレプリカの復旧プロセスを示すログの詳細を示します。この例は、同じ AWS リージョンで PostgreSQL バージョン 12.9 をソース DB として実行している RDS for PostgreSQL DB インスタンスの例です。そのため、レプリケーションスロットは使用されません。復旧プロセスは、リージョン内リードレプリカでバージョン 14.1 より前の PostgreSQL を実行している他の RDS for PostgreSQL DB インスタンスでも同じです。

リードレプリカがソース DB インスタンスとの接続を失った場合、Amazon RDS は問題を `FATAL: could not receive data from WAL stream` メッセージとして `ERROR: requested WAL`

segment ... has already been removed とともにログに記録します。太字の行は、アーカイブされた WAL ファイルを再生することで Amazon RDS によりレプリカが復旧されたことを示しています。

```
2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  switched WAL source from archive to stream
 after failure
2014-11-07 19:01:10 UTC::@:[11575]:LOG: started streaming WAL from primary at 1A/
D3000000 on timeline 1
2014-11-07 19:01:10 UTC::@:[11575]:FATAL: could not receive data from WAL stream:
ERROR:  requested WAL segment 000000010000001A000000D3 has already been removed
2014-11-07 19:01:10 UTC::@:[23180]:DEBUG: could not restore file "00000002.history"
 from archive: return code 0
2014-11-07 19:01:15 UTC::@:[23180]:DEBUG: switched WAL source from stream to archive
 after failure recovering 000000010000001A000000D3
2014-11-07 19:01:16 UTC::@:[23180]:LOG:  restored log file "000000010000001A000000D3"
 from archive
```

Amazon RDS がレプリカで遅延を解消するのに十分な、アーカイブされた WAL データを再生すると、リードレプリカへのストリーミングが再開されます。ストリーミングが再開されると、Amazon RDS によって次のようなエントリがログファイルに書き込まれます。

```
2014-11-07 19:41:36 UTC::@:[24714]:LOG:started streaming WAL from primary at 1B/
B6000000 on timeline 1
```

共有メモリを制御するパラメータの設定

設定したパラメータによって、トランザクション ID、ロック、準備済みトランザクションを追跡するための共有メモリのサイズが決まります。スタンバイインスタンスの共有メモリ構造は、プライマリインスタンスの共有メモリ構造と同じかそれ以上でなければなりません。これにより、リカバリ中に前者の共有メモリが不足することがなくなります。レプリカのパラメータ値がプライマリのパラメータ値よりも小さい場合、Amazon RDS はレプリカのパラメータを自動的に調整し、エンジンを再起動します。

影響を受けるパラメータは以下のとおりです。

- max_connections
- max_worker_processes
- max_wal_senders
- max_prepared_transactions

- max_locks_per_transaction

メモリ不足が原因でレプリカが RDS で再起動されるのを防ぐため、パラメータの変更を各レプリカにローリング再起動として適用することをお勧めします。パラメータ設定時には、次のルールを適用する必要があります。

- パラメータ値を増やす:
  - 必ず最初にすべてのリードレプリカのパラメータ値を増やし、すべてのレプリカのローリングリブートを実行する必要があります。次に、パラメータの変更をプライマリインスタンスに適用し、再起動します。
- パラメータ値を減らす:
  - まず、プライマリインスタンスのパラメータ値を減らし、再起動する必要があります。次に、パラメータの変更を関連するすべてのリードレプリカに適用し、ローリングリブートを実行します。

レプリケーションプロセスのモニタリングとチューニング

RDS for PostgreSQL DB インスタンスとリードレプリカを定期的にモニタリングすることを強くお勧めします。リードレプリカがソース DB インスタンスの変更に対応していることを確認する必要があります。Amazon RDS では、レプリケーションプロセスで中断が発生した場合に、リードレプリカを透過的に復旧できます。ただし、最善なのは、復旧の必要性を最初から避けることです。レプリケーションスロットを使用した復旧は、Amazon S3 アーカイブを使用するよりも高速ですが、復旧プロセスは読み取りパフォーマンスに影響する可能性があります。

リードレプリカが最新のソース DB インスタンスにどの程度対応しているかを判断するには、次の操作を行います。

- ソース DB インスタンスとレプリカ間の **ReplicaLag** の量を確認する。レプリカ遅延は、ソース DB インスタンスに対するリードレプリカの遅延時間 (秒単位) です。このメトリクスは、次のクエリの結果を返します。

  ```
  SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS "ReplicaLag";
  ```

  レプリカ遅延は、リードレプリカがソース DB インスタンスの変更をどの程度反映しているかを示します。これは、ソース DB インスタンスと特定のリードインスタンス間のレイテンシーの量です。レプリカ遅延の値が大きい場合は、ソース DB インスタンスとそのリードレプリカで使用され

る DB インスタンスクラスまたはストレージタイプ (またはその両方) の不一致を示している可能性があります。DB ソースインスタンスとすべてのリードレプリカの DB インスタンスクラスとストレージタイプは同じである必要があります。

レプリカ遅延は、断続的な接続問題の結果でもあります。Amazon CloudWatch のレプリケーションのラグをモニタリングするには、Amazon RDS `ReplicaLag` メトリクスを表示します。`ReplicaLag` についての詳細および Amazon RDS のその他のメトリクスについては、「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。

- PostgreSQL ログで、設定を調整するために使用できる情報を確認する。次の例に示すように、PostgreSQL ログでは、すべてのチェックポイントで、リサイクルされるトランザクションログファイルの数がキャプチャされます。

```
2014-11-07 19:59:35 UTC::@:[26820]:LOG:  checkpoint complete: wrote 376 buffers
 (0.2%);
0 transaction log file(s) added, 0 removed, 1 recycled; write=35.681 s, sync=0.013 s,
 total=35.703 s;
sync files=10, longest=0.013 s, average=0.001 s
```

この情報を使用して、指定された期間にリサイクルされるトランザクションファイルの数を把握できます。必要に応じて、`wal_keep_segments` の設定を変更できます。例えば、`checkpoint complete` の PostgreSQL ログが 5 分間隔で `35 recycled` を表示するとします。この場合、`wal_keep_segments` のデフォルト値 32 では、ストリーミングアクティビティのペースを維持するには不十分です。そのため、このパラメータの値を大きくする必要があります。

- Amazon CloudWatch を使用して、レプリケーションの問題を予測できるメトリクスをモニタリングする。PostgreSQL ログを直接分析する代わりに、Amazon CloudWatch を使用することで、収集されたメトリクスを確認できます。例えば、`TransactionLogsGeneration` メトリクスの値を確認すると、ソース DB インスタンスによって生成されている WAL データの量を把握できます。場合によっては、DB インスタンスのワークロードによって大量の WAL データが生成されることがあります。その場合、ソース DB インスタンスとリードレプリカの DB インスタンスクラスを変更する必要があります。高いネットワークパフォーマンス (10 Gbps) のインスタンスクラスを使用すると、レプリカの遅延を低減することができます。

RDS for PostgreSQL DB インスタンスのレプリケーションスロットのモニタリング

RDS for PostgreSQL のすべてのバージョンでは、クロスリージョンのリードレプリカにレプリケーションスロットを使用します。RDS for PostgreSQL 14.1 以降のバージョンでは、リージョン内のリードレプリカにレプリケーションスロットを使用します。リージョン内のリードレプリカ

は、Amazon S3 を使用して WAL データをアーカイブします。つまり、DB インスタンスとリードレプリカが PostgreSQL 14.1 以降を実行している場合、レプリケーションスロットと Amazon S3 アーカイブの両方を使用してリードレプリカを復旧できます。レプリケーションスロットを使用したリードレプリカの復旧は、Amazon S3 アーカイブからの復旧よりも高速です。そのため、レプリケーションスロットおよび関連するメトリクスをモニタリングすることをお勧めします。

RDS for PostgreSQL DB インスタンスのレプリケーションスロットは、次に示すように、pg_replication_slots ビューをクエリすることで表示できます。

```
postgres=> SELECT * FROM pg_replication_slots;
slot_name                  | plugin | slot_type | datoid | database | temporary |
 active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn |
 wal_status | safe_wal_size | two_phase
---------------------------+--------+-----------+--------+----------+-----------
+--------+------------+------+--------------+-------------+---------------------
+------------+--------------+-----------
rds_us_west_1_db_555555555 |        | physical  |        |          | f         | t
  |      13194 |      |              | 23/D8000060 |                     | reserved   |
               | f
(1 row)
```

wal_status/reserved の値は、max_wal_size パラメータの境界内のスロットで保持されている WAL データの量を示します。つまり、レプリケーションスロットは適切にサイズ設定されています。その他のステータス値は次のとおりです。

- extended – スロットは max_wal_size 設定を超過しますが、WAL データは保持されます。
- unreserved – スロットに必要な WAL データがすべて含まれていません。データの一部は次のチェックポイントで削除されます。
- lost – 必要な WAL データの一部が削除されました。スロットは使用できません。

wal_status の unreserved および lost 状態は、max_slot_wal_keep_size が負でない場合にのみ表示されます。

pg_replication_slots ビューには、レプリケーションスロットの現在の状態が表示されます。レプリケーションスロットのパフォーマンスを評価する場合、Amazon CloudWatch を使用すると、次のメトリクスをモニタリングできます。

- **OldestReplicationSlotLag** – 最も長い遅延が発生しているレプリカによって消費されていないソースの先書き口グ (WAL) データの量を表示します。

- **`TransactionLogsDiskUsage`** – WAL データに使用されているストレージの量を示します。リードレプリカが著しく遅れると、このメトリクスの値が大幅に増加する可能性があります。

RDS for PostgreSQL での Amazon CloudWatch とそのメトリクスの使用のについての詳細は、「Amazon CloudWatch を使用した Amazon RDS メトリクスのモニタリング」を参照してください。RDS for PostgreSQL DB インスタンスでストリーミングレプリケーションをモニタリングする方法の詳細については、AWS データベースブログの「Best practices for Amazon RDS PostgreSQL replication」(Amazon RDS PostgreSQL レプリケーションのベストプラクティス) を参照してください。

RDS for PostgreSQL での遅延レプリケーションの設定

概要と利点

RDS for PostgreSQL の遅延レプリケーション機能を使用すると、プライマリデータベースから 1 つ以上のスタンバイ (リードレプリカ) サーバーへのデータ変更のレプリケーションを意図的に遅延できます。これにより、データの破損、偶発的なデータ損失、またはすべてのレプリカにすぐに伝播される可能性のある誤ったトランザクションに対する貴重な保護が提供されます。

遅延レプリケーションは、以下の RDS for PostgreSQL バージョンでサポートされています。

- 14.19 以降の 14 バージョン
- 15.14 以降の 15 バージョン
- 16.10 以降の 16 バージョン
- 17.6 以降の 17 バージョン

レプリケーションプロセスにタイムラグを導入することで、データ関連のインシデントが DB クラスター全体に影響を与える前に、それを検出して対応する機会が得られます。遅延レプリケーションの主な利点は次のとおりです。

- 偶発的な削除、更新、その他の論理的なミスから復旧できます。
- DB クラスター全体の破損したデータの分散に対するバッファを提供します。
- 従来のバックアップ戦略を補完する追加の復旧ポイントオプションを提供します。
- 組織固有のニーズとリスク許容度に基づいて遅延期間を設定できます。

遅延レプリケーションの有効化と設定

RDS for PostgreSQL リードレプリカで遅延レプリケーションを有効にするには、次の手順に従います。

> **Note**
>
> カスケードされたリードレプリカの場合は、以下で説明するのと同じ `recovery_min_apply_delay` パラメータと手順を使用します。

遅延レプリケーションを有効にするには

1. 新しいカスタムパラメータグループを作成するか、既存のものを変更します。詳細については、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。
2. パラメータグループの `recovery_min_apply_delay` パラメータを設定します。
   - 希望する遅延の値をミリ秒単位で設定します。例えば、1 時間の遅延の場合は 3600000 です。
   - 許可される範囲: 0 ~ 86400000 ms (0 ~ 24 時間)
   - デフォルト: 0
3. パラメータグループを、遅延レプリケーション用に設定するリードレプリカインスタンスに適用します。
4. 変更を有効にするために、リードレプリカインスタンスを再起動します。

> **Note**
>
> `recovery_min_apply_delay` パラメータは動的です。インスタンスに既にアタッチされている既存のパラメータグループを変更すると、その変更は再起動を必要とせずにすぐに有効になります。ただし、インスタンスに新しいパラメータグループを適用する場合は、変更を有効にするために再起動する必要があります。

遅延レプリケーションリカバリの管理

遅延レプリケーションは、従来のポイントインタイムリカバリ方法が不十分な場合や時間がかかりすぎる場合に特に役立ちます。

遅延レプリケーション期間中は、次の PostgreSQL 関数を使用して復旧プロセスを管理できます。

- pg_wal_replay_pause(): 遅延レプリカの復旧プロセスを一時停止するようにリクエストします。
- pg_wal_replay_resume(): 以前に一時停止していた場合は、復旧プロセスを再起動します。
- pg_is_wal_replay_paused(): 復旧プロセスが現在一時停止されているかどうかを確認します。
- pg_get_wal_replay_pause_state(): 復旧プロセスの現在の状態を取得します (一時停止、リクエスト、一時停止なし)。

rds_superuser ロールを持つユーザーには、pg_wal_replay_pause() および pg_wal_replay_resume() に対する EXECUTE 権限があります。他のデータベースユーザーがこれらの関数にアクセスする必要がある場合は、rds_superuser ロールを付与する必要があります。rds_superuser ロールの詳細については、「rds_superuser ロールを理解する」を参照してください。

pg_is_wal_replay_paused() や pg_get_wal_replay_pause_state() などの他の関数へのアクセスには、rds_superuser ロールは必要ありません。

次の復旧ターゲットパラメータを使用して、遅延レプリカが復旧する時点を正確に制御できます。これらのパラメータは静的であり、変更を適用するにはデータベースを再起動する必要があります。

- recovery_target
- recovery_target_lsn
- recovery_target_name
- recovery_target_time
- recovery_target_xid
- recovery_target_inclusive

> **Important**
>
> 一度に指定できる復旧先パラメータは 1 つだけです。設定ファイルに複数の復旧先パラメータを設定すると、エラーが発生します。

計画に関する考慮事項

RDS for PostgreSQL で遅延レプリケーションを計画する場合は、次の点を考慮してください。

- `rdsrepladmin` 認証情報の自動ローテーション中 (90 日ごとに発生)、遅延リードレプリカが一時的に `REPLICATION_ERROR` 状態になることがあります。遅延レプリカに設定された遅延を維持するのに十分な WAL ログがある場合、WAL レシーバープロセスが一時停止し、ソースに WAL が蓄積される可能性があります。ストレージがいっぱいにならないように、レプリカのレプリケーションステータスとソースのストレージ消費量をモニタリングする必要があります。
- 遅延リードレプリカでシステムイベント (再起動や再起動など) が発生すると、設定された遅延期間が終了するまで WAL レシーバープロセスが非アクティブのままになる `REPLICATION_ERROR` 状態になります。この動作により、ソースインスタンスに WAL が蓄積され、ストレージが枯渇する可能性があります。以下の予防策を検討してください。
  - ソースインスタンスのストレージ使用率をモニタリングするように CloudWatch アラームを設定します。
  - ストレージの自動スケーリングを有効にして、予期しない WAL の増加を処理します。
  - レプリケーションスロットあたりの WAL 保持を制限するには、レプリケート元インスタンスの `max_slot_wal_keep_size` パラメータを設定します。
  - レプリケーションラグとスロットステータスを定期的にモニタリングします。
- 遅延が長くなると、レプリカの WAL ログが増加し、より多くのストレージが消費されます。CloudWatch アラームを使用したストレージスペースのモニタリング、自動スケーリングの有効化、または必要に応じてレプリカのキャッチアップを行います。
- 遅延リードレプリカを昇格する場合、`recovery_min_apply_delay` パラメータは受け入れられず、保留中のすべての WAL レコードがすぐに適用されます。
- `recovery_min_apply_delay` パラメータは、カスケードレプリケーション設定の各レベルで独立しています。レプリカに遅延を設定しても、カスケードされたレプリカの遅延には追加されません。

詳細については、「RDS for PostgreSQL リードレプリカのドキュメント」と「RDS for PostgreSQL ディザスタリカバリのドキュメント」を参照してください。

制限を理解する

Amazon RDS for PostgreSQL の遅延レプリケーション機能には、次の制限があります。

- ブルー/グリーンデプロイでは、遅延レプリケーションを設定するときに以下の制限があります。

  - グリーンソースインスタンス – パラメータグループで設定されていても、recovery_min_apply_delay parameter は無視されます。グリーンソースインスタンスの遅延設定は有効になりません。
  - グリーンレプリカインスタンス – recovery_min_apply_delay parameter は完全にサポートされ、PostgreSQL 設定ファイルに適用されます。遅延設定は、スイッチオーバーワークフロー中に想定どおりに機能します。
  - メジャーバージョンアップグレード用の RDS ブルー/グリーンデプロイ
- メジャーバージョンのアップグレード中、遅延したリードレプリカは自動的に終了され、ソースインスタンスが最小限のダウンタイムを確保するためにアップグレードプロセスを続行できるようになります。ソースインスタンスのアップグレードが完了したら、遅延レプリカを手動で再作成する必要があります。
- 遅延レプリケーションは、以下の機能と互換性がありません。
  - RDS for PostgreSQL の論理的なレプリケーション
  - RDS for PostgreSQL マルチ AZ クラスター (インバウンドレプリケーションとアウトバウンドレプリケーションの両方を含む)
  - Aurora PostgreSQL

RDS for PostgreSQL リードレプリカのトラブルシューティング

RDS for PostgreSQL リードレプリカに関するよくある問題のトラブルシューティングについてのアイデアをいくつか以下に紹介します。

リードレプリカの遅延の原因となるクエリを終了する

トランザクションの状態がアクティブまたはアイドルであり、データベース内で長時間実行されているトランザクションは、WAL レプリケーションプロセスへの妨害となる可能性があります。これは、レプリケーション遅延の悪化につながります。このため、PostgreSQL pg_stat_activity ビューを使用して、このようなトランザクションのランタイムを必ずモニタリングしてください。

プライマリインスタンスで次のようなクエリを実行して、長時間実行されているクエリのプロセス ID (PID) を検索します。

```
SELECT datname, pid,usename, client_addr, backend_start,
xact_start, current_timestamp - xact_start AS xact_runtime, state,
```

```
backend_xmin FROM pg_stat_activity WHERE state='active';
```

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

クエリの PID を特定したら、そのクエリを終了することができます。

プライマリインスタンスで次のようなクエリを実行して、長時間実行されているクエリを終了します。

```
SELECT pg_terminate_backend(PID);
```

Amazon RDS Optimized Reads による RDS for PostgreSQL のクエリパフォーマンスの向上

Amazon RDS Optimized Reads によって、RDS for PostgreSQL の高速クエリ処理を実現できます。RDS Optimized Reads を使用する RDS for PostgreSQL DB インスタンスまたはマルチ AZ DB クラスターは、これを使用しないものに比べて、クエリ処理を最大 2 倍高速化できます。

トピック

- PostgreSQL の RDS Optimized Reads の概要
- RDS Optimized Reads のユースケース
- RDS Optimized Reads のベストプラクティス
- RDS Optimized Reads の使用
- RDS Optimized Reads を使用する DB インスタンスのモニタリング
- PostgreSQL の RDS Optimized Reads についての制限事項

PostgreSQL の RDS Optimized Reads の概要

NVMe ベースの DB インスタンスクラスを使用する場合、RDS for PostgreSQL バージョン 15.2 以上、14.7 以上、13.10 以上では、Optimized Reads をデフォルトで利用できます。NVMe を使用するインスタンスを示すハードウェア仕様については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

RDS Optimized Reads が有効になっている RDS for PostgreSQL DB インスタンスまたはマルチ AZ DB クラスターを使用する場合、それがローカルの不揮発性メモリエクスプレス (NVMe) ベースのソリッドステートドライブ (SSD) ブロックレベルストレージを使用することで、クエリのパフォーマンスが最大 2 倍高速化されます。PostgreSQL によって生成された一時テーブルをローカルストレージに配置することで、クエリ処理を高速化できます。これにより、ネットワーク経由の Elastic Block Storage (EBS) へのトラフィックが減少します。

PostgreSQL では、一時オブジェクトは一時的な名前空間に割り当てられ、セッションの終了時に自動的に削除されます。ドロップ中の一時的な名前空間は、テーブル、関数、演算子、さらには拡張機能などのスキーマ修飾オブジェクトを含む、セッションに依存するオブジェクトをすべて削除します。

RDS for PostgreSQL では、`temp_tablespaces` パラメータは一時オブジェクトが格納されるこの一時的な作業領域に設定されます。

次のクエリは、テーブルスペースの名前とその場所を返します。

```
postgres=> show temp_tablespaces;
temp_tablespaces
--------------------
rds_temp_tablespace
(1 row)
```

rds_temp_tablespace は、NVMe ローカルストレージを指す RDS によって設定された表領域です。Parameter group 内のこのパラメータを AWS マネジメントコンソール を使用して変更し、rds_temp_tablespace 以外の任意のテーブルスペースを指すようにすることで、いつでも Amazon EBS ストレージに戻すことができます。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。SET コマンドを使用して、セッションレベルで temp_tablespaces パラメータの値を pg_default に変更することもできます。パラメータを変更すると、一時的な作業領域が Amazon EBS にリダイレクトされます。Amazon EBS に戻すことは、RDS インスタンスまたはクラスターのローカルストレージが特定の SQL 操作を実行するのに十分でない場合に役立ちます。

```
postgres=> SET temp_tablespaces TO 'pg_default';
SET
```

```
postgres=> show temp_tablespaces;

 temp_tablespaces
------------------
 pg_default
```

RDS Optimized Reads のユースケース

以下に、Optimized Reads を使用することでメリットが得られるユースケースをいくつか紹介します。

- テーブル共通式 (CTE)、派生テーブル、グループ化オペレーションを含む分析クエリ。
- アプリケーションの最適化されていないクエリを処理するリードレプリカ。
- GROUP BY や ORDER BY などの複雑な操作を伴うオンデマンドまたは動的なレポートクエリで、常に適切なインデックスを使用できるとは限らないもの。
- 内部の一時テーブルを使用するその他のワークロード。

- ソート用の CREATE INDEX または REINDEX 操作。

RDS Optimized Reads のベストプラクティス

RDS Optimized Reads を使用するベストプラクティスは次のとおりです。

- インスタンスストアが実行中にストレージ不足によって失敗した場合に備えて、読み取り専用クエリの再試行ロジックを追加します。
- CloudWatch メトリクスの FreeLocalStorage を使用して、インスタンスストアで使用可能なストレージ容量をモニタリングします。DB インスタンスまたはマルチ AZ DB クラスターのワークロードが原因でインスタンスストアが上限に達している場合は、より大きな DB インスタンスクラスを使用するように変更します。

RDS Optimized Reads の使用

シングル AZ DB インスタンスデプロイ、マルチ AZ DB インスタンスデプロイ、またはマルチ AZ DB クラスターデプロイで、NVMe ベースの DB インスタンスクラスのいずれかを使用して RDS for PostgreSQL DB インスタンスをプロビジョニングすると、DB インスタンスは自動的に RDS Optimized Reads を使用します。

マルチ AZ 配置については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

RDS Optimized Reads をオンにするには、次のいずれかの操作を行います。

- NVMe ベースの DB インスタンスクラスの 1 つを使用して、RDS for PostgreSQL DB インスタンスまたはマルチ AZ DB クラスターを作成します。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。
- NVMe ベースの DB インスタンスクラスの 1 つを使用して、既存の RDS for PostgreSQL DB インスタンスまたはマルチ AZ DB クラスターを変更します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS Optimized Reads は、ローカル NVMe SSD ストレージのある DB インスタンスクラスの 1 つ以上がサポートされているすべての AWS リージョン で使用できます。詳細については、「 DB インスタンスクラス」を参照してください。

最適化されていない読み取り RDS インスタンスに戻すには、RDS インスタンスまたはクラスターの DB インスタンスクラスを、データベースワークロードの EBS ストレージのみをサポートする同様のインスタンスクラスに変更します。例えば、現在の DB インスタンスクラスが db.r6gd.4xlarge の場合、db.r6g.4xlarge を選択して元に戻します。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS Optimized Reads を使用する DB インスタンスのモニタリング

RDS Optimized Reads を使用する DB インスタンスは、次の CloudWatch メトリクスでモニタリングできます。

- `FreeLocalStorage`
- `ReadIOPSLocalStorage`
- `ReadLatencyLocalStorage`
- `ReadThroughputLocalStorage`
- `WriteIOPSLocalStorage`
- `WriteLatencyLocalStorage`
- `WriteThroughputLocalStorage`

これらのメトリクスでは、利用可能なインスタンスストアストレージ、IOPS、スループットに関するデータを提供します。これらのメトリクスの詳細については、「Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス」を参照してください。

ローカルストレージの現在の使用状況をモニタリングするには、データベースにログインして次のクエリを実行します。

```
SELECT
    spcname AS "Name",
    pg_catalog.pg_size_pretty(pg_catalog.pg_tablespace_size(oid)) AS "size"
FROM
    pg_catalog.pg_tablespace
WHERE
    spcname IN ('rds_temp_tablespace');
```

一時ファイルとその使用方法の詳細については、「PostgreSQL による一時ファイルの管理」を参照してください。

PostgreSQL の RDS Optimized Reads についての制限事項

PostgreSQL の RDS Optimized Reads には次の制限事項が適用されます。

- トランザクションは、インスタンスストアが満杯になると エラーになる可能性があります。

Amazon RDS の PostgreSQL にデータをインポートする

Amazon RDS に移動させる既存の PostgreSQL デプロイがあるとします。タスクの複雑さは、データベースのサイズと転送するデータベースオブジェクトの種類に依存しています。例えば、データベースにギガバイトのオーダーのデータセット、さらにストアドプロシージャとトリガーが含まれているとします。このようなデータベースは、単純なデータベース (数メガバイトのテストデータを含むだけで、トリガーやストアドプロシージャを含まないもの) よりも複雑になります。

次の条件で、ネイティブ PostgreSQL データベース移行ツールを使用することをお勧めします。

- ターゲットデータベースエンジンと同じデータベースエンジンを持つデータベースから移行する、同機種移行である。
- データベース全体を移行する。
- ネイティブツールでは、最小のダウンタイムでシステムを移行することができます。

他の多くの場合、データベースの移行には、AWS Database Migration Service (AWS DMS) を使用することが最良のアプローチとなります。AWSDMS により、ダウンタイムなしでデータベースを移行できます。また、多くのデータベースエンジンでは、ターゲットデータベースへの切り替え準備ができるまで、進行中のレプリケーションを続行することができます。AWS DMS を使用することで、同じデータベースエンジン、または異なるデータベースエンジンへの移行が可能です。ソースデータベースとは別のデータベースエンジンへ移行する場合は、AWS Schema Conversion Tool (AWS SCT) を使用できます。AWS SCT を使用して、AWS DMS で移行されないスキーマオブジェクトを移行します。AWS DMS の詳細については、「AWS Database Migration Service とは」を参照してください。

DB パラメータグループを変更し、次のインポート専用の設定を含めます。DB インスタンスサイズの最も効率的な設定を見つけるために、パラメータ設定をテストする必要があります。さらに、インポートが完了したら、これらのパラメータを本番稼働用の値に戻す必要があります。

DB インスタンスの設定を次のように変更します。

- DB インスタンスのバックアップを無効にします (backup_retention を 0 に設定します)。
- マルチ AZ を無効にする。

次の設定を含むように DB パラメータグループを変更します。これらの設定は、データのインポート時にのみ使用してください。DB インスタンスサイズの最も効率的な設定を見つけるために、パラ

メータ設定をテストする必要があります。さらに、インポートが完了したら、これらのパラメータを本番稼働用の値に戻す必要があります。

Parameter	インポート時の推奨値	説明
maintenance_work_mem	524288、1048576、2097152、または 4194304 (KB 単位)。これらの設定は、512 MB、1 GB、2 GB、および 4 GB と同等です。	この設定の値は、ホストのサイズによって異なります。このパラメータは、CREATE INDEX ステートメントで使用され、各パラレルコマンドがこの量のメモリを使用できます。この設定値が大きすぎてメモリ不足が生じることのないように、最適な値を計算します。
max_wal_size	256 (バージョン 9.6 の場合)、4096 (バージョン 10 以降の場合)	自動チェックポイント中に WAL を拡張するための最大サイズ。このパラメータを増やすと、クラッシュ回復に必要な時間が長く可能性があります。PostgreSQL 9.6 以降では、このパラメータは checkpoint_segments を置き換えられます。 PostgreSQL バージョン 9.6 の場合、この値は 16 MB 単位です。それ以降のバージョンでは、値は 1 MB 単位です。例えば、バージョン 9.6 では、128 は、それぞれ 16 MB のサイズである 128 個のチャンクを意味します。バージョン 12.4 では、2048 は、それぞれ 1 MB のサイズである 2048 個のチャンクを意味します。
checkpoint_timeout	1800	この値に設定すると、WAL ローテーションの頻度を低くすることができます。
synchronous_commit	オフ	この設定を無効にすると、書き込みが速くなります。このパラメータをオフにすると、サーバークラッシュ時にデータが損失するリスクを下げることができます (FSYNC はオフにしないでください)。

Parameter	インポート時の推奨値	説明
wal_buffers	8192	この値は、8 KB 単位です。これも WAL の生成速度に貢献します。
autovacuum	0	リソースが使用されないように、データのロード時に PostgreSQL の自動バキュームパラメータを無効にします。

これらの設定で、pg_dump -Fc (圧縮) または pg_restore -j (パラレル) コマンドを使用します。

> **Note**
>
> PostgreSQL コマンド pg_dumpall の実行には SUPER_USER 権限が必要ですが、この権限は DB インスタンスの作成時に付与されません。そのため、このコマンドをデータのインポートに使用することはできません。

トピック

- Amazon EC2 インスタンスから PostgreSQL データベースをインポートする
- \copy コマンドを使用して PostgreSQL DB インスタンスのテーブルにデータをインポートする
- Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする
- DB インスタンス 間での PostgreSQL データベースの移行

Amazon EC2 インスタンスから PostgreSQL データベースをインポートする

Amazon EC2 インスタンス上の PostgreSQL サーバーにデータがあり、そのデータを PostgreSQL DB インスタンスに移動する場合は、次のプロセスに従ってデータを移行できます。

1. pg_dump を使用して、ロードするデータを格納したファイルを作成する
2. ターゲット DB インスタンスを作成する
3. psql を使用して、DB インスタンスにデータベースを作成し、データをロードする

4. DB インスタンスの DB スナップショットを作成する

次のセクションでは、上記の各ステップについて詳しく説明します。

ステップ 1: ロードするデータが含まれている pg_dump を使用してファイルを作成する

pg_dump ユーティリティでは、COPY コマンドを使用して、PostgreSQL データベースのスキーマとデータダンプを作成します。pg_dump によって生成されるダンプスクリプトは、同じ名前のデータベースにデータをロードし、テーブル、インデックス、外部キーを再作成します。pg_restore コマンドと -d パラメータを使用して、データを別の名前でデータベースに復元できます。

データダンプの作成前に、ダンプするテーブルに対してクエリを実行して行数を取得し、ターゲット DB インスタンスでその行数を確認できるようにする必要があります。

以下のコマンドでは、mydb2 というデータベース用に mydb2dump.sql というダンプファイルを作成しています。

```
prompt>pg_dump dbname=mydb2 -f mydb2dump.sql
```

ステップ 2: ターゲット DB インスタンスを作成する

Amazon RDS コンソール、AWS CLI、または API のいずれかを使用して、ターゲット PostgreSQL DB インスタンスを作成します。バックアップの保持設定を 0 にし、マルチ AZ を無効にして、インスタンスを作成します。これにより、データのインポートが高速化されます。データをダンプする前に、インスタンスにデータベースを作成する必要があります。データベースは、ダンプしたデータが含まれていたデータベースと同じ名前で作成できます。または、別の名前でデータベースを作成できます。この場合は、pg_restore コマンドと -d パラメータを使用して、新しい名前のデータベース内にデータを復元します。

例えば、データベースのダンプ、復元、名前変更に以下のコマンドを使用できます。

```
pg_dump -Fc -v -h [endpoint of instance] -U [master username] [database]
 > [database].dump
createdb [new database name]
pg_restore -v -h [endpoint of instance] -U [master username] -d [new database
 name] [database].dump
```

ステップ 3: psql を使用して DB インスタンスにデータベースを作成し、データをロードする

pg_dump コマンドの実行に使用した同じ接続を使用して、ターゲット DB インスタンスに接続し、データベースを再作成できます。psql により、マスターユーザー名とマスターパスワードを使用して DB インスタンスにデータベースを作成できます。

以下の例では、psql と、mydb2dump.sql という名前のダンプファイルを使用して、mypginstance という PostgreSQL DB インスタンスに mydb2 というデータベースを作成しています。

Linux、macOS、Unix の場合:

```
psql \
   -f mydb2dump.sql \
   --host mypginstance.555555555555.aws-region.rds.amazonaws.com \
   --port 8199 \
   --username myawsuser \
   --password password \
   --dbname mydb2
```

Windows の場合:

```
psql ^
   -f mydb2dump.sql ^
   --host mypginstance.555555555555.aws-region.rds.amazonaws.com ^
   --port 8199 ^
   --username myawsuser ^
   --password password ^
   --dbname mydb2
```

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外のパスワードを指定してください。

ステップ 4: DB インスタンスの DB スナップショットを作成する

データが DB インスタンスにロードされたことを確認したら、ターゲット PostgreSQL DB インスタンスの DB スナップショットを作成することをお勧めします。DB スナップショットは DB インスタ

ンスの完全なバックアップであり、DB インスタンスを既知の状態に復元するために使用できます。ロード直後に DB スナップショットを作成しておくと、何らかの事故のときにそのスナップショットを使用すれば、データを再ロードせずに済みます。また、そのスナップショットを使用して、新しい DB インスタンスをシードすることもできます。DB スナップショットの作成については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

\copy コマンドを使用して PostgreSQL DB インスタンスのテーブルにデータをインポートする

PostgreSQL の \copy コマンドは、psql の対話型クライアントツールでメタコマンドを利用できます。\copy を使うと、RDS for PostgreSQL DB インスタンスで、テーブルにデータをインポートできます。\copy コマンドを使うには、まず対象のDB インスタンスにテーブル構造を作成して、\copy がデータをコピーする先を用意する必要があります。

\copy を使用すると、クライアントのワークステーションにエクスポートして保存しておいた、カンマ区切り値 (CSV) 形式のファイルなどから、データを読み込むことができます。

CSV データを対象の RDS for PostgreSQL DB インスタンスにインポートするには、まず psql を使用して、対象の DB インスタンスに接続します。

```
psql --host=db-instance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=target-db
```

その後、次のパラメータを指定して \copy コマンドを実行し、対象のデータとその形式を識別します。

- target_table — CSV ファイルからコピーされるデータを受け取るテーブルの名前。
- column_list — テーブルの列の仕様。
- 'filename' — ローカルワークステーションにある CSV ファイルの絶対パス。

```
 \copy target_table from '/path/to/local/filename.csv' WITH DELIMITER ',' CSV;
```

CSV ファイルに列見出しがある場合は、このバージョンのコマンドとパラメータを使用できます。

```
\copy target_table (column-1, column-2, column-3, ...)
    from '/path/to/local/filename.csv' WITH DELIMITER ',' CSV HEADER;
```

\copy コマンドが失敗した場合は、PostgreSQL はエラーメッセージを出力します。

以下の例で示すように、\copy メタコマンドを指定した psql コマンドで、データベースプレビュー環境に新しい DB インスタンスを作成します。この例では、ソーステーブル名として source-table、.csv ファイルとして source-table.csv、ターゲットデータベースとして target-db を使用しています。

Linux、macOS、Unix の場合:

```
$psql target-db \
    -U <admin user> \
    -p <port> \
    -h <DB instance name> \
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

Windows の場合:

```
$psql target-db ^
    -U <admin user> ^
    -p <port> ^
    -h <DB instance name> ^
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

\copy コマンドの詳細については、PostgreSQL のドキュメント の 「psql」ページにある、「メタコマンド」セクションを参照してください。

Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする

Amazon Simple Storage Service を使用して保存されたデータを、 RDS for PostgreSQL DB インスタンス上のテーブルにインポートできます。これを行うには、 RDS for PostgreSQL aws_s3拡張機能を最初にインストールします。この拡張機能には、Amazon S3 バケットからのデータのインポートに使用する関数が含まれます。バケットとは、Amazon S3 のオブジェクトおよびファイルのコンテナです。データは、カンマ区切り値 (CSV) ファイル、テキストファイル、または圧縮 (gzip) ファイルでインポートできます。次に、拡張機能のインストール方法と、Amazon S3 からテーブルにデータをインポートする方法について説明します。

Amazon S3 から RDS for PostgreSQL にインポートするには、データベースで PostgreSQL バージョン 10.7 以降を実行している必要があります。

Amazon S3 にデータが保存されていない場合は、まずバケットを作成し、データを保存する必要があります。詳細については、Amazon Simple Storage Service コンソールユーザーガイドの以下のトピックを参照してください。

- バケットの作成
- バケットにオブジェクトを追加する

Amazon S3 からのクロスアカウントインポートがサポートされています。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「クロスアカウントアクセス許可の付与」を参照してください。

S3 からデータをインポートする際は、カスタマーマネージドキーを暗号化に使用できます。詳細については、Amazon Simple Storage Service ユーザーガイドの「AWS KMS に保存される KMS キー」を参照してください。

トピック

- aws_s3 拡張機能のインストール
- Amazon S3 データからのデータのインポートの概要
- Amazon S3 バケットへのアクセスを設定する
- Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする
- 関数リファレンス

aws_s3 拡張機能のインストール

RDS for PostgreSQL DB インスタンスで Amazon S3 を使用する前に、aws_s3 拡張機能をインストールする必要があります。この拡張機能には、Amazon S3 からデータをインポートするための関数が含まれます。また、 RDS for PostgreSQL DB インスタンスから Amazon S3 バケットへデータをエクスポートするための関数も含まれています。詳しくは、「RDS for PostgreSQL DB インスタンスから Amazon S3 へのデータのエクスポート」を参照してください。aws_s3 拡張機能は aws_commons 拡張機能の一部のヘルパー関数に依存しており、必要に応じて自動的にインストールされます。

aws_s3 拡張機能をインストールするには

1. rds_superuser 権限があるユーザーとして、psql (または pgAdmin) を使用して RDS for PostgreSQL DB インスタンスに接続します。設定プロセス中にデフォルトの名前を保持している場合は、postgres として接続します。

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 拡張機能をインストールするには、次のコマンドを実行します。

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. 拡張機能がインストールされていることを確認するには、psql \dx メタコマンドを使用します。

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Amazon S3 からデータをインポートし、データを Amazon S3 にエクスポートするための関数が使用できるようになりました。

Amazon S3 データからのデータのインポートの概要

S3 データを Amazon RDS にインポートするには

まず、関数で指定する必要がある詳細情報を収集します。この情報には、 RDS for PostgreSQL DB インスタンスのテーブルの名前、バケット名、ファイルパス、ファイルタイプ、Amazon S3 データが保存される AWS リージョンが含まれます。詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトの表示」を参照してください。

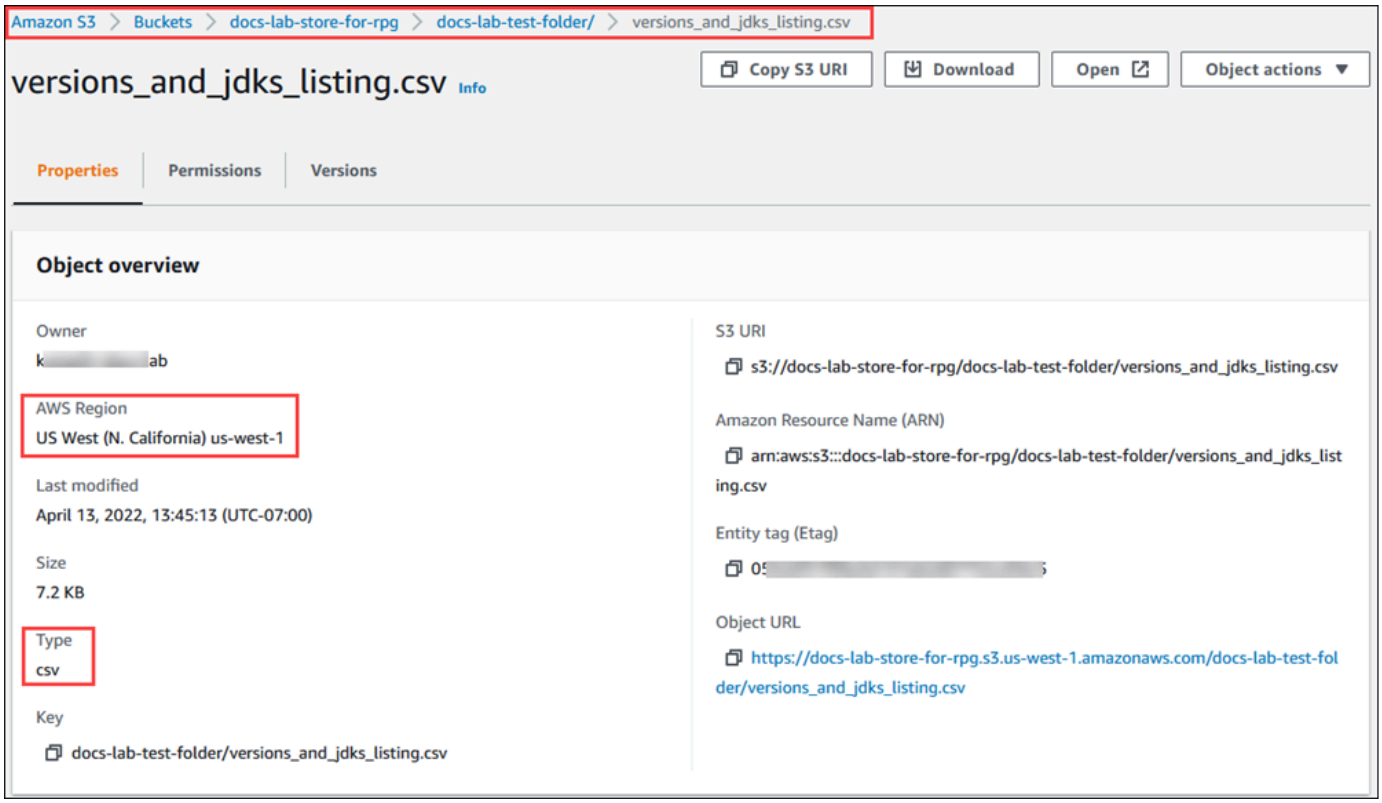
> **Note**
>
> Amazon S3 からのマルチパートデータインポートは現在サポートされていません。

1. `aws_s3.table_import_from_s3` 関数によってデータがインポートされるテーブルの名前を取得します。例えば、次のコマンドにより、後の手順で使用されるテーブル `t1` が作成されます。

```
postgres=> CREATE TABLE t1
    (col1 varchar(80),
    col2 varchar(80),
    col3 varchar(80));
```

2. Amazon S3 バケットの詳細とインポートするデータを取得します。これを実行するには、Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開き、[Bucket] (バケット) を選択します。リストで、データを含むバケットを探します。バケットを選択し、オブジェクト概要ページを開き、[Properties] (プロパティ) を選択します。

   バケット名、パス、AWS リージョン、およびファイルタイプを書き留めておきます。IAM ロールによる Amazon S3 へのアクセスを設定するには、後で Amazon リソースネーム (ARN) が必要になります。詳細については、「Amazon S3 バケットへのアクセスを設定する」を参照してください。次のイメージは例を示しています。



3. AWS CLI コマンド `aws s3 cp` を使用して、Amazon S3 バケットのデータへのパスを確認できます。情報が正しい場合、このコマンドは Amazon S3 ファイルのコピーをダウンロードします。

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. RDS for PostgreSQL DB インスタンスに対するアクセス許可を設定して、Amazon S3 バケット上のファイルへのアクセスを許可します。これを行うには、AWS Identity and Access Management (IAM) ロールまたはセキュリティ認証情報を使用します。詳しくは、「Amazon S3 バケットへのアクセスを設定する」を参照してください。
5. 収集したパスと他の Amazon S3 オブジェクトの詳細 (ステップ 2 を参照) を `create_s3_uri` 関数で指定し、Amazon S3 URI オブジェクトを構成します。この関数の詳細については、「aws_commons.create_s3_uri」を参照してください。psql セッション中にこのオブジェクトを構成する例は次のとおりです。

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

次のステップでは、このオブジェクト (`aws_commons._s3_uri_1`) を `aws_s3.table_import_from_s3` 関数に渡して、データをテーブルにインポートします。

6. `aws_s3.table_import_from_s3` 関数を呼び出して、Amazon S3 からテーブルにデータをインポートします。参考情報については、「aws_s3.table_import_from_s3」を参照してください。例については、Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする を参照してください。

Amazon S3 バケットへのアクセスを設定する

Amazon S3 ファイルからデータをインポートするには、RDS for PostgreSQL DB インスタンスに、ファイルが含まれている Amazon S3 バケットへのアクセス許可を与える必要があります。次のトピックで説明する 2 つの方法のいずれかで、Amazon S3 バケットへのアクセスを提供します。

トピック

- IAM ロールを使用した Amazon S3 バケットへのアクセス
- セキュリティ認証情報を使用して Amazon S3 バケットにアクセスする
- Amazon S3 へのアクセスのトラブルシューティング

IAM ロールを使用した Amazon S3 バケットへのアクセス

Amazon S3 ファイルからデータをロードするには、ファイルが含まれる Amazon S3 バケットへのアクセス許可を RDS for PostgreSQL DB インスタンスに与えます。こうすれば、追加の認証情報を管理したり、aws_s3.table_import_from_s3 関数呼び出しで提供したりする必要はありません。

これを行うには、Amazon S3 バケットへのアクセスを提供する IAM ポリシーを作成します。IAM ロールを作成して、ポリシーをロールにアタッチします。次に、IAM ロールを DB インスタンスに割り当てます。

IAM ロールを使用して、Amazon S3 へのアクセス権を RDS for PostgreSQL DB インスタンスに付与するには

1. IAM ポリシーを作成します。

   ポリシーは、RDS for PostgreSQL DB インスタンスに Amazon S3 へのアクセスを許可するバケットとオブジェクトのアクセス許可を付与します。

   ポリシーに、Amazon S3 バケットから Amazon RDS へのファイル転送を許可ための次の必須アクションを含めます。

   - `s3:GetObject`
   - `s3:ListBucket`

   ポリシーに次のリソースを含めて、Amazon S3 バケットとバケット内のオブジェクトを識別します。これは、Amazon S3 にアクセスするための Amazon リソースネーム (ARN) 形式を示しています。

   - arn:aws:s3:::*amzn-s3-demo-bucket*
   - arn:aws:s3:::*amzn-s3-demo-bucket*/*

   RDS for PostgreSQL の IAM ポリシーの作成の詳細については、「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。IAM ユーザーガイドの「チュートリアル: はじめてのカスタマー管理ポリシーの作成とアタッチ」も参照してください。

   以下の AWS CLI コマンドでは、これらのオプションを指定して、`rds-s3-import-policy` という名前の IAM ポリシーを作成します。*amzn-s3-demo-bucket* という名前のバケットへのアクセスを許可します。

> **Note**
>
> このコマンドによって返されるポリシー のAmazon リソースネーム (ARN) をメモしておきます。ポリシーを IAM ロールにアタッチする場合、後続のステップで ARN が必要です。

Example

Linux、macOS、Unix の場合:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Windows の場合:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
```

```
        "Action": [
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }'
```

2. IAM ロールを作成します。

   これを行うと、Amazon RDS がユーザーに代わってこの IAM ロールを引き受け、Amazon S3 バケットにアクセスできます。詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

   リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与するリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

   グローバル条件コンテキストキーの両方を使用し、aws:SourceArn の値にアカウント ID が含まれている場合、同じポリシーステートメントで使用する場合は、aws:SourceArn の値と aws:SourceAccount の値のアカウントでは同じアカウント ID を使用する必要があります。

   - 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
   - そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

   ポリシーでは、必ずリソースの完全な ARN を持つ aws:SourceArn グローバル条件コンテキストキーを使用してください。以下の例は、AWS CLI コマンドを使用して、rds-s3-import-role という名前のロールを作成する方法を示しています。

   Example

   Linux、macOS、Unix の場合:

```
aws iam create-role \
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
             }
       }
     ]
   }'
```

Windows の場合:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
             }
       }
     ]
```

```
    }'
```

3. 作成した IAM ポリシーを、作成した IAM ロールにアタッチします。

   次の AWS CLI コマンドは、先ほどのステップで作成したポリシーを `rds-s3-import-role` という名前のロールに添付し、*`your-policy-arn`* を前のステップでメモしたポリシー ARN に置き換えます。

   Example

   Linux、macOS、Unix の場合:

```
aws iam attach-role-policy \
   --policy-arn your-policy-arn \
   --role-name rds-s3-import-role
```

   Windows の場合:

```
aws iam attach-role-policy ^
   --policy-arn your-policy-arn ^
   --role-name rds-s3-import-role
```

4. DB インスタンスに IAM ロールを追加します。

   これを行うには、以下で説明するように、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

コンソールを使用して PostgreSQL DB インスタンスの IAM ロールを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 詳細を表示するには、PostgreSQL DB インスタンスの名前を選択します。
3. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、このインスタンスに [IAM ロールを追加] で追加するロールを選択します。
4. [Feature] で、[s3Import] を選択します。
5. [Add role] を選択します。

AWS CLI

CLI を使用して PostgreSQL DB インスタンスの IAM ロールを追加するには

- 次のコマンドを使用して、`my-db-instance` という名前の PostgreSQL DB インスタンスにロールを追加します。*`your-role-arn`* を、以前のステップで書き留めたロール ARN に置き換えます。`s3Import` オプションの値に `--feature-name` を使用します。

  Example

  Linux、macOS、Unix の場合:

  ```
  aws rds add-role-to-db-instance \
     --db-instance-identifier my-db-instance \
     --feature-name s3Import \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Windows の場合:

  ```
  aws rds add-role-to-db-instance ^
     --db-instance-identifier my-db-instance ^
     --feature-name s3Import ^
     --role-arn your-role-arn ^
     --region your-region
  ```

RDS API

Amazon RDS API を使用して PostgreSQL DB インスタンスに IAM ロールを追加するには、AddRoleToDBInstance オペレーションを呼び出します。

セキュリティ認証情報を使用して Amazon S3 バケットにアクセスする

必要に応じて、IAM ロールでアクセスを提供する代わりに、セキュリティ認証情報を使用して Amazon S3 バケットへのアクセスを提供できます このためには、aws_s3.table_import_from_s3 関数呼び出しで `credentials` パラメータを指定します。

`credentials` パラメータは、型の構造体 `aws_commons._aws_credentials_1` で、AWS 認証情報を含みます。aws_commons.create_aws_credentials 関数を使用し

て、aws_commons._aws_credentials_1 構造でアクセスキーおよびシークレットキーを設定します。以下に例を示します。

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

aws_commons._aws_credentials_1 構造を作成したら、以下に示すように、aws_s3.table_import_from_s3 関数を credentials パラメータと共に使用してデータをインポートします。

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

または、aws_commons.create_aws_credentials 関数の呼び出しのインラインを aws_s3.table_import_from_s3 関数の呼び出し内に含めることもできます。

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Amazon S3 へのアクセスのトラブルシューティング

Amazon S3 からデータをインポートしようとしたときに接続の問題が発生した場合は、次の推奨事項を参照してください。

- Amazon RDS のアイデンティティおよびアクセスのトラブルシューティング
- Amazon Simple Storage Service ユーザーガイド の「Troubleshooting Amazon S3」
- IAM ユーザーガイド の Amazon S3 のトラブルシューティングと IAM

Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする

aws_s3 拡張機能の table_import_from_s3 関数を使用して Amazon S3 バケットからデータをインポートします。参考情報については、「aws_s3.table_import_from_s3」を参照してください。

> **Note**
>
> 以下の例では、IAM ロールメソッドを使用して、Amazon S3 バケットへのアクセスを許可します。したがって、aws_s3.table_import_from_s3 関数呼び出しには認証情報パラメータは含まれません。

次の例は、代表的な例を示しています。

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

パラメータは次のとおりです。

- t1 - データのコピー先となる PostgreSQL DB インスタンス内のテーブルの名前。
- '' - データベーステーブル内の列のオプションのリスト。S3 データをコピーする列とテーブル列を指定するには、このパラメータを使用します。列を指定しない場合は、すべての列がテーブルにコピーされます。列のリストの使用例については、カスタム区切り文字を使用する Amazon S3 ファイルをインポートする を参照してください。
- (format csv) - PostgreSQL COPY 引数。このコピープロセスでは、PostgreSQL COPY コマンドの引数と形式を使用してデータをインポートします。フォーマットとしては、この例のようなカンマ区切り値 (CSV)、テキスト、およびバイナリを指定できます。デフォルトではテキストに設定されています。
- s3_uri - Amazon S3 ファイルを識別する情報を含む構造。aws_commons.create_s3_uri 関数を使用して s3_uri 構造を作成する例については、「Amazon S3 データからのデータのインポートの概要」を参照してください。

この関数の詳細については、「aws_s3.table_import_from_s3」を参照してください。

このaws_s3.table_import_from_s3関数はテキストを返します。Amazon S3 バケットからインポートする他の種類のファイルを指定するには、次の例のいずれかを参照してください。

> **Note**
>
> 0 バイトファイルをインポートすると、エラーが発生します。

トピック

- カスタム区切り文字を使用する Amazon S3 ファイルをインポートする
- Amazon S3 圧縮 (gzip) ファイルをインポートする
- エンコードされた Amazon S3 ファイルをインポートする

カスタム区切り文字を使用する Amazon S3 ファイルをインポートする

以下の例では、カスタム区切り文字を使用するファイルのインポート方法を示します。また、column_list 関数の aws_s3.table_import_from_s3 パラメータを使用して、データベースのデータを置く場所を制御する方法を示します。

この例では、次の情報が Amazon S3 ファイル内のパイプ区切りの列に編成されているとします。

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

カスタム区切り文字を使用するファイルをインポートするには

1. インポートされたデータのテーブルをデータベースに作成します。

   ```
   postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
   ```

2. データを Amazon S3 からインポートするには、次の形式の aws_s3.table_import_from_s3 関数を使用します。

   または、aws_commons.create_s3_uri 関数の呼び出しのインラインを aws_s3.table_import_from_s3 関数の呼び出し内に含めて、ファイルを指定することもできます。

   ```
   postgres=> SELECT aws_s3.table_import_from_s3(
      'test',
   ```

```
   'a,b,d,e',
   'DELIMITER ''|''',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
east-2')
);
```

データが、次の列のテーブル内に入りました。

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Amazon S3 圧縮 (gzip) ファイルをインポートする

以下の例では、gzip で圧縮されているファイルを Amazon S3 からインポートする方法を示します。インポートするファイルには、次の Amazon S3 メタデータが必要です。

- キー: `Content-Encoding`
- 値: `gzip`

AWS マネジメントコンソール を使用してファイルをアップロードする場合、通常このメタデータは、システムにより適用されます。AWS マネジメントコンソール、AWS CLI、または API による Amazon S3 へのファイルのアップロードについては、「Amazon Simple Storage Service ユーザーガイド」の「オブジェクトのアップロード」を参照してください。

Amazon S3 のメタデータに関する情報、およびシステム提供メタデータの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 コンソールでのオブジェクトメタデータの編集」を参照してください。

以下に示されているように、gzip ファイルを RDS for PostgreSQL DB インスタンスにインポートします。

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
```

```
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

エンコードされた Amazon S3 ファイルをインポートする

以下の例では、Windows-1252 でエンコードされているファイルを Amazon S3 からインポートする方法を示します。

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

関数リファレンス

関数

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Amazon S3 データを Amazon RDS テーブルにインポートします。aws_s3 拡張機能には、aws_s3.table_import_from_s3 関数が含まれます。戻り値はテキストです。

構文

必須のパラメータは、table_name、column_list、options です。これらのパラメータを使用して、データベースを特定し、データをテーブルにコピーする方法を指定します。

また、次のパラメータを使用することもできます。

- s3_info パラメータは、インポートする Amazon S3 ファイルを指定します。このパラメータを使用する場合、PostgreSQL DB インスタンスの IAM ロールを使用して、Amazon S3 へのアクセス権を付与します。

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
```

```
   s3_info aws_commons._s3_uri_1
)
```

- `credentials` パラメータは、Amazon S3 にアクセスするための認証情報を指定します。このパラメータを使用する場合、IAM ロールは使用しません。

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

パラメータ

table_name

データのインポート先となる PostgreSQL データベーステーブルの名前を含む必須のテキスト文字列。

column_list

データをコピーする PostgreSQL データベーステーブル列のオプションリストを含む必須のテキスト文字列。文字列が空の場合、テーブルの列がすべて使用されます。例については、「カスタム区切り文字を使用する Amazon S3 ファイルをインポートする」を参照してください。

オプション

PostgreSQL COPY コマンドの引数を含む必須のテキスト文字列。これらの引数は PostgreSQL のテーブルにデータをコピーする方法を指定します。詳細については、「PostgreSQL COPY ドキュメント」を参照してください。

s3_info

S3 オブジェクトに関する以下の情報を含む `aws_commons._s3_uri_1` 複合型。

- `bucket` - ファイルを含む Amazon S3 バケット名。
- `file_path` - Amazon S3ファイルのパスを含むファイル名。
- `region` - ファイルがある AWS リージョン。AWS リージョン名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

credentials

インポートオペレーションに使用する次の認証情報を含む aws_commons._aws_credentials_1 複合型。

- アクセスキー
- シークレットキー
- セッショントークン

aws_commons._aws_credentials_1 複合構造を作成する方法については、「aws_commons.create_aws_credentials」を参照してください。

代替構文

テストしやすいように、s3_info パラメータや credentials パラメータではなく、拡張されたパラメータセットを使用することができます。以下は、aws_s3.table_import_from_s3 関数の構文のバリエーションです。

- Amazon S3 ファイルを識別するために s3_info パラメータを使用する代わりに、bucket、file_path、および region パラメータの組み合わせを使用します。この関数の形式を使用する場合は、PostgreSQL DB インスタンスの IAM ロールを使用して、Amazon S3 へのアクセス権を付与します。

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- Amazon S3 アクセスを指定するために credentials パラメータを使用する代わりに、access_key、session_key、および session_token パラメータの組み合わせを使用します。

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
```

```
    file_path text,
    region text,
    access_key text,
    secret_key text,
    session_token text
)
```

代替パラメータ

バケット

ファイルを含む Amazon S3 バケットの名前を含むテキスト文字列。

file_path

ファイルのパスを含むAmazon S3ファイル名を含むテキスト文字列。

リージョン

ファイルの AWS リージョンの場所を識別するテキスト文字列。AWS リージョン 名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

access_key

インポートオペレーションに使用するアクセスキーを含むテキスト文字列。デフォルトは NULL です。

secret_key

インポートオペレーションに使用するシークレットキーを含むテキスト文字列。デフォルトは NULL です。

session_token

(オプション) インポートオペレーションに使用するセッションキーを含むテキスト文字列。デフォルトは NULL です。

aws_commons.create_s3_uri

Amazon S3 ファイル情報を保持するように、aws_commons._s3_uri_1 構造を作成します。aws_commons.create_s3_uri 関数の結果は、s3_info 関数の aws_s3.table_import_from_s3 パラメータで使用します。

構文

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

パラメータ

バケット

ファイルの Amazon S3 バケット名を含む必須のテキスト文字列。

file_path

ファイルのパスを含む Amazon S3 ファイル名を含む必須テキスト文字列。

リージョン

ファイルがある AWS リージョン を含む必須のテキスト文字列。AWS リージョン 名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

aws_commons.create_aws_credentials

`aws_commons._aws_credentials_1` 構造でアクセスキーとシークレットキーを設定します。`aws_commons.create_aws_credentials` 関数の結果は、`credentials` 関数の aws_s3.table_import_from_s3 パラメータで使用します。

構文

```
aws_commons.create_aws_credentials(
   access_key text,
   secret_key text,
   session_token text
)
```

パラメータ

access_key

Amazon S3 ファイルのインポートに使用するアクセスキーを含む必須のテキスト文字列。デフォルトは NULL です。

secret_key

Amazon S3 ファイルのインポートに使用するシークレットキーを含む必須のテキスト文字列。デフォルトは NULL です。

session_token

Amazon S3 ファイルのインポートに使用するセッショントークンを含む必須のテキスト文字列。デフォルトは NULL です。オプションの `session_token` を指定した場合は、一時的な認証情報を使用することができます。

DB インスタンス 間での PostgreSQL データベースの移行

Amazon RDS の PostgreSQL トランスポータブルデータベースを使用することで、2 つの DB インスタンス間で PostgreSQL データベースを移行できます。この手法により、異なる DB インスタンス間での大規模なデータベースの移行が大幅に高速化されます。このアプローチを使用するには、対象の DB インスタンスの両方で、PostgreSQL の同じ主要バージョンを実行している必要があります。

この機能を使用するには、移行元の DB インスタンスと移行先の DB インスタンスで、ともに `pg_transport` 拡張機能をインストールします。`pg_transport` 拡張は、最小限のプロセスでデータベースファイルを移動するための、物理的な移行メカニズムを提供します。このメカニズムにより、ダンプとロードによる従来のプロセスと比較してデータの移行が大幅に高速化され、ダウンタイムが最小限に抑えられます。

> **Note**
>
> PostgreSQL トランスポータブルデータベースは、RDS for PostgreSQL 11.5 以降、および RDS for PostgreSQL のバージョン 10.10 以降で利用可能です。

ある RDS for PostgreSQL DB インスタンスから別のインスタンスへの、PostgreSQL DB インスタンスの移行を行うには、最初に、「移行に向けた DB インスタンスの設定」での説明に従いながら、ソースインスタンスとデスティネーションインスタンスの設定を行います。その後、「PostgreSQL データベースの移行」で説明されている関数を使用してデータベースを移行します。

トピック

- データベースの移行中に何が起こるか
- PostgreSQL トランスポータブルデータベースの使用に関する制約事項
- PostgreSQL データベース移行の設定
- 移行元から移行先への PostgreSQL データベースの転送
- トランスポータブルデータベースの関数リファレンス
- トランスポータブルデータベースのパラメータリファレンス

データベースの移行中に何が起こるか

PostgreSQL のトランスポータブルデータベース機能は、移行先 DB インスタンスが移行元 DB インスタンスからデータベースをインポートするプルモデルを使用します。`transport.import_from_server` 関数を使うと、移行先 DB インスタンスで移行中のデータベースが作成されます。移行中、移行中のデータベースは、移行先 DB インスタンスではアクセスすることはできません。

移行スタートの際は、移行元データベースのすべての現在のセッションが終了します。移行元 DB インスタンスの移行元データベース以外のすべてのデータベースには移行による影響はありません。

移行元データベースは、特別な読み取り専用モードとなります。このモードの最中は、移行元データベースにアクセス可能で、読み取り専用のクエリを実行できます。ですが、書き込み可能なクエリやその他の種類のコマンドはブロックされます。移行された特定の移行元データベースのみがこれら制限による影響を受けます。

移行中、移行元 DB インスタンスは、ポイントインタイムの復元はできません。これは、移行がトランザクションではなく、変更を記録するための PostgreSQL ログ先行書き込みを使用しないからです。移行先 DB インスタンスの自動バックアップが有効になっていれば、移行後に、バックアップが自動で実行されます。ポイントインタイムの復元は、バックアップが終了した後に実行が可能になります。

移行に失敗した場合、`pg_transport` のエクステンションが、移行先/元 DB インスタンスで行ったすべての変更をやり直します。これには、移行先で一部のみ移行されたデータベースの削除も含まれます。失敗の内容によっては、移行元データベースで引き続き、書き込み可能クエリが拒否されます。これが発生した場合、以下のコマンドを使い、書き込み可能クエリを許可します。

```
ALTER DATABASE db-name SET default_transaction_read_only = false;
```

PostgreSQL トランスポータブルデータベースの使用に関する制約事項

トランスポータブルデータベースには以下の制限があります。

- リードレプリカ - リードレプリカまたはリードレプリカの親インスタンスでは、トランスポータブルデータベースを使用できません。
- サポートされていない列タイプ - このメソッドを使用して移行するすべてのデータベーステーブル内の `reg` データ型を使用することはできません。これらタイプは、システムカタログオブジェクト ID (OID) に依存し、移行中に変わることがあります。
- テーブルスペース　-すべてのソースデータベースオブジェクトは既定の `pg_default` テーブルスペースに既存しなければいけません。
- 互換性 - 移行先および移行元両方の DB インスタンスは、PostgreSQL の同じ主要バージョンを実行しなければなりません。
- 拡張機能 – 移行元 DB インスタンスには、`pg_transport` のみがインストールされています。
- ロールおよび ACL - 移行元データベースのアクセス権限および所有権情報は、移行先データベースには移行されません。すべてのデータベースオブジェクトは、移行するローカルの移行先ユーザーが作成および所有します。
- 同時移行数 – ワーカープロセスが適切に設定されていれば、単一の DB インスタンスで同時に最大 32 個の (インポートとエクスポートの両方を含む) 移行をサポートできます。
- RDS for PostgreSQL DB インスタンスのみ – PostgreSQL のトランスポータブルデータベースは、RDS for PostgreSQL DB インスタンスでのみサポートされます。オンプレミスのデータベースや Amazon EC2 で実行されているデータベースでは使用できません。

PostgreSQL データベース移行の設定

この作業を開始する前に、対象の RDS for PostgreSQL DB インスタンスが、以下の要件を満たしていることを確認してください。

- 移行先および移行元両方の RDS for PostgreSQL DB インスタンスは、PostgreSQL の同じバージョンを実行している必要があります。
- 移行先の DB は、移行する元である DB と同じ名前のデータベースを持つことはできません。
- 移行を実行するために使用するアカウントでは、移行元と移行先の両方の DB で、`rds_superuser` の権限が付与される必要があります。
- 移行元 DB インスタンスのセキュリティグループは、移行先 DB インスタンスからのインバウンドアクセスを許可する必要があります。この許可は、移行元と移行先の DB インスタンスが VPC 内

に存在する場合には、既に設定されている場合があります。セキュリティグループの詳細については、セキュリティグループによるアクセス制御 を参照してください。

データベースを移行元 DB インスタンスから移行先 DB インスタンスに移行する際には、各インスタンスに関連付けられた DB パラメータグループをいくつか変更する必要があります。つまり、移行元の DB インスタンスと移行先の DB インスタンスのそれぞれのために、カスタムの DB パラメータグループを作成する必要があります。

> **Note**
>
> DB インスタンスで、カスタム DB パラメータグループの使用が設定済みである場合は、以下の手順のステップ 2 から開始できます。

データベースを移行するためのカスタム DB グループパラメータを構成するには

以下の手順では、`rds_superuser` の権限を持つアカウントを使用します。

1. 移行元と移行先の DB インスタンスがデフォルトの DB パラメータグループを使用している場合は、そのインスタンス用として適切なバージョンの、カスタム DB パラメータグループを作成する必要があります。これにより、複数のパラメータの値を変更できるようになります。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
2. カスタム DB パラメータグループ内で、以下のパラメータの値を変更します。

   - `shared_preload_libraries` – ライブラリのリストに `pg_transport` を追加します。
   - `pg_transport.num_workers` – デフォルト値は 3 です。データベースの必要に応じて、この値を増減します。200 GB のデータベースでは、8 以下を推奨します。このパラメータのデフォルト値を増加させた場合は、`max_worker_processes` の値も増やす必要があることに注意してください。
   - `pg_transport.work_mem` – デフォルト値は、PostgreSQL のバージョンによって 128 MB または 256 MB のどちらかになります。デフォルト設定は、通常、特に変更する必要はありません。
   - `max_worker_processes` — このパラメータの値は、次の計算を使用して設定する必要があります。

     ```
     (3 * pg_transport.num_workers) + 9
     ```

この値は、トランスポートに関連するさまざまなバックグラウンドワーカープロセスを処理するために、接続先で必要です。max_worker_processes, の詳細については、PostgreSQL ドキュメントの「Resource Consumption」(資源の消費) を参照してください。

pg_transport パラメータの詳細については、「トランスポータブルデータベースのパラメータリファレンス」を参照してください。

3. 移行元と移行先の RDS for PostgreSQL DB インスタンスをともに再起動して、パラメータの設定を有効にします。
4. 移行元の RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=source-instance.111122223333.aws-region.rds.amazonaws.com --port=5432
 --username=postgres --password
```

5. DB インスタンスのパブリックスキーマから無関係な拡張機能を削除します。実際の移行オペレーション中に使用が許可されるのは、pg_transport 拡張機能のみです。
6. 次のように pg_transport 拡張機能をインストールします。

```
postgres=> CREATE EXTENSION pg_transport;
CREATE EXTENSION
```

7. 移行先の RDS for PostgreSQL DB インスタンスに接続します。無関係な拡張機能をすべて削除した上で、pg_transport 拡張機能をインストールします。

```
postgres=> CREATE EXTENSION pg_transport;
CREATE EXTENSION
```

移行元から移行先への PostgreSQL データベースの転送

PostgreSQL データベース移行の設定 で記載したプロセスを完了すると、移行をスタートすることが可能です。移行をスタートするには、移行先 DB インスタンスで transport.import_from_server 関数を実行します。次の構文では、関数に使用するパラメータを確認できます。

```
SELECT transport.import_from_server(
   'source-db-instance-endpoint',
    source-db-instance-port,
   'source-db-instance-user',
```

```
    'source-user-password',
    'source-database-name',
    'destination-user-password',
    false);
```

この例での false 値は、この処理がドライランではないことを関数に伝えます。移行の設定をテストするには、以下のように、関数の呼び出し時に dry_run オプションで true を指定します。

```
postgres=> SELECT transport.import_from_server(
    'docs-lab-source-db.666666666666aws-region.rds.amazonaws.com', 5432,
    'postgres', '********', 'labdb', '******', true);
INFO:  Starting dry-run of import of database "labdb".
INFO:  Created connections to remote database        (took 0.03 seconds).
INFO:  Checked remote cluster compatibility          (took 0.05 seconds).
INFO:  Dry-run complete                           (took 0.08 seconds total).
 import_from_server
--------------------

(1 row)
```

pg_transport.timing パラメータがデフォルト値の true に設定されているため、INFO 行が出力されます。次に示すように、dry_run に false を設定してコマンドを実行し、データベースを移行元から移行先にインポートします。

```
INFO:  Starting import of database "labdb".
INFO:  Created connections to remote database        (took 0.02 seconds).
INFO:  Marked remote database as read only           (took 0.13 seconds).
INFO:  Checked remote cluster compatibility          (took 0.03 seconds).
INFO:  Signaled creation of PITR blackout window     (took 2.01 seconds).
INFO:  Applied remote database schema pre-data       (took 0.50 seconds).
INFO:  Created connections to local cluster          (took 0.01 seconds).
INFO:  Locked down destination database              (took 0.00 seconds).
INFO:  Completed transfer of database files          (took 0.24 seconds).
INFO:  Completed clean up                            (took 1.02 seconds).
INFO:  Physical transport complete                (took 3.97 seconds total).
import_from_server
--------------------
(1 row)
```

この関数には、データベースユーザーパスワードを入力する必要があります。よって、移行完了後は、使用したユーザーロールのパスワードを変更することをお勧めします。または、SQL のバイン

ド可変を使用するとユーザーロールを一時的に作成することができます。このようなテンポラリロールを移行に使ったら、破棄することが可能です。

移行が成功しなかった場合、次のようなエラーメッセージが表示されることがあります。

```
pg_transport.num_workers=8 25% of files transported failed to download file data
```

「failed to download file data (ファイルデータのダウンロードに失敗しました)」というエラーメッセージは、データベースのサイズに対してワーカープロセスの数が正しく設定されていないことを示します。pg_transport.num_workers に設定した値の増減が必要な場合があります。失敗が発生するたびに、処理の完了率がレポートされるため、変更の影響度合いを確認できます。例えば、あるケースで設定を 8 から 4 に変更した場合、次のような結果になります。

```
pg_transport.num_workers=4 75% of files transported failed to download file data
```

max_worker_processes パラメーターは、移行のプロセス中にも考慮されることにご留意ください。つまり、データベースを正常に移行するためには、pg_transport.num_workers と max_worker_processes の両方で変更が必要な場合があります。pg_transport.num_workers に 2 を設定することで、最終的にこの例は正しく機能します。

```
pg_transport.num_workers=2 100% of files transported
```

transport.import_from_server 関数とそのパラメータの詳細については、「トランスポータブルデータベースの関数リファレンス」を参照してください。

トランスポータブルデータベースの関数リファレンス

transport.import_from_server 関数は、PostgreSQL データベースを移行元 DB インスタンスから移行先 DB インスタンスにインポートします。これは、物理的なデータベース接続移行メカニズムを使って実行されます。

この関数は、移行元と移行先の DB インスタンスが同じバージョンであり、移行のための互換性があることを、移行の開始前に確認します。また、移行先の DB インスタンスに、移行元のサイズに見合うの十分な領域があることも確認します。

[Syntax] (構文)

```
transport.import_from_server(
   host text,
   port int,
```

```
   username text,
   password text,
   database text,
   local_password text,
   dry_run bool
)
```

戻り値

なし。

パラメータ

transport.import_from_server 関数パラメータの説明に関しては、以下のテーブルをご参照ください。

Parameter	説明
host	移行元 DB インスタンスのエンドポイント。
port	整数は、移行元 DB インスタンスを表しています。 PostgreSQL DB インスタンスは、通常ポート 5432 を使います。
username	移行元 DB インスタンスのユーザー。このユーザーは、rds_superuser ロールのメンバーでなければなりません。
password	移行元 DB インスタンスのユーザーパスワード。
database	移行する移行元 DB インスタンスのデータベース名。
local_password	移行先 DB インスタンスの現在のユーザーのローカルパスワード。このユーザーは、rds_superuser ロールのメンバーでなければなりません。
dry_run	リハーサルを実施するかどうかを判断する任意のブール値。デフォルトは、false で、移行を実行することを意味します。 実際に移行せずに、移行元と移行先 DB インスタンスの互換性を確認するには、dry_run を true に設定します。

例

例については、「移行元から移行先への PostgreSQL データベースの転送」を参照してください。

トランスポータブルデータベースのパラメータリファレンス

複数のパラメータにより、pg_transport 拡張機能の動作が制御されます。以下で、これらのパラメータの説明を参照してください。

pg_transport.num_workers

移行プロセスに使用するワーカー数。デフォルトは 3 です。有効な値は、1-32 です。大規模なデータベースを移行する場合でも、通常必要なワーカー数は 8 未満です。移行中には、移行先 DB インスタンスの設定が、移行先および移行元の両方の DB インスタンスで使用されます。

pg_transport.timing

移行中にタイミング情報を報告するかどうかを指定します。デフォルトは true で、タイミング情報が報告されることを意味します。進行状況を監視できるようにするため、このパラメータは true に設定しておくことをお勧めします。出力例については、「移行元から移行先への PostgreSQL データベースの転送」を参照してください。

pg_transport.work_mem

メモリの最大容量を各ワーカーに配分する。PostgreSQL のバージョンに応じて、この設定のデフォルトは、131,072 キロバイト (KB) または 262,144 KB (256 MB) のどちらかになります。最小値は 64 メガバイト (65,536 KB) です。2 進法ベースの 2 ユニット (1 KB = 1,024 バイト) なので、有効値は、キロバイト (KB) で表記されます。

移行は、このパラメータで指定されたメモリより少ないメモリを使う場合があります。移行するデータベースが大規模な場合でも、通常必要なメモリは、ワーカー当たり 256 MB (262,144 KB) 未満です。

RDS for PostgreSQL DB インスタンスから Amazon S3 へのデータのエクスポート

RDS for PostgreSQL DB インスタンスからデータをクエリし、Amazon S3 バケットに保存されているファイルに直接エクスポートできます。これを行うには、 RDS for PostgreSQL aws_s3拡張機能を最初にインストールします。このエクステンションでは、Amazon S3 へのクエリの結果のエクスポートに使用する関数が利用できます。次に、拡張機能のインストール方法と Amazon S3 へのデータのエクスポート方法を説明します。

> **Note**
>
> クロスアカウントでの Amazon S3 はサポートされていません。

現在利用可能な RDS for PostgreSQL のバージョンでは、データの Amazon Simple Storage Service へのエクスポートがサポートされています。詳細なバージョン情報については、「Amazon RDS for PostgreSQL リリースノート」の「Amazon RDS for PostgreSQL の更新」を参照してください。

エクスポートにバケットを設定していない場合は、Amazon Simple Storage Service ユーザーガイドで次のトピックを参照してください。

- Amazon S3 のセットアップ
- バケットの作成

デフォルトでは、RDS for PostgreSQL から Amazon S3 にエクスポートされたデータは、AWS マネージドキー によるサーバー側の暗号化が使用されます。バケット暗号化を使用している場合は、Amazon S3 バケットは AWS Key Management Service (AWS KMS) キー (SSE-KMS) で暗号化されている必要があります。現在、Amazon S3 マネージドキー (SSE-S3) で暗号化されたバケットはサポートされていません。

> **Note**
>
> AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用して、DB スナップショットのデータを Amazon S3 に保存できます。詳しくは、「Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。

トピック

- aws_s3 拡張機能のインストール
- Amazon S3 へのデータのエクスポートの概要
- エクスポート先の Amazon S3 ファイルパスを指定する
- Amazon S3 バケットへのアクセスを設定する
- aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート
- 関数リファレンス
- Amazon S3 へのアクセスのトラブルシューティング

aws_s3 拡張機能のインストール

RDS for PostgreSQL DB インスタンスで Amazon Simple Storage Service を使用する前に、aws_s3 拡張機能をインストールする必要があります。この拡張機能には、 RDS for PostgreSQL DB インスタンスから Amazon S3 バケットへデータをエクスポートするための関数も含まれています。また、Amazon S3 からデータをインポートするための関数も含まれます。詳しくは、「Amazon S3 から RDS for PostgreSQL DB インスタンスにデータをインポートする」を参照してください。aws_s3 拡張機能は aws_commons 拡張機能の一部のヘルパー関数に依存しており、必要に応じて自動的にインストールされます。

aws_s3 拡張機能をインストールするには

1. rds_superuser 権限があるユーザーとして、psql (または pgAdmin) を使用して RDS for PostgreSQL DB インスタンスに接続します。設定プロセス中にデフォルトの名前を保持している場合は、postgres として接続します。

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 拡張機能をインストールするには、次のコマンドを実行します。

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. 拡張機能がインストールされていることを確認するには、psql \dx メタコマンドを使用します。

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                Description
-------------+---------+------------+-------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Amazon S3 からデータをインポートし、データを Amazon S3 にエクスポートするための関数が使用できるようになりました。

ご使用の RDS for PostgreSQL バージョンで、Amazon S3 へのエクスポートがサポートされていることを確認します

`describe-db-engine-versions` コマンドを使用して、RDS for PostgreSQL バージョンが Amazon S3 へのエクスポートをサポートしていることを確認できます。次に、バージョン 10.14 のサポートを確認する例を示します。

```
aws rds describe-db-engine-versions --region us-east-1
--engine postgres --engine-version 10.14 | grep s3Export
```

出力に `"s3Export"` の文字列が含まれている場合 、エンジンは Amazon S3 エクスポートをサポートします。それ以外の場合、エンジンはエクスポートをサポートしません。

Amazon S3 へのデータのエクスポートの概要

RDS for PostgreSQL データベースに格納されたデータを Amazon S3 バケットにエクスポートするには、以下の手順に従います。

RDS for PostgreSQL データを S3 にエクスポートするには

1. データのエクスポートに使用する Amazon S3 ファイルパスを指定します。このプロセスの詳細については、「エクスポート先の Amazon S3 ファイルパスを指定する」を参照してください。
2. Amazon S3 バケットへのアクセス許可を提供します。

Amazon S3 ファイルにデータをエクスポートするには、RDS for PostgreSQL DB インスタンスに、エクスポートの際に保存に使用される Amazon S3 バケットへのアクセス許可を付与する必要があります。これには、次のステップが含まれます。

1. エクスポート先の Amazon S3 バケットへのアクセスを提供する IAM ポリシーを作成します。
2. IAM ロールを作成します。
3. 作成したポリシーを、作成したロールにアタッチします。
4. この IAM ロールを DB インスタンスに追加します。

このプロセスの詳細については、「Amazon S3 バケットへのアクセスを設定する」を参照してください。

3. データを取得するためのデータベースクエリを識別します。`aws_s3.query_export_to_s3` 関数を呼び出して、クエリデータをエクスポートします。

   前述の準備タスクを完了したら、aws_s3.query_export_to_s3 関数を使用してクエリ結果を Amazon S3 にエクスポートします。このプロセスの詳細については、「aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート」を参照してください。

エクスポート先の Amazon S3 ファイルパスを指定する

次の情報を指定して、Amazon S3 データのエクスポート先となる場所を指定します。

- バケット名 - バケットは、Amazon S3 オブジェクトまたはファイルのコンテナです。

  Amazon S3 を使用したデータの保存の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの作成」と「オブジェクトの使用」を参照してください。
- ファイルパス - ファイルパスは、Amazon S3 バケット内のエクスポートが格納される場所を識別します。ファイルパスは、次のもので構成されます。
  - 仮想フォルダパスを識別するオプションのパスプレフィックス。
  - 保存する 1 つ以上のファイルを識別するファイルプレフィックス。より大きなエクスポートは複数のファイルに格納され、それぞれの最大サイズは約 6 GB です。追加のファイル名には、同じファイルプレフィックスが付いていますが、末尾に `_part`*XX* が付加されます。*XX* は、2、3 などを表します。

例えば、exports フォルダとファイルプレフィックスを持つ query-1-export ファイルパスは /exports/query-1-export です。

- AWS リージョン (オプション) - Amazon S3 バケットがある AWS リージョン。AWS リージョンの値を指定しない場合、Amazon RDS は、エクスポートする DB インスタンスと同じ AWS リージョンの Amazon S3 にファイルを保存します。

> **Note**
>
> 現在、AWS リージョンは、エクスポートする DB インスタンスのリージョンと同じである必要があります。

AWS リージョン名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

エクスポートの保存先に関する Amazon S3 ファイル情報を保持するには、aws_commons.create_s3_uri 関数を使用して、次のように aws_commons._s3_uri_1 複合構造を作成します。

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

その後、この s3_uri_1 値を aws_s3.query_export_to_s3 関数の呼び出しでパラメータとして指定します。例については、「aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート」を参照してください。

Amazon S3 バケットへのアクセスを設定する

データを Amazon S3 にエクスポートするには、PostgreSQL DB インスタンスに、ファイルが入る Amazon S3 バケットに対するアクセス許可を付与します。

これには、以下の手順を使用します。

IAM ロールを介して PostgreSQLDB のインスタンスに Amazon S3 へのアクセスを許可するには

1. IAM ポリシーを作成します。

   このポリシーは、PostgreSQL DB インスタンスに、Amazon S3 のバケットとオブジェクトに対するアクセス許可を付与します。

   このポリシーの作成の一環として、次のステップを実行します。

   a. ポリシーに、PostgreSQL DB インスタンスから Amazon S3 バケットへのファイル転送を許可するための以下の必須アクションを含めます。

      - `s3:PutObject`
      - `s3:AbortMultipartUpload`

   b. Amazon S3 バケットとバケット内のオブジェクトを識別する Amazon リソースネーム (ARN) を含めます。Amazon S3 アクセス用の ARN 形式は `arn:aws:s3:::`*`amzn-s3-demo-bucket`*`/*` です。

   Amazon RDS for PostgreSQL の IAM ポリシーの作成の詳細については、IAM データベースアクセス用の IAM ポリシーの作成と使用 を参照してください。IAM ユーザーガイドの「チュートリアル: はじめてのカスタマー管理ポリシーの作成とアタッチ」も参照してください。

   以下の AWS CLI コマンドでは、これらのオプションを指定して、`rds-s3-export-policy` という名前の IAM ポリシーを作成します。*`amzn-s3-demo-bucket`* という名前のバケットへのアクセスを許可します。

   > **Warning**
   >
   > 特定のバケットにアクセスするようにエンドポイントポリシーが設定されているプライベート VPC 内にデータベースをセットアップすることをお勧めします。詳細については、Amazon VPC ユーザーガイドの「Amazon S3 のエンドポイントポリシーの使用」を参照してください。
   > すべてのリソースへのアクセスを持つポリシーを作成しないことを強くお勧めします。このアクセスは、データセキュリティにとって脅威になる可能性があります。`S3:PutObject` を使用してすべてのリソースへのアクセスを `"Resource":"*"` に許可するポリシーを作成すると、エクスポート権限を持つユーザーはアカウント内のす

べてのバケットにデータをエクスポートできます。さらに、ユーザーは AWS リージョン内のパブリックに書き込み可能なバケットにデータをエクスポートできます。

ポリシーを作成したら、そのポリシーの Amazon リソースネーム (ARN) を書き留めます。ポリシーを IAM ロールにアタッチする場合、後続のステップで ARN が必要です。

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

2. IAM ロールを作成します。

これを行うと、Amazon RDS がユーザーに代わってこの IAM ロールを引き受け、Amazon S3 バケットにアクセスできます。詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、サービスに付与するリソースへのアクセス許可を制限することをお勧めします。これは、混乱した使節の問題に対する最も効果的な保護方法です。

グローバル条件コンテキストキーの両方を使用し、aws:SourceArn の値にアカウント ID が含まれている場合、同じポリシーステートメントで使用する場合は、aws:SourceArn の値と aws:SourceAccount の値のアカウントでは同じアカウント ID を使用する必要があります。

- 単一リソースに対するクロスサービスアクセスが必要な場合は aws:SourceArn を使用します。
- そのアカウント内の任意のリソースをクロスサービス使用に関連付けることを許可する場合、aws:SourceAccountを使用します。

ポリシーでは、必ずリソースの完全な ARN を持つ aws:SourceArn グローバル条件コンテキストキーを使用してください。以下の例は、AWS CLI コマンドを使用して、rds-s3-export-role という名前のロールを作成する方法を示しています。

Example

Linux、macOS、Unix の場合:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
                }
             }
       }
     ]
   }'
```

Windows の場合:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
```

```
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
               "aws:SourceAccount": "111122223333",
               "aws:SourceArn": "arn:aws:rds:us-east-1:111122223333:db:dbname"
               }
            }
      }
    ]
  }'
```

3. 作成した IAM ポリシーを、作成した IAM ロールにアタッチします。

   次の AWS CLI コマンドは、先ほど作成したポリシーを `rds-s3-export-role.` という名前のロールにアタッチします。*`your-policy-arn`* を前のステップでメモしたポリシー ARN に置き換えます。

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

4. DB インスタンスに IAM ロールを追加します。これを行うには、以下で説明するように、AWS マネジメントコンソール または AWS CLI を使用します。

コンソール

コンソールを使用して PostgreSQL DB インスタンスの IAM ロールを追加するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. 詳細を表示するには、PostgreSQL DB インスタンスの名前を選択します。
3. [接続とセキュリティ] タブの [IAM ロールの管理] セクションで、[このインスタンスに IAM ロールを追加] で追加するロールを選択します。
4. [Feature] で、[s3Export] を選択します。
5. [Add role] を選択します。

AWS CLI

CLI を使用して PostgreSQL DB インスタンスの IAM ロールを追加するには

- 次のコマンドを使用して、`my-db-instance` という名前の PostgreSQL DB インスタンスにロールを追加します。*`your-role-arn`* を、以前のステップで書き留めたロール ARN に置き換えます。`s3Export` オプションの値に `--feature-name` を使用します。

  Example

  Linux、macOS、Unix の場合:

  ```
  aws rds add-role-to-db-instance \
     --db-instance-identifier my-db-instance \
     --feature-name s3Export \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Windows の場合:

  ```
  aws rds add-role-to-db-instance ^
     --db-instance-identifier my-db-instance ^
     --feature-name s3Export ^
     --role-arn your-role-arn ^
     --region your-region
  ```

aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート

aws_s3.query_export_to_s3 関数を呼び出して、PostgreSQL データを Amazon S3 にエクスポートします。

トピック

- 前提条件
- aws_s3.query_export_to_s3 の呼び出し
- カスタム区切り文字を使用する CSV ファイルへのエクスポート
- エンコードを使用したバイナリファイルへのエクスポート

前提条件

aws_s3.query_export_to_s3 関数を使用する前に、以下の前提条件を満たしていることを確認してください。

- 「Amazon S3 へのデータのエクスポートの概要」の説明に従って、必要な PostgreSQL エクステンションをインストールします。
- 「エクスポート先の Amazon S3 ファイルパスを指定する。」の説明に従って、データの Amazon S3 のエクスポート先を決定します。
- 「Amazon S3 バケットへのアクセスを設定する」の説明にとおり、DB インスタンスが Amazon S3 へのエクスポートアクセス権があることを確認します。

次の例では、sample_table というデータベーステーブルを使用しています。次の例では、データを *amzn-s3-demo-bucket* というバケットにエクスポートします。サンプルのテーブルとデータは、psql で次の SQL ステートメントを使用して作成されます。

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

aws_s3.query_export_to_s3 の呼び出し

次に、 aws_s3.query_export_to_s3 関数を呼び出す基本的な方法を示します。

これらの例では、可変 s3_uri_1 を使用して、Amazon S3 ファイルを識別する情報を含む構造を指定しています。aws_commons.create_s3_uri 関数を使用して構造を作成します。

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

以下の 2 つの aws_s3.query_export_to_s3 関数呼び出しのパラメータは異なりますが、これらの例の結果は同じです。sample_table テーブルのすべての行が *amzn-s3-demo-bucket* というバケットにエクスポートされます。

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');
```

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

パラメータの説明は次のとおりです。

- `'SELECT * FROM sample_table'` - 初期のパラメータは、SQL クエリを含む必須のテキスト文字列です。PostgreSQL エンジンはこのクエリを実行します。クエリの結果は、他のパラメータで指定された S3 バケットにコピーされます。
- `:'s3_uri_1'` - このパラメータは、Amazon S3 ファイルを識別する構造です。この例では、可変を使用して、前に作成した構造を指定します。代わりに、以下のように `aws_commons.create_s3_uri` 関数呼び出し内にインラインで `aws_s3.query_export_to_s3` 関数呼び出しを含めることで、同じ構造を作成できます。

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  );
  ```

- `options :='format text'` - `options` パラメータは、PostgreSQL COPY 引数を含むオプションのテキスト文字列です。このコピープロセスでは、PostgreSQL COPY コマンドの引数と形式を使用します。

指定したファイルが Amazon S3 バケットに存在しない場合は、作成されます。このファイルが存在している場合は、上書きされます。Amazon S3 でエクスポートされたデータにアクセスするための構文は次のとおりです。

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

より大きなエクスポートは複数のファイルに格納され、それぞれの最大サイズは約 6 GB です。追加のファイル名には、同じファイルプレフィックスが付いていますが、末尾に _part*XX* が付加されます。*XX* は、2、3 などを表します。例えば、次のようにデータファイルを格納するパスを指定するとします。

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

エクスポートで 3 つのデータファイルを作成する必要がある場合、Amazon S3 バケットには次のデータファイルが含まれます。

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

この関数の完全なリファレンスと、それを呼び出すその他の方法については、「aws_s3.query_export_to_s3」を参照してください。Amazon S3 でファイルにアクセスする方法の詳細については、Amazon Simple Storage Service ユーザーガイドの「View an object」を参照してください。

カスタム区切り文字を使用する CSV ファイルへのエクスポート

次の例は、aws_s3.query_export_to_s3 関数を呼び出して、カスタム区切り文字を使用するファイルにデータをエクスポートする方法を示しています。この例では、PostgreSQL COPY コマンドの引数を使用して、カンマ区切り値 (CSV) 形式とコロン (:) 区切り文字を指定します。

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

エンコードを使用したバイナリファイルへのエクスポート

次の例は、aws_s3.query_export_to_s3 関数を呼び出して、Windows-1253 エンコーディングのバイナリファイルにデータをエクスポートする方法を示しています。

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

関数リファレンス

関数

- aws_s3.query_export_to_s3
- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

PostgreSQL クエリ結果を Amazon S3 バケットにエクスポートします。aws_s3 エクステンションには、aws_s3.query_export_to_s3 関数が含まれます。

2 つの必須パラメータは、query および s3_info です。これらは、エクスポートするクエリを定義し、エクスポート先の Amazon S3 バケットを特定します。options と呼ばれるオプションのパラメータは、さまざまなエクスポートパラメータを定義するために用意されています。aws_s3.query_export_to_s3 関数の使用例については、「aws_s3.query_export_to_s3 関数を使用したクエリデータのエクスポート」を参照してください。

構文

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

入力パラメータ

query()

PostgreSQL エンジンが実行する SQL クエリを含む必須のテキスト文字列。このクエリ結果は、s3_info パラメータで指定された S3 バケットにコピーされます。

s3_info

S3 オブジェクトに関する以下の情報を含む aws_commons._s3_uri_1 複合型。

- bucket - ファイルを格納する Amazon S3 バケットの名前。
- file_path - Amazon S3 ファイル名とパス
- region - バケットが存在する AWS リージョン。AWS リージョン名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

  現在、この値は、エクスポートする DB DB インスタンスの AWS リージョンと同じリージョンである必要があります。デフォルトは、エクスポートする DB DB インスタンスの AWS リージョンです。

aws_commons._s3_uri_1 複合構造を作成するには、aws_commons.create_s3_uri 関数を参照してください。

options

PostgreSQL COPY コマンドの引数を含むオプションのテキスト文字列。これらの引数は、エクスポート時のデータのコピー方法を指定します。詳細については、「PostgreSQL COPY ドキュメント」を参照してください。

kms_key text

データのエクスポート先となる S3 バケットのカスタマーマネージド KMS キーを含む任意のテキスト文字列。

代替入力パラメータ

テストしやすいように、s3_info パラメータではなく、拡張されたパラメータセットを使用することができます。以下は、aws_s3.query_export_to_s3 関数の構文のバリエーションです。

Amazon S3 ファイルを識別するために s3_info パラメータを使用する代わりに、bucket、file_path、および region パラメータの組み合わせを使用します。

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query()

PostgreSQL エンジンが実行する SQL クエリを含む必須のテキスト文字列。このクエリ結果は、s3_info パラメータで指定された S3 バケットにコピーされます。

バケット

ファイルを含む Amazon S3 バケットの名前を含む必須テキスト文字列。

file_path

ファイルのパスを含む Amazon S3 ファイル名を含む必須テキスト文字列。

リージョン

バケットが存在する AWS リージョンを含むオプションのテキスト文字列。AWS リージョン名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

現在、この値は、エクスポートする DB DB インスタンスの AWS リージョンと同じリージョンである必要があります。デフォルトは、エクスポートする DB DB インスタンスの AWS リージョンです。

options

PostgreSQL COPY コマンドの引数を含むオプションのテキスト文字列。これらの引数は、エクスポート時のデータのコピー方法を指定します。詳細については、「PostgreSQL COPY ドキュメント」を参照してください。

kms_key text

データのエクスポート先となる S3 バケットのカスタマーマネージド KMS キーを含む任意のテキスト文字列。

出力パラメータ

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

指定されたクエリで Amazon S3 に正常にアップロードされたテーブルローの数。

files_uploaded

Amazon S3 にアップロードされたファイルの数。ファイルは、約 6 GB のサイズで作成されます。作成される各追加ファイルは、名前に _part*XX* が付加されています。*XX* は、必要に応じて 2、3 などを表します。

bytes_uploaded

Amazon S3 にアップロードされた合計バイト数。

例

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Amazon S3 ファイル情報を保持するように、aws_commons._s3_uri_1 構造を作成します。aws_commons.create_s3_uri 関数の結果は、s3_info 関数の aws_s3.query_export_to_s3 パラメータで使用します。aws_commons.create_s3_uri 関数の使用例については、「エクスポート先の Amazon S3 ファイルパスを指定する」を参照してください。

Syntax

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

入力パラメータ

バケット

ファイルの Amazon S3 バケット名を含む必須のテキスト文字列。

file_path

ファイルのパスを含む Amazon S3 ファイル名を含む必須テキスト文字列。

リージョン

ファイルがある AWS リージョンを含む必須のテキスト文字列。AWS リージョン名と関連する値のリストについては、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。

Amazon S3 へのアクセスのトラブルシューティング

Amazon S3 へのデータのエクスポート試行時に接続の問題が発生した場合は、まず DB インスタンスに関連付けられた VPC セキュリティグループのアウトバウンドアクセスルールがネットワーク接続を許可していることを確認します。具体的には、DB インスタンスにポート 443 および任意の IPv4 アドレス (0.0.0.0/0) への TCP トラフィックの送信を許可するルールをセキュリティグループに作成します。詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

推奨事項については、以下も参照してください。

- Amazon RDS のアイデンティティおよびアクセスのトラブルシューティング
- Amazon Simple Storage Service ユーザーガイド の「Troubleshooting Amazon S3」
- IAM ユーザーガイド の Amazon S3 のトラブルシューティングと IAM

RDS for PostgreSQL DB インスタンスから AWS Lambda 関数を呼び出す

AWS Lambda は、サーバーのプロビジョニングや管理を行わなくてもコードの実行が可能な、イベント駆動型のコンピューティングサービスです。この機能は、RDS for PostgreSQL を含む多くの AWS サービスで利用可能です。例えば、データベースからのイベント通知の処理や、新しいファイルが Amazon S3 にアップロードされるたびに行うファイルからのデータロードのために、Lambda を使用することができます。詳細については、「AWS Lambda デベロッパーガイドの「AWS Lambda とは」を参照してください。

> **Note**
>
> RDS for PostgreSQL では、以下のバージョンで AWS Lambda 関数の呼び出しがサポートされています。
>
> - すべての PostgreSQL 18 バージョン
> - すべての PostgreSQL 17 バージョン
> - すべての PostgreSQL 16 バージョン
> - すべての PostgreSQL 15 バージョン
> - PostgreSQL 14.1 以降のマイナーバージョン
> - PostgreSQL 13.2 以降のマイナーバージョン
> - PostgreSQL 12.6 以降のマイナーバージョン

RDS for PostgreSQL で Lambda 関数を操作するためのセットアップは、AWS Lambda、IAM、VPC、および RDS for PostgreSQL DB インスタンスが関係する複数ステップのプロセスとなります。以下に、必要なステップの概要を示します。

Lambda 関数の詳細については、「AWS Lambda デベロッパーガイド」の「Lambda の開始方法」および「AWS Lambda の基礎」を参照してください。

トピック

- ステップ 1: RDS for PostgreSQL DB インスタンスで、AWS Lambda へのアウトバウンド接続を設定する。
- ステップ 2: RDS for PostgreSQL DB インスタンスおよび AWS Lambda のために IAM を設定する
- ステップ 3: RDS for PostgreSQL DB インスタンス用に aws_lambda 拡張機能をインストールする

- ステップ 4: RDS for PostgreSQL DB インスタンスで Lambda のヘルパー関数を使用する (オプション)
- ステップ 5: RDS for PostgreSQL DB インスタンスから Lambda 関数を呼びだす
- ステップ 6: Lambda 関数を呼び出すその他のユーザー許可を付与する
- 例: RDS for PostgreSQL DB インスタンスから Lambda 関数を呼びだす
- Lambda 関数のエラーメッセージ
- AWS Lambda 関数とパラメータのリファレンス

ステップ 1: RDS for PostgreSQL DB インスタンスで、AWS Lambda へのアウトバウンド接続を設定する。

Lambda 関数は、常に AWS Lambda サービスが所有する Amazon VPC 内で実行されます。Lambda はこの VPC にネットワークアクセスとセキュリティルールを適用し、この VPC を自動的にモニタリングおよび維持します。RDS for PostgreSQL DB インスタンスは、Lambda サービスの VPC にネットワークトラフィックを送信します。このための構成方法は、DB インスタンスが、パブリックであるかプライベートであるかにより異なります。

- パブリック RDS for PostgreSQL DB インスタンス — VPC のパブリックサブネット内に置かれた DB インスタンスで、「PubliclyAccessible」プロパティに `true` が設定されている場合、そのインスタンスはパブリックです。このプロパティの値は、AWS CLI コマンド describe-db-instances を使用して確認できます。または、AWS マネジメントコンソール を使用して [Connectivity & security] (接続とセキュリティ) タブを開き、[Publicly accessible] (パブリックアクセス可能) が「はい」となっているかを確認します。インスタンスが VPC のパブリックサブネット内に置かれていることを確認するには、AWS マネジメントコンソール または AWS CLI を使用します。

  Lambda へのアクセスを設定するには、AWS マネジメントコンソール または AWS CLI を使用して、VPC のセキュリティグループでアウトバウンドルールを作成します。アウトバウンドルールでは、TCP がポート 443 を使用して任意の IPv4 アドレス (0.0.0.0/0) にパケットを送信するように定義しています。
- プライベートRDS for PostgreSQL DB インスタンス — この例では、インスタンスの「PubliclyAccessible」プロパティが `false` に指定されているか、インスタンスがプライベートサブネット内に置かれています。インスタンスが Lambda で動作できるようにするには、ネットワークアドレス変換 (NAT) ゲートウェイを使用します。詳細については、「NAT ゲートウェイ」を参照してください。または、VPC で Lambda の VPC エンドポイントを設定できます。詳細については、Amazon VPC ユーザーガイド の「VPC エンドポイント」を参照してください。このエ

ンドポイントは、RDS for PostgreSQL DB インスタンスが Lambda 関数に対して発行した、呼び出しに対して応答します。VPC エンドポイントは、独自のプライベートな DNS 解決を使用します。`rds.custom_dns_resolution` の値がデフォルトの 0 (有効化されていない) から 1 に変更されない限り、RDS for PostgreSQL は、Lambda VPC エンドポイントを使用することはできません。そのためには、次の操作を行います。

- カスタム DB パラメータグループを作成します。
- `rds.custom_dns_resolution` パラメータの値を、デフォルトの `0` から `1` に変更します。
- カスタムの DB パラメータグループを使用するように DB インスタンスを変更します。
- 修正されたパラメータを反映させるために、インスタンスを再起動します。

ご使用の VPC は、ネットワークレベルで AWS Lambda VPC とやり取りできるようになります。次に、IAM を使用してアクセス権限を設定します。

ステップ 2: RDS for PostgreSQL DB インスタンスおよび AWS Lambda のために IAM を設定する

RDS for PostgreSQL DB インスタンスからの Lambda 関数の呼び出しには、特定の権限が必要です。必要な権限を設定するには、Lambda 関数の呼び出しを許可する IAM ポリシーを作成し、そのポリシーをロールに割り当てた上で、そのロールを DB インスタンスに適用することをお勧めします。このアプローチでは、指定された Lambda 関数をユーザーに代わって呼び出すための権限を、DB インスタンスに対し付与します。以下のステップで、AWS CLI を使用してこれを行う方法を示します。

Amazon RDS インスタンスで Lambda を使用するために IAM のアクセス許可を設定するには

1. AWS CLI コマンド create-policy を実行して、指定された Lambda 関数を、RDS for PostgreSQL DB インスタンスが呼び出すことを許可する、IAM ポリシーを作成します。(ステートメント ID (Sid) は、ポリシーステートメントのオプションの記述であり、使用には影響しません。) このポリシーは、DB インスタンスに対し、指定された Lambda 関数を呼び出すための最小限のアクセス許可を付与します。

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
```

```
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

または、任意の Lambda 関数の呼び出しを許可する、事前定義済みの AWSLambdaRole ポリシーを使用することもできます。詳細については、「Lambda のアイデンティティベースの IAM ポリシー」を参照してください。

2. AWS CLI コマンド create-role を使用して、実行時にポリシーが引き受けることができる IAM ロールを作成します。

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. AWS CLI コマンド attach-role-policy を使用して、このポリシーをロールに適用します。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. AWS CLI コマンド add-role-to-db-instance を使用して、このロールを RDS for PostgreSQL DB インスタンスに適用します。この最後のステップにより、DB インスタンスのデータベースユーザーに対し、Lambda 関数の呼び出しを許可します。

```
aws rds add-role-to-db-instance \
       --db-instance-identifier my-instance-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

VPC と IAM の設定が完了したので、ここで aws_lambda 拡張をインストールできます。(拡張機能は任意のタイミングでインストールできますが、先に VPC サポートと IAM 権限を適切に設定する必要があります。aws_lambda 拡張機能は、RDS for PostgreSQL DB インスタンスの機能に対し何も追加しません。)

ステップ 3: RDS for PostgreSQL DB インスタンス用に **aws_lambda** 拡張機能をインストールする

RDS for PostgreSQL DB インスタンスで AWS Lambda を使用し、RDS for PostgreSQL DB インスタンスに対し aws_lambda PostgreSQL 拡張機能を追加します。この拡張機能は、RDS for PostgreSQL DB インスタンスに対し、PostgreSQL からの Lambda 関数呼び出し機能を追加します。

RDS for PostgreSQL DB インスタンスに **aws_lambda** 拡張機能をインストールするには

PostgreSQL の psql コマンドライン、または pgAdmin ツールを使用して、RDS for PostgreSQL DB インスタンスに接続します。

1. RDS for PostgreSQL DB インスタンスに、rds_superuser 権限を持つユーザーとして接続します。例では、デフォルトの postgres ユーザが示されています。

   ```
   psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. aws_lambda 拡張機能をインストールします。aws_commons 拡張機能も必要です。これは、aws_lambda や、他の多数の PostgreSQL向け Aurora 拡張機能にヘルパー関数を提供します。この拡張機能が、RDS for PostgreSQL DB インスタンス上で見つからない場合は、次のように aws_lambda を使用してインストールされています。

   ```
   CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
   NOTICE:  installing required extension "aws_commons"
   CREATE EXTENSION
   ```

aws_lambda 拡張機能は、 DB インスタンスにインストールされています。この段階で、Lambda 関数を呼び出すための、使いやすい構造を作成することが可能です。

ステップ 4: RDS for PostgreSQL DB インスタンスで Lambda のヘルパー関数を使用する (オプション)

aws_commons 拡張機能のヘルパー関数を使用すると、PostgreSQL からより簡単に呼び出すことができるエンティティを準備することができます。これを行うには、Lambda 関数に関する以下の情報が必要です。

- [Function name] (関数名) – Lambda 関数の名前、Amazon リソースネーム (ARN)、バージョンまたはエイリアス。ステップ 2: インスタンスおよび Lambda のために IAM を設定する で作成された IAM ポリシーは ARN を必要とするため、関数の ARN を使用することをお勧めします。
- [AWS Region] (リージョン) – (オプション) Lambda 関数が RDS for PostgreSQL DB インスタンスと同じリージョンに存在しない場合の、Lambda 関数が置かれている AWS リージョン。

Lambda 関数名の情報を保持するには、aws_commons.create_lambda_function_arn 関数を使用します。このヘルパー関数は、呼び出し関数に必要な詳細を含む aws_commons._lambda_function_arn_1 複合構造を作成します。以下に、この複合構造を設定するための 3 つの代替手段を説明します。

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

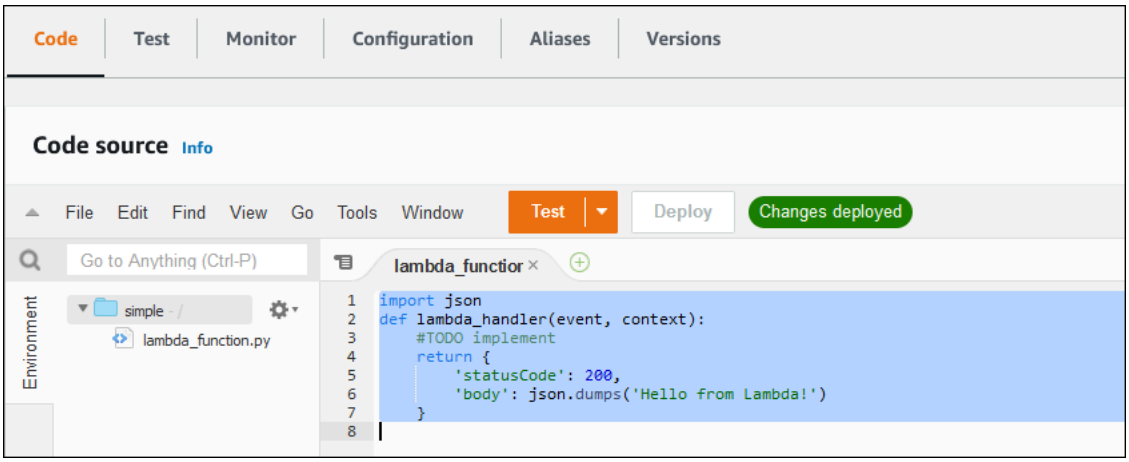
これらの値はいずれも、aws_lambda.invoke 関数の呼び出し時に使用されます。例については、ステップ 5: RDS for PostgreSQL DB インスタンスから Lambda 関数を呼びだす を参照してください。

ステップ 5: RDS for PostgreSQL DB インスタンスから Lambda 関数を呼びだす

aws_lambda.invoke 関数は、invocation_type に応じて同期または非同期的で動作します。以下のように、このパラメーターには 2 つの選択肢、RequestResponse (デフォルト) と Event があります。

- **RequestResponse** – この呼び出しタイプは同期です。これは、呼び出しタイプを指定せずに呼び出しが行われた場合のデフォルトの動作です。レスポンスペイロードには、aws_lambda.invoke 関数の結果が含まれます。処理を続行する前に Lambda 関数から結果を受け取る必要があるワークフローの場合は、この呼び出しタイプを使用します。
- **Event** – この呼び出しタイプは非同期です。この場合の応答には、結果を含むペイロードは含まれません。この呼び出しタイプは、処理を続行するために Lambda 関数の結果を必要としないワークフローで使用します。

セットアップの簡単なテストとして、psql を使用して DB インスタンスに接続し、コマンドラインからサンプル関数を起動します。今、次のスクリーンショットに示すシンプルな Python 関数のような基本的関数の 1 つが、Lambda サービスに設定されているとします。



サンプル関数を呼び出すには

1. psql または pgAdmin を使用して、 DB インスタンスに接続します。

   ```
   psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. ARN を使用して関数を呼びだします。

```
SELECT * from
 aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
 Postgres!"}'::json );
```

この応答は次のようになります。

```
status_code |                          payload                          |
 executed_version | log_result
-------------+-----------------------------------------------------------
+------------------+------------
          200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
     |
(1 row)
```

呼び出しが成功しなかった場合は、「Lambda 関数のエラーメッセージ」を参照してください。

ステップ 6: Lambda 関数を呼び出すその他のユーザー許可を付与する

手順のこの時点で、rds_superuser であるユーザーだけが Lambda 関数を呼び出すことができます。作成した関数の呼び出しを他のユーザーに許可するには、許可を付与する必要があります。

Lambda 関数を呼び出すアクセス許可を付与するには

1. psql または pgAdmin を使用して、 DB インスタンスに接続します。

```
psql -h instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. 次の SQL コマンドを実行します。

```
postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
```

例: RDS for PostgreSQL DB インスタンスから Lambda 関数を呼びだす

以下に、aws_lambda.invoke 関数の呼び出し例をいくつか示します。ほとんどの例では、関数の詳細を簡単に渡せるように、ステップ 4: RDS for PostgreSQL DB インスタンスで Lambda のヘルパー関数を使用する (オプション) で作成した複合構造 aws_lambda_arn_1 を使用しています。非同期

呼び出しの例については、「例: Lambda 関数の (Event による) 非同期呼び出し」を参照してください。ここに示されたその他の例はすべて、同期呼び出しを使用します。

Lambda 呼び出しタイプの詳細については、「AWS Lambdaデベロッパーガイド」の「Lambda 関数を呼び出す」を参照してください。aws_lambda_arn_1の詳細については、「aws_commons.create_lambda_function_arn」を参照してください。

サンプルリスト

- 例: Lambda 関数の (RequestResponse による) 同期呼び出し
- 例: Lambda 関数の (Event による) 非同期呼び出し
- 例: 関数レスポンスからの Lambda 実行ログのキャプチャリング
- 例: Lambda 関数にクライアントコンテキストを含める
- 例: Lambda 関数の特定のバージョンの呼び出し

例: Lambda 関数の (RequestResponse による) 同期呼び出し

以下に、Lambda 関数の同期呼び出しの例を 2 つ示します。これらの aws_lambda.invoke 関数呼び出しの結果は同じです。

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

パラメータの説明は次のとおりです。

- :'aws_lambda_arn_1' – このパラメータは、ヘルパー関数 aws_commons.create_lambda_function_arn を使用して、ステップ 4: RDS for PostgreSQL DB インスタンスで Lambda のヘルパー関数を使用する (オプション) で作成される複合構造を識別します。この構造は、次のように aws_lambda.invoke 呼び出しの中で、インラインで作成することもできます。

  ```
  SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
   'aws-region'),
  '{"body": "Hello from Postgres!"}'::json
  );
  ```

- '{"body": "Hello from PostgreSQL!"}'::json - Lambda関数に渡す JSON ペイロード。
- 'RequestResponse'-Lambda 呼び出しタイプ。

例: Lambda 関数の (Event による) 非同期呼び出し

以下は、Lambda 関数の非同期呼び出しの例です。Event 呼び出しタイプは、指定された入力ペイロードを使用して Lambda 関数の呼び出しをスケジュールし、すぐに返します。Lambda 関数の結果に依存しない特定のワークフローでは、Event 呼び出しタイプを使用します。

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

例: 関数レスポンスからの Lambda 実行ログのキャプチャリング

関数レスポンスに実行ログの最後の 4 KB を含めるには、log_type パラメーターを使用しながら aws_lambda.invoke 関数を呼び出します。デフォルトでは、このパラメータには None が設定されています。レスポンス内の Lambda 実行ログの結果をキャプチャする場合は、以下のように Tail を指定します。

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

aws_lambda.invoke 関数の log_type パラメータを Tail に設定して、実行ログをレスポンスに含めます。この log_type パラメータのデフォルト値は None です。

返された log_result は、base64 エンコードされた文字列です。このコンテンツは、decode と convert_from PostgreSQL 関数の組み合わせを使用してデコードできます。

log_type の詳細については、「aws_lambda.invoke」を参照してください。

例: Lambda 関数にクライアントコンテキストを含める

aws_lambda.invoke 関数では、次に示すとおり context パラメータを使用して、ペイロードとは別の情報を渡すことができます。

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

クライアントコンテキストを含めるときは、aws_lambda.invoke 関数の context パラメータに JSON オブジェクトを使用します。

context パラメータの詳細については、「aws_lambda.invoke」でリファレンスを参照してください。

例: Lambda 関数の特定のバージョンの呼び出し

aws_lambda.invoke 呼び出しに qualifier パラメータを含めることで、Lambda 関数の特定のバージョンを指定することが可能です。以下は、'*custom_version*' をバージョンのエイリアスに使用してこれを行う場合の例です。

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

代わりに、Lambda 関数名の詳細により、次のように関数の修飾子を指定することもできます。

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

qualifier および他のパラメータの詳細については、「aws_lambda.invoke」でリファレンスを参照してください。

Lambda 関数のエラーメッセージ

次のリストには、エラーメッセージに関する情報と、考えられる原因と解決策が表示されます。

- VPC 設定の問題

  VPC の設定の問題により、接続しようとすると次のエラーメッセージが表示されることがあります。

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  このエラーの一般的な原因は、VPC セキュリティグループが不適切に設定されていることです。VPC セキュリティグループのポート 443 で TCP のアウトバウンドルールが開いており、VPC が Lambda VPC に接続できるようになっていることを確認します。

プライベートの DB インスタンスを使用している場合は、VPC のプライベート DNS 設定を確認します。rds.custom_dns_resolution パラメータには 1 が設定されており、AWS PrivateLink は ステップ 1: RDS for PostgreSQL DB インスタンスで、AWS Lambda へのアウトバウンド接続を設定する。 での概説どおりにセットアップされていることを確認します。詳細については、「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

- Lambda 関数を呼び出すために必要な許可がない

  次のいずれかのエラーメッセージが表示された場合、関数を呼び出すユーザー (ロール) に適切な許可がありません。

  ```
  ERROR:  permission denied for schema aws_lambda
  ```

  ```
  ERROR:  permission denied for function invoke
  ```

  Lambda 関数を呼び出すには、ユーザー (ロール) に特定の許可を付与する必要があります。詳しくは、「ステップ 6: Lambda 関数を呼び出すその他のユーザー許可を付与する」を参照してください。

- Lambda 関数でのエラーの不適切な処理

  リクエストの処理中に Lambda 関数が例外をスローした場合、aws_lambda.invoke は、次のように PostgreSQL エラーで失敗します。

  ```
  SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
   Postgres!"}'::json);
  ERROR:  lambda invocation failed
  DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
   "Unhandled", details: "<Error details string>".
  ```

  Lambda 関数または PostgreSQL アプリケーションの中でエラーに対処します。

AWS Lambda 関数とパラメータのリファレンス

以下は、RDS for PostgreSQL で Lambda を呼び出すために使用する関数とパラメータのリファレンスです。

関数とパラメータ

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- aws_lambda パラメータ

aws_lambda.invoke

の RDS for PostgreSQL DB インスタンスの Lambda 関数を実行します。

Lambda関数の呼び出しの詳細については、AWS Lambda デベロッパーガイドの「呼び出し」も参照してください。

[Syntax] (構文)

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

入力パラメータ

function_name

Lambda 関数の識別名。値には、関数名、ARN、または部分的な ARN を指定できます。可能な形式のリストについては、AWS Lambda デベロッパーガイドの「Lambda関数名の形式」を参照してください。

payload

Lambda 関数の入力。形式には、JSON または JSONB を使用できます。詳細については、PostgreSQL ドキュメントの「JSON タイプ」を参照してください。

リージョン

(オプション) 関数の Lambda リージョン。デフォルトでは、RDS は AWS の完全な ARN から function_name リージョンを解決するか、 RDS for PostgreSQL DB インスタンスのリージョンを使用します。このリージョン値が function_name ARN で指定されたものと競合する場合、エラーが発生します。

invocation_type

Lambda 関数の呼び出しタイプ。 値は大文字と小文字が区別されます。以下に示しているのは、可能な値です。

- RequestResponse-デフォルト。Lambda 関数の呼び出しタイプは同期で、結果にレスポンスペイロードを返します。ワークフローが Lambda 関数の結果をすぐに受け取ることに依存しているときは、RequestResponse 呼び出しのタイプを使用します。
- Event- Lambda 関数の呼び出しタイプは非同期で、返されたペイロードなしにすぐに返されます。ワークフローを先に進める前に Lambda 関数の結果を知る必要がないときは、Event の呼び出しタイプを使用します。
- DryRun- この呼び出しタイプは、Lambda 関数を実行せずに、アクセスをテストします。

log_type

log_result出力パラメータで返される Lambda ログのタイプ。 値は大文字と小文字が区別されます。以下に示しているのは、可能な値です。

- Tail - 返された log_result 出力パラメータには、実行ログの最後の 4 KB が含まれます。
- None - Lambda のないログ情報は返されません。

context

JSON または JSONB形式のクライアントコンテキスト。使用されるフィールドには custom と env が含まれます 。

修飾子

呼び出される Lambda 関数のバージョンを識別する修飾子。この値が function_name ARN で指定されたものと競合する場合、エラーが発生します。

出力パラメータ

status_code

HTTP ステータスレスポンスコード。詳細については、AWS Lambda デベロッパーガイドの「Lambda 応答要素の呼び出し」を参照してください。

payload

実行された Lambda 関数から返された情報。形式は JSON または JSONB です。

executedversion

実行された Lambda 関数のバージョン。

result

Lambda 関数が呼び出されたとき `log_type` 値が `Tail` である場合に返される実行ログ情報。結果には、Base64 でエンコードされた実行ログの最後の 4 KB が含まれます。

aws_commons.create_lambda_function_arn

Lambda 関数名情報を保持するように、`aws_commons._lambda_function_arn_1` 構造を作成します。`aws_commons.create_lambda_function_arn` 関数の結果は、aws_lambda.invoke `function_name` 関数の aws_lambda.invoke パラメータで使用します。

[Syntax] (構文)

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

入力パラメータ

function_name

Lambda 関数名を含む必須のテキスト文字列。値には、関数名、部分的な ARN、または完全な ARN を指定します。

リージョン

Lambda 関数がある AWS リージョンを含む、オプションのテキスト文字列。 リージョン名と関連する値のリストについては、「」を参照してください。リージョン、アベイラビリティーゾーン、および Local Zones

aws_lambda パラメータ

この表には、aws_lambda 関数に関連するパラメータが記載されています。

パラメータ	説明
aws_lambda.connect_timeout_ms	これは動的パラメータであり、AWS Lambda への接続中の最大待機時間を設定します。デフォルト値は 1000 です。このパラメータに指定できる値は、1 ~ 900000 です。
aws_lambda.request_timeout_ms	これは動的パラメータであり、AWS Lambda からのレスポンスの最大待機時間を設定します。デフォルト値は 3000 です。このパラメータに指定できる値は、1 ~ 900000 です。
aws_lambda.endpoint_override	AWS Lambda への接続に使用できるエンドポイントを指定します。空の文字列は、リージョンのデフォルトの AWS Lambda エンドポイントを選択します。この静的パラメータの変更を有効にするには、データベースを再起動する必要があります。

Amazon RDS for PostgreSQL の一般的な DBA タスク

Amazon RDS for PostgreSQL DB インスタンスを管理するときに、データベース管理者 (DBA) は、さまざまなタスクを実行します。すでに PostgreSQL に精通している DBA の場合は、ハードウェア上で PostgreSQL を実行することと RDS for PostgreSQL との重要な違いのいくつかに注意する必要があります。例えば、マネージドサービスであるため、Amazon RDS では DB インスタンスへのシェルアクセスができません。つまり、`pg_hba.conf` および他の設定ファイルに直接アクセスすることはできません。RDS for PostgreSQL の場合、オンプレミスインスタンスの PostgreSQL 設定ファイルに通常加えられる変更は、RDS for PostgreSQL DB インスタンスに関連付けられたカスタム DB パラメータグループに対して行われます。詳細については、「Amazon RDS のパラメータグループ」を参照してください。

また、オンプレミスの PostgreSQL インスタンスと同じ方法では、ログファイルにアクセスできません。ログ記録の詳細については、「 RDS for PostgreSQL データベースログファイル」を参照してください。

別の例としては、PostgreSQL `superuser` アカウントにアクセスできなくなります。RDS for PostgreSQL では、`rds_superuser` ロールが最も高い権限を持つロールであり、設定時に `postgres` に付与されます。オンプレミスで PostgreSQL を使い慣れている場合でも、RDS for PostgreSQL を初めて使用する場合でも、`rds_superuser` ロールについて、およびロール、ユーザー、グループ、アクセス権限の操作方法を理解することをお勧めします。詳細については、「PostgreSQL のロールとアクセス権限について」を参照してください。

RDS for PostgreSQL の一般的な DBA タスクの一部を次に示します。

トピック

- RDS for PostgreSQL でサポートされる照合。
- PostgreSQL のロールとアクセス権限について
- PostgreSQL でのデッド接続の処理
- Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用
- Amazon RDS for PostgreSQL での高いオブジェクト数の管理
- Amazon RDS for PostgreSQL での TOAST OID 競合の管理
- RDS for PostgreSQL でサポートされているログ記録メカニズムの使用
- PostgreSQL による一時ファイルの管理
- pgBadger を使用した PostgreSQL でのログ分析
- PostgreSQL をモニタリングするために PGSnapper を使用する

- RDS for PostgreSQL でのカスタムキャストの管理
- RDS for PostgreSQL での並列クエリのベストプラクティス
- RDS for PostgreSQL で REPLICA IDENTITY FULL のパフォーマンスの問題を回避する
- RDSfor PostgreSQL の一般的な PostgreSQL パフォーマンス問題の初期トラブルシューティング
- RDS for PostgreSQL DB インスタンスでのパラメータの使用

RDS for PostgreSQL でサポートされる照合。

照合は、データベースに保存されている文字列をソートして比較する方法を決定する一連のルールです。照合は、コンピュータシステムにおいて基本的な役割を果たし、オペレーティングシステムの一部として組み込まれています。照合は、言語に新しい文字が追加されたり、順序規則が変更されたりすると、時間の経過とともに変化します。

照合ライブラリは、照合の特定のルールとアルゴリズムを定義します。PostgreSQL で使用される最も一般的な照合ライブラリは GNU C (glibc) と Unicode 用の国際化コンポーネント (ICU) です。デフォルトでは、RDS for PostgreSQL は、マルチバイト文字シーケンスの Unicode 文字ソート順序を含む glibc 照合を使用します。

新しい RDS for PostgreSQL の DB インスタンスを作成すると、オペレーティングシステムで使用可能な照合がチェックされます。`CREATE DATABASE` コマンド `LC_COLLATE` および `LC_CTYPE` の PostgreSQL パラメーターは、照合順序を指定するために使用され、そのデータベースのデフォルトの照合となります。または、`CREATE DATABASE` で `LOCALE` パラメータを使用して、これらのパラメータを設定することもできます。これにより、データベース内の文字列のデフォルトの照合と、文字を文字、数字、または記号として分類する規則が決まります。列、インデックス、またはクエリで使用する照合を選択することもできます。

RDS for PostgreSQL は、照合をサポートするためにオペレーティングシステムの glibc ライブラリに依存しています。RDS for PostgreSQL インスタンスは、オペレーティングシステムの最新バージョンで定期的に更新されます。これらのアップデートには glibc ライブラリの新しいバージョンが含まれることがあります。ごくまれに、新しいバージョンの glibc で一部の文字のソート順序や照合順序が変更されるため、データのソート方法が変わったり、無効なインデックスエントリが生成されることがあります。更新中に照合のソート順序の問題が見つかった場合は、インデックスの再構築が必要になることがあります。

glibc の更新による影響を減らすために、RDS for PostgreSQL に独立したデフォルトの照合ライブラリが含まれるようになりました。この照合ライブラリは、RDS for PostgreSQL 14.6、13.9、12.13、11.18、10.23、およびそれ以降のマイナーバージョンリリースで利用できま

す。glibc 2.26-59.amzn2 と互換性があり、誤ったクエリ結果を防ぐためにソート順序が安定しています。

PostgreSQL のロールとアクセス権限について

AWS マネジメントコンソール を使用して RDS for PostgreSQL DB インスタンスを作成すると、管理者アカウントが同時に作成されます。次のスクリーンショットに示すように、デフォルトでは postgres という名前になります。

デフォルト設定 (postgres) を受け入れるのではなく、別の名前を選択することもできます。この場合、選択する名前はアルファベットで始まり、1 文字以上 16 文字以下の英数字である必要があります。このガイドでは、わかりやすくするために、このメインユーザーアカウントをデフォルトの値 (postgres) で表記しています。

AWS マネジメントコンソール ではなく、create-db-instance AWS CLI を使用する場合は、コマンドの master-username パラメータと一緒に渡すことで名前を作成します。詳細については、を参照してください。Amazon RDS DB インスタンスの作成

AWS マネジメントコンソール、AWS CLI、または Amazon RDS API のいずれを使用する場合でも、またデフォルトの postgres 名を使用するか、別の名前を選択するかにかかわらず、この最初のデータベースユーザーアカウントは rds_superuser グループのメンバーであり、rds_superuser 権限を持つことになります。

トピック

- rds_superuser ロールを理解する
- PostgreSQL データベースへのユーザーアクセスのコントロール
- ユーザーパスワード管理の委任と制御
- PostgreSQL のパスワード暗号化に SCRAM を使用する

rds_superuser ロールを理解する

PostgreSQL では、ロールはデータベース内のさまざまなオブジェクトに対して、ユーザー、グループ、またはグループやユーザーに与えられた特定のアクセス権限を定義することができます。CREATE USER と CREATE GROUP に対する PostgreSQL コマンドは、データベースユーザーを区別するために、より一般的な、特定のプロパティを持つ CREATE ROLE に置き換えられました。データベースユーザーは、LOGIN 権限を持つロールと考えることができます。

> **Note**
>
> CREATE USER および CREATE GROUP コマンドは引き続き使用できます。詳細については、PostgreSQL のドキュメントの「データベースロール」セクションを参照してください。

postgres ユーザーは、 RDS for PostgreSQL DB インスタンス で最も権限があるデータベースユーザーです。ユーザーには、次の CREATE ROLE ステートメントで定義される特性があります。

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

特に指定がない限り、プロパティ NOSUPERUSER、NOREPLICATION、INHERIT、および VALID UNTIL 'infinity' が CREATE ROLE のデフォルトオプションです。

デフォルトでは、postgres には rds_superuser ロールに付与された権限と、ロールとデータベースを作成するアクセス許可があります。rds_superuser ロールでは、postgres ユーザーによる次の操作を許可します。

- Amazon RDS で使用できる拡張機能の追加 詳細については、「Amazon RDS for PostgreSQL でサポートされている PostgreSQL の機能を使用する
- ユーザーのロールを作成し、ユーザーに権限を付与します。詳細については、PostgreSQL のドキュメントの「CREATE ROLE」および「GRANT」セクションを参照してください。
- データベースの作成 詳細については、PostgreSQL のドキュメントの「CREATE DATABASE」を参照してください。
- これらの権限を持たないユーザーロールに対する rds_superuser 権限を付与し、必要に応じてそれらの権限を取り消します。このロールは、スーパーユーザータスクを実行するユーザーにのみ付与することをお勧めします。つまり、データベース管理者 (DBA) またはシステム管理者にこのロールを付与できます。

- rds_superuser ロールを持たないデータベースユーザーに rds_replication ロールを付与 (または取り消し) します。
- rds_superuser ロールを持たないデータベースユーザーに rds_password ロールを付与 (または取り消し) します。
- pg_stat_activity ビューを使用して、すべてのデータベース接続に関するステータス情報を取得します。必要に応じて、rds_superuser で pg_terminate_backend または pg_cancel_backend を使用して接続を停止できます。

CREATE ROLE postgres... ステートメントで、postgres ユーザーロールは PostgreSQL の superuser アクセス許可を特に禁止することがわかります。RDS for PostgreSQL はマネージドサービスのため、ホスト OS へのアクセスや、PostgreSQL superuser アカウントを使用した接続はできません。スタンドアロンの PostgreSQL で superuser のアクセスが必要な作業の多くは、Amazon RDS で自動的に管理されます。

権限の付与に関する詳細については、PostgreSQL のドキュメントの「GRANT」を参照してください。

rds_superuser ロールは、におけるいくつかの事前定義済みロールの 1 つです。RDS for PostgreSQL DB インスタンス。

> **Note**
>
> PostgreSQL 13 以前のリリースでは、定義済みロールはデフォルトロールと呼ばれていました。

次のリストに、新しい のために自動的に作成される他の定義済みロールの一部を示します。RDS for PostgreSQL DB インスタンス。定義済みロールとその権限は変更できません。これらの定義済みロールに対して削除、名前の変更、変更を行うことはできません。それらの操作を試みると、エラーが発生します。

- rds_password - データベースユーザーのパスワードを変更し、パスワード制約を設定できるロールです。rds_superuser ロールにはデフォルトでこのロールが付与され、データベースユーザーにロールを付与できます。詳細については、「PostgreSQL データベースへのユーザーアクセスのコントロール」を参照してください。
  - 14 より前のバージョンの RDS for PostgreSQL の場合、rds_password ロールはパスワードを変更し、データベースユーザーと rds_superuser ロールを持つユーザーのパスワード制約を

設定できます。RDS for PostgreSQL 14 以降のバージョンでは、rds_password ロールがユーザーのパスワードを変更し、パスワード制約を設定できるのは、データベースユーザーに対してのみです。rds_superuser ロールを持つユーザーのみが、rds_superuser ロールを持つ他のユーザーに対して上記のアクションを実行できます。

- rdsadmin - superuser 権限を持つ管理者がスタンドアロンの PostgreSQL データベースで行う管理タスクの多くを処理するために作成されるロールです。このロールは、RDS for PostgreSQL によって多くの管理タスクのために内部的に使用されます。
- rdstopmgr - マルチ AZ 配置をサポートするために Amazon RDS によって内部的に使用されるロールです。
- rds_reserved – データベース接続を予約するために Amazon RDS によって内部的に使用されるロール。

ロールとその権限の表示

PostgreSQL バージョン別に異なるコマンドを使用して、RDS for PostgreSQL DB インスタンスで事前定義されたロールとその権限を表示できます。事前定義されたロールをすべて表示するには、RDS for PostgreSQL DB インスタンスに接続し、psql を使用して以下のコマンドを実行します。

psql バージョン 15 以前の場合

RDS for PostgreSQL DB インスタンスに接続し、psql で \du コマンドを使用します。

```
postgres=> \du
                                                          List of roles
    Role name    |                         Attributes                          |
                 Member of
-----------------+-----------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                      +|
 {rds_superuser}
                 | Password valid until infinity                                |
 rds_ad          | Cannot login                                                 | {}
 rds_iam         | Cannot login                                                 | {}
 rds_password    | Cannot login                                                 | {}
 rds_replication | Cannot login                                                 | {}
 rds_superuser   | Cannot login                                                 |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
```

```
                | Password valid until infinity                                 |
```

psql バージョン 16 以降の場合

```
postgres=> \drg+
                              List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

バージョンに関係なくロールのメンバーシップを確認するには、次の SQL クエリを使用できます。

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

出力では、rds_superuser がデータベースユーザーロールではない (ログインできない) が、他の多くのロールの特権を持っていることがわかります。また、そのデータベースユーザー postgres は rds_superuser ロールのメンバーであることも確認できます。前述のように、postgres が Amazon RDS コンソールの [Create database] (データベースを作成) ページのデフォルト値です。別の名前を選択した場合、代わりにその名前がロールのリストに表示されます。

PostgreSQL データベースへのユーザーアクセスのコントロール

PostgreSQL の新しいデータベースは、常にデータベースの public スキーマに、すべてのデータベースユーザーとロールがオブジェクトを作成できるようなデフォルトの権限セットで作成されます。これらの権限により、例えば、データベースユーザーがデータベースに接続し、接続しながら一時テーブルを作成することができます。

RDS for PostgreSQL DB インスタンスに作成するデータベースインスタンスへのユーザーアクセスをよりよく制御するために、これらのデフォルトの public 権限を取り消すことを推奨します。その後、次の手順で示すように、データベースのユーザーに特定の権限をより詳細に付与します。

新しいデータベースインスタンスのロールと権限を設定するには

全員がデータベースへの読み取り/書き込みアクセスを必要とする複数の研究者が使用するために、新しく作成された RDS for PostgreSQL DB インスタンス上にデータベースをセットアップしているとします。

1. psql (または pgAdmin) を使用して、 RDS for PostgreSQL DB インスタンスに接続します。

   ```
   psql --host=your-db-instance.666666666666.aws-region.rds.amazonaws.com --port=5432
    --username=postgres --password
   ```

   プロンプトが表示されたら、パスワードを入力します。psql クライアントが接続し、プロンプトとしてデフォルトの管理用接続データベースである postgres=> を表示します。

2. データベースユーザーが public スキーマでオブジェクトを作成できないようにするには、次の操作を行います。

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   REVOKE
   ```

3. 次に、新しいデータベースインスタンスを作成します。

   ```
   postgres=> CREATE DATABASE lab_db;
   CREATE DATABASE
   ```

4. この新しいデータベースの PUBLIC スキーマからすべての権限を取り消します。

   ```
   postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
   REVOKE
   ```

5. データベースユーザーのロールを作成します。

   ```
   postgres=> CREATE ROLE lab_tech;
   CREATE ROLE
   ```

6. このロールを持つデータベースユーザーに、データベースに接続する機能を付与します。

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. lab_tech ロールを持つすべてのユーザーに、このデータベースのすべての権限を付与します。

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. 次のように、データベースユーザーを作成します。

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. これら 2 人のユーザーに lab_tech ロールに関連付けられた権限を付与します。

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

この時点で、lab_user1 と lab_user2 は lab_db データベースに接続できます。この例は、複数のデータベースインスタンスの作成、異なるスキーマの作成、制限されたアクセス許可の付与などを含む、エンタープライズで使用するためのベストプラクティスに従ったものではありません。詳細な情報と追加のシナリオについては、「PostgreSQL ユーザーとロールの管理」を参照してください。

PostgreSQL データベースでの権限の詳細については、PostgreSQL のドキュメントの GRANT コマンドを参照してください。

ユーザーパスワード管理の委任と制御

DBA は、ユーザーパスワードの管理を委任する場合があります。または、データベースユーザーがパスワードを変更したり、パスワードの有効期間などのパスワード制約を再設定したりしないようにする場合もあります。選択したデータベースユーザーのみがパスワード設定を変更できるようにするには、制限されたパスワード管理の機能をオンにします。この機能をアクティブにすると、rds_password ロールを付与されたデータベースユーザーのみがパスワードを管理できます。

> **Note**
>
> 制限されたパスワード管理を使用するには、RDS for PostgreSQL DB インスタンス PostgreSQL 10.6 以上を実行している必要があります。

次に示すように、デフォルトではこの機能は off になっています。

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

この機能をオンにするには、カスタムパラメータグループを使用して、rds.restrict_password_commands の設定を 1 に変更します。設定を有効にするには、RDS for PostgreSQL DB インスタンスを必ず再起動してください。

この機能をアクティブにすると、次の SQL コマンドには rds_password 権限が必要になります。

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

ロールの名前の変更 (ALTER ROLE myrole RENAME TO newname) は、パスワードが MD5 ハッシュアルゴリズムを使用する場合にも制限されます。

この機能が有効な場合、rds_password ロールのアクセス許可なしでこれらの SQL コマンドの実行を試みると、次のエラーが発生します。

```
ERROR: must be a member of rds_password to alter passwords
```

rds_password は、パスワード管理専用の少数のロールにのみ付与することをお勧めします。rds_superuser 権限を持たないデータベースユーザーに rds_password 権限を付与する場合は、CREATEROLE 属性も付与する必要があります。

パスワード要件 (クライアント側の有効期限や必要な複雑さなど) を確認してください。パスワード関連の変更に独自のクライアント側ユーティリティを使用する場合、そのユーティリティは `rds_password` のメンバーであり、`CREATE ROLE` 権限を持つ必要があります。

PostgreSQL のパスワード暗号化に SCRAM を使用する

SCRAM (Salted Challenge Response Authentication Mechanism) は、パスワードを暗号化するための PostgreSQL のデフォルトのメッセージダイジェスト (MD5) アルゴリズムの代替手段です。SCRAM 認証メカニズムは MD5 よりも安全であると見なされます。これら 2 つの異なるパスワードを保護する方法の詳細については、PostgreSQL のドキュメントの「パスワード認証」を参照してください。

に対しては、パスワード暗号化方式として MD5 ではなく SCRAM を使用することをお勧めします。RDS for PostgreSQL DB インスタンス。これは、パスワード認証と暗号化のために scram-sha-256 アルゴリズムを使用する暗号化チャレンジレスポンスのメカニズムです。

SCRAM をサポートするために、クライアントアプリケーションのライブラリを更新する必要があります。例えば、42.2.0 より前の JDBC バージョンで SCRAM はサポートされていません。詳細については、PostgreSQL JDBC ドライバーのドキュメントの「PostgreSQL JDBC ドライバー」を参照してください。その他の PostgreSQL ドライバーおよび SCRAM サポートの一覧については、PostgreSQL のドキュメントの「ドライバーの一覧」を参照してください。

RDS for PostgreSQL 13.1 以降のバージョンは scram-sha-256 をサポートします。これらのバージョンでは、次の手順で説明するように、DB インスタンスに SCRAM を要求するように設定することもできます。

SCRAM を要求するために RDS for PostgreSQL DB インスタンスを設定する

では、scram-sha-256 アルゴリズムを使用するパスワードのみを受け入れるために、に RDS for PostgreSQL DB インスタンスを要求できます。

> **Important**
>
> PostgreSQL データベースを使用する既存の RDS プロキシでは、`SCRAM` のみを使用するようにデータベース認証を変更すると、プロキシは最大 60 秒間使用できなくなります。この問題を回避するには、以下のいずれかの方法で対応します。
>
> - データベースが `SCRAM` と `MD5` 認証の両方を許可していることを確認します。

- SCRAM 認証のみを使用するには、新しいプロキシを作成し、アプリケーショントラフィックを新しいプロキシに移行してから、以前にデータベースに関連付けられていたプロキシを削除します。

システムに変更を加える前に、次の完全なプロセスを理解していることを確認してください。

- すべてのデータベースユーザーのすべてのロールとパスワードの暗号化に関する情報を取得します。
- パスワードの暗号化を制御するパラメータを指定するために、 RDS for PostgreSQL DB インスタンスのパラメータ設定を再確認してください。
- RDS for PostgreSQL DB インスタンスでデフォルトのパラメータグループを使用する場合は、カスタムの DB パラメータグループを作成して、それを RDS for PostgreSQL DB インスタンスに適用し、必要なときにパラメータを変更できるようにする必要があります。 RDS for PostgreSQL DB インスタンスがカスタムパラメータグループを使用している場合、必要に応じて、プロセスの後で必要なパラメータを変更できます。
- `password_encryption` パラメータを `scram-sha-256` に変更します。
- パスワードを更新する必要があることをすべてのデータベースユーザーに通知します。`postgres` アカウントに同じ操作を行います。新しいパスワードは暗号化され、scram-sha-256 アルゴリズムを使用して保存されます。
- 暗号化の種類を使用して、すべてのパスワードが暗号化されていることを確認します。
- すべてのパスワードで scram-sha-256 が使用されている場合、`rds.accepted_password_auth_method` パラメータを `md5+scram` から `scram-sha-256` に変更できます。

> **Warning**
>
> `rds.accepted_password_auth_method` を scram-sha-256 のみに変更した後、`md5` で暗号化されたパスワードを持つすべてのユーザー (ロール) は接続できなくなります。

RDS for PostgreSQL DB インスタンスに SCRAM を要求する準備

お使いの 、RDS for PostgreSQL DB インスタンスに変更を加える前に、既存のデータベースユーザーアカウントをすべて確認します。また、パスワードに使用されている暗号化の種類を確認して

ください。確認するためには、rds_tools 拡張機能を使用します。rds_tools をサポートする PostgreSQL バージョンを確認するには、「Extension versions for Amazon RDS for PostgreSQL」を参照してください。

データベースユーザー (ロール) とパスワードの暗号化方法のリストを取得するには

1. 次のように、psql を使用して を RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=db-name.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. rds_tools 拡張機能をインストールします。

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. ロールと暗号化のリストを取得します。

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

以下のような出力結果が表示されます。

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

カスタム DB パラメータグループの作成

> **Note**
>
> RDS for PostgreSQL DB インスタンスで既にカスタムパラメータグループを使用している場合、新しいパラメータグループを作成する必要はありません。

Amazon RDS のパラメータグループの概要については、「RDS for PostgreSQL DB インスタンスでのパラメータの使用」を参照してください。

パスワードに使用されるパスワード暗号化タイプは、1 つのパラメータ password_encryption で設定します。 RDS for PostgreSQL DB インスタンスで許可される暗号化は、別のパラメータ rds.accepted_password_auth_method で設定されます。これらのいずれかをデフォルト値から変更するには、カスタム DB パラメータグループを作成して、 インスタンスに適用する必要があります。

また、AWS マネジメントコンソール または RDS API を使用して、カスタムの DB パラメータグループを作成することもできます。詳細については、「」を参照してください。

これで、カスタムパラメータグループを DB インスタンスに関連付けることができます。

カスタム DB パラメータグループを作成するには

1. create-db-parameter-group CLI コマンドを使用して、カスタムの DB パラメータグループを作成します。この例では postgres13 をこのカスタムパラメータグループのソースとして使用します。

   Linux、macOS、Unix の場合:

   ```
   aws rds create-db-parameter-group --db-parameter-group-name 'docs-lab-scram-
   passwords' \
     --db-parameter-group-family postgres13  --description 'Custom parameter group for
    SCRAM'
   ```

   Windows の場合:

   ```
   aws rds create-db-parameter-group --db-parameter-group-name "docs-lab-scram-
   passwords" ^
   ```

```
  --db-parameter-group-family postgres13  --description "Custom DB parameter group
 for SCRAM"
```

2. modify-db-instance CLI コマンドを使用して、このカスタムパラメータグループを RDS for PostgreSQL DB クラスターに適用します。

   Linux、macOS、Unix の場合:

   ```
   aws rds modify-db-instance --db-instance-identifier 'your-instance-name' \
           --db-parameter-group-name "docs-lab-scram-passwords
   ```

   Windows の場合:

   ```
   aws rds modify-db-instance --db-instance-identifier "your-instance-name" ^
           --db-parameter-group-name "docs-lab-scram-passwords
   ```

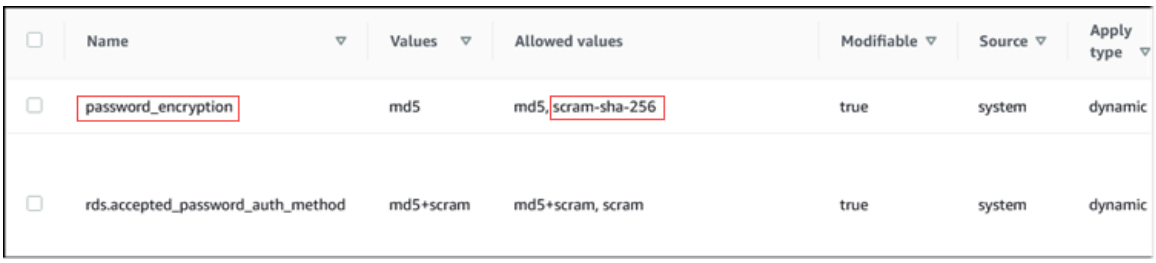
   PostgreSQL DB インスタンスとカスタム DB クラスターパラメータグループと再同期するには、プライマリインスタンスとクラスターの他のすべてのインスタンスを再起動する必要があります。ユーザーへの影響を最小限に抑えるため、定期的なメンテナンス期間中に実施するように計画してください。

SCRAM を使用するためのパスワード暗号化の設定

RDS for PostgreSQL DB インスタンスで使用されるパスワード暗号化メカニズムは、password_encryption パラメータの DB パラメータグループに設定されています。指定できる値は、未設定、md5 または scram-sha-256 です。デフォルト値は、次のように RDS for PostgreSQL のバージョンによって異なります。

- RDS for PostgreSQL 14 以上 – デフォルトは scram-sha-256
- RDS for PostgreSQL 13 – デフォルトは md5

RDS for PostgreSQL DB インスタンスにアタッチされているカスタム DB パラメータグループでは、パスワード暗号化パラメータの値を変更できます。



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	md5	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

パスワード暗号化の設定を scram-sha-256 に変更するには

- 次のように、パスワード暗号化の値を scram-sha-256 に設定します。パラメータが動的であるため、変更をすぐに適用できます。そのため、変更を有効にするために再起動は不要です。

  Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group --db-parameter-group-name \
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

  Windows の場合:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

ユーザーロールのパスワードを SCRAM に移行する

以下に説明するように、ユーザーロールのパスワードを SCRAM に移行できます。

データベースユーザー (ロール) のパスワードを MD5 から SCRAM に移行するには

1. 次のように、管理者ユーザーとしてログインします (デフォルトのユーザー名、postgres)。

```
psql --host=db-name.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 次のコマンドを使って、RDS for PostgreSQL DB インスタンスの password_encryption パラメータの設定を確認します。

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. このパラメータの値を scram-sha-256 に変更します。詳細については、「SCRAM を使用するためのパスワード暗号化の設定」を参照してください。
4. 値をもう一度チェックして、次のように scram-sha-256 に設定されていることを確認します。

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

5. パスワードの変更をすべてのデータベースユーザーに通知します。アカウント postgres (rds_superuser 権限を持つデータベースユーザー) のパスワードも必ず変更してください。

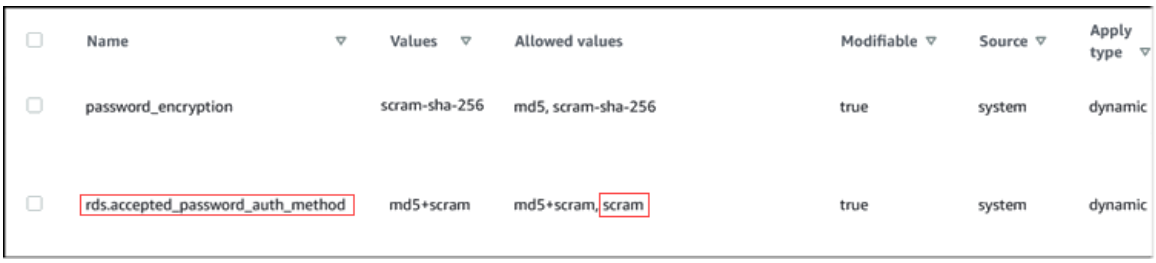
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

6. のすべてのデータベースに対してこの処理を繰り返します。RDS for PostgreSQL DB インスタンス。

SCRAM を要求するようにパラメータを変更する

これがプロセスの最後のステップです。次の手順で変更した後、パスワードに引き続き md5 暗号化を使用するユーザーアカウント (ロール) は にログインできません。RDS for PostgreSQL DB インスタンス。

rds.accepted_password_auth_method は、ログインプロセス中に RDS for PostgreSQL DB インスタンスがユーザーパスワードに対して受け入れる暗号化方式を指定します。デフォルト値は md5+scram です。つまり、どちらの方法も受け入れられます。次の画像では、このパラメータのデフォルト設定が表示されています。



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

このパラメータに指定できる値は、md5+scram または scram のみです。このパラメータの値を scram に変更すると、これが要件となります。

パスワードの SCRAM 認証を要求するようにパラメータ値を変更するには

1. RDS for PostgreSQL DB インスタンスの上のすべてのデータベースに対するすべてのデータベースユーザーパスワードが、パスワード暗号化に scram-sha-256 を使用していることを確認します。そのためには、rds_tools にロール (ユーザー) と暗号化タイプについて、次のようにクエリします。

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+-----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. のすべての DB インスタンスでクエリを繰り返します。RDS for PostgreSQL DB インスタンス。

   すべてのパスワードで scram-sha-256 が使用されている場合は続行できます。

3. 次のように、受け入れたパスワード認証の値を scram-sha-256 に設定します。

   Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group --db-parameter-group-name 'docs-lab-scram-
passwords' \
```

```
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Windows の場合:

```
aws rds modify-db-parameter-group --db-parameter-group-name "docs-lab-scram-
passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

PostgreSQL でのデッド接続の処理

デッド接続は、クライアントアプリケーションが終了または異常終了したにもかかわらず、データベースセッションがサーバー上でアクティブのままである場合に発生します。この状況は通常、データベース接続を適切に閉じたり、進行中のリクエストをキャンセルしたりせずに、クライアントプロセスがクラッシュまたは予期せず終了した場合に発生します。

PostgreSQL は、サーバープロセスがアイドル状態のとき、またはクライアントにデータを送信しようとしたときに、デッド接続を効率的に識別してクリーンアップします。ただし、アイドル状態のセッション、クライアント入力を待っているセッション、またはクエリがアクティブに実行されているセッションでは、検出は困難です。これらのシナリオを処理するために、PostgreSQL は `tcp_keepalives_*`、`tcp_user_timeout`、および `client_connection_check_interval` パラメータを提供します。

トピック

- TCP キープアライブを理解する
- RDS for PostgreSQL の主要な TCP キープアライブパラメータ
- TCP キープアライブ設定のユースケース
- ベストプラクティス

TCP キープアライブを理解する

TCP キープアライブは、接続の整合性の維持と検証に役立つプロトコルレベルのメカニズムです。各 TCP 接続は、キープアライブ動作を管理するカーネルレベルの設定を維持します。キープアライブタイマーの有効期限が切れると、システムは以下を実行します。

- データがなく、ACK フラグが設定されているプローブパケットを送信します。

- TCP/IP 仕様に従ってリモートエンドポイントからのレスポンスを待ちます。
- 応答の有無に基づいて接続状態を管理します。

RDS for PostgreSQL の主要な TCP キープアライブパラメータ

パラメータ	説明	デフォルト値
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300
`tcp_keepalives_interval`	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30
`tcp_keepalives_count`	Maximum lost keepalive messages before declaring connection dead	2
`tcp_user_timeout`	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0
`client_connection_check_interval`	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

TCP キープアライブ設定のユースケース

アイドルセッションを存続させる

アイドル状態が原因でファイアウォールまたはルーターによってアイドル状態の接続が終了されないようにするには

- キープアライブパケットを定期的に送信するように `tcp_keepalives_idle` を設定します。

デッド接続の検出

デッド接続を迅速に検出するには

- `tcp_keepalives_idle`、`tcp_keepalives_interval`、および `tcp_keepalives_count` を調整します。例えば、Aurora PostgreSQL のデフォルトでは、デッド接続を検出するのに約 1 分 (2 プローブ × 30 秒) かかります。これらの値を小さくすると、検出を速めることができます。
- `tcp_user_timeout` を使用して、確認の最大待機時間を指定します。

TCP キープアライブ設定は、カーネルがデッド接続を検出するのに役立ちますが、PostgreSQL はソケットが使用されるまで動作しない場合があります。セッションが長いクエリを実行している場合、デッド接続はクエリの完了後にのみ検出される可能性があります。PostgreSQL 14 以降のバージョンでは、`client_connection_check_interval` はクエリの実行中にソケットを定期的にポーリングすることで、デッド接続の検出を迅速化できます。

ベストプラクティス

- 妥当なキープアライブ間隔を設定する: `tcp_user_timeout`、`tcp_keepalives_idle`、`tcp_keepalives_count`、および `tcp_keepalives_interval` を調整して、検出速度とリソースの使用のバランスを取ります。
- 環境に合わせて最適化する: 設定をネットワーク動作、ファイアウォールポリシー、およびセッションのニーズに合わせます。
- PostgreSQL の機能を活用する: PostgreSQL 14 以降のバージョンで `client_connection_check_interval` を使用して、効率的な接続チェックを行います。

Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用

autovacuum 機能を使用して、PostgreSQL DB インスタンスの状態を維持することを強くお勧めします。autovacuum は、VACUUM コマンドと ANALYZE コマンドのスタートを自動化します。自動バキュームが、多数のタプルが挿入、更新、または削除されたテーブルを確認します。確認後、自動バキュームは PostgreSQL データベースから古いデータやタプルを削除することで、ストレージを再利用します。

デフォルトの PostgreSQL DB パラメータグループのいずれかを使用して作成した RDS for PostgreSQL DB インスタンスでは、デフォルトで自動バキュームがオンになっています。autovacuum 機能に関連するその他の設定パラメータもデフォルトで設定されます。これらのデフォルト値は汎用的であるため、特定のワークロードに対して、autovacuum 機能に関連付けられているパラメータの一部をチューニングすることには利点があります。

次に、autovacuum の詳細と、RDS for PostgreSQL DB インスタンスでそのパラメータの一部をチューニングする方法について説明します。概要については、「PostgreSQL を使用するためのベストプラクティス」を参照してください。

トピック

- autovacuum のメモリを割り当てる
- トランザクション ID の循環の可能性を減らす
- データベース内のテーブルにバキューム処理が必要かどうかの判別
- 現在 autovacuum の対象となっているテーブルの判別
- Autovacuum が現在実行されているかどうかと実行されている時間の判別
- 手動バキュームフリーズの実行
- autovacuum の実行中にテーブルのインデックスを再作成する
- 大きなインデックスを使った autovacuum の管理
- autovacuum に影響を与えるその他のパラメータ
- テーブルレベルの autovacuum パラメータを設定する
- 自動バキュームおよびバキュームアクティビティのログ記録
- 無効なデータベースでの自動バキュームの動作を理解する
- RDS for PostgreSQL で積極的なバキュームのブロック要因を特定して解決する
- PostgreSQL バージョン 18 での適応型 autovacuum の機能強化

autovacuum のメモリを割り当てる

autovacuum のパフォーマンスに影響を与える最も重要なパラメータの 1 つは、autovacuum_work_mem パラメータです。RDS for PostgreSQL バージョン 14 以前では、autovacuum_work_mem パラメータは -1 に設定されており、maintenance_work_mem の設定が代わりに使用されていることを示します。他のすべてのバージョンでは、autovacuum_work_mem は GREATEST({DBInstanceClassMemory/32768}, 65536) によって決定されます。

手動バキュームオペレーションは常に maintenance_work_mem 設定を使用し、デフォルト設定は GREATEST({DBInstanceClassMemory/63963136*1024}, 65536) です。また、より的を絞った手動 VACUUM オペレーションを実現するために、SET のコマンドを使用してセッションレベルで調整することもできます。

autovacuum_work_mem は、インデックスをバキュームするためのデッドタプル (pg_stat_all_tables.n_dead_tup) の識別子を保持するため、autovacuum のメモリを決定します。

計算を実行して autovacuum_work_mem パラメータの値を決定するときは、次の点に注意してください。

- パラメータの設定値が低すぎると、バキューム処理が完了するまでにテーブルを複数回スキャンすることが必要になる場合があります。このような複数のスキャンは、パフォーマンスに悪影響を及ぼすことがあります。より大きなインスタンスでは、maintenance_work_mem または autovacuum_work_mem を少なくとも 1 GB に設定することで、デッドタプル数が多いテーブルをバキュームするためのパフォーマンスが向上します。ただし、PostgreSQL バージョン 16 以前では、バキュームのメモリ使用量は 1 GB に制限されています。これは、1 回のパスで約 1 億 7,900 万個のデッドタプルを処理するのに十分な量です。テーブルのデッドタプルがこれよりも多い場合、バキュームはテーブルのインデックスを複数回通過させる必要があり、所要時間が大幅に増加します。PostgreSQL バージョン 17 以降、1 GB の制限はなく、自動バキュームは基数ツリーを使用して 1 億 7,900 万を超えるタプルを処理できます。

  タプル識別子のサイズは 6 バイトです。テーブルのインデックスのバキュームに必要なメモリを推定するには、pg_stat_all_tables.n_dead_tup をクエリしてデッドタプル数を求め、この数に 6 を掛けて、1 回のパスでインデックスをバキュームするのに必要なメモリを決定します。以下のクエリを使用できます。

```
SELECT
    relname AS table_name,
```

```
    n_dead_tup,
    pg_size_pretty(n_dead_tup * 6) AS estimated_memory
FROM
    pg_stat_all_tables
WHERE
    relname = 'name_of_the_table';
```

- autovacuum_work_mem パラメータは、autovacuum_max_workers パラメータと連動して機能します。autovacuum_max_workers 間の各ワーカーは、割り当てたメモリを使用できます。小さいテーブルが多数ある場合、autovacuum_max_workers の割り当てを増やして autovacuum_work_mem の割り当てを減らします。大きなテーブル (100 GB 以上) がある場合は、メモリの割り当てを増やしてワーカープロセス数を減らします。最も大きいテーブルを正常に処理するには、十分なメモリを割り当てる必要があります。したがって、ワーカープロセスとメモリの組み合わせが、割り当てるメモリの合計と等しくなることを確認してください。

トランザクション ID の循環の可能性を減らす

autovacuum に関連するパラメータグループの設定は、トランザクション ID の循環を防ぐほどは排除率が高くない場合があります。この問題に対処するために、RDS for PostgreSQL には autovacuum パラメータ値を自動的に適応させるメカニズムが用意されています。適応型 autovacuum は、RDS for PostgreSQL の機能です。トランザクション ID の循環に関する詳しい説明については、PostgreSQL ドキュメントを参照してください。

適応型 autovacuum は、動的パラメータ rds.adaptive_autovacuum が ON に設定されている RDS for PostgreSQL インスタンスでは、デフォルトでオンになります。この設定をオンにしておくことを強くお勧めします。ただし、autovacuum パラメータのアダプティブチューニングをオフにする場合は、rds.adaptive_autovacuum パラメータを 0 または OFF に設定します。

トランザクション ID の循環は、Amazon RDS で autovacuum パラメータをチューニングした後でも発生する場合があります。トランザクション ID の循環に対して Amazon CloudWatch アラームを実装することをお勧めします。詳細については、AWS データベースブログの記事「Implement an early warning system for transaction ID wraparound in RDS for PostgreSQL」(RDS for PostgreSQL でトランザクション ID の循環に早期警告システムを実装する) を参照してください。

自動バキュームパラメータのアダプティブチューニングをオンにすると、CloudWatch メトリクス MaximumUsedTransactionIDs が autovacuum_freeze_max_age パラメータの値または 500,000,000 のいずれか大きいほうに達したときに、Amazon RDS で自動バキュームパラメータの調整が開始されます。

テーブルでトランザクション ID の循環の傾向が続く場合、Amazon RDS では自動バキュームパラメータの調整が続行されます。続行される調整ごとに、循環を避けるために autovacuum に割り当てられる専用のリソースが増えます。Amazon RDS は、以下の autovacuum 関連のパラメータを更新します。

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`
- autovacuum_naptime
- autovacuum_max_workers (PostgreSQL バージョン 18 以降のみ)

  RDS for PostgreSQL バージョン 18 では、適応型 autovacuum が `autovacuum_max_workers` を動的にスケールします。詳細については、「PostgreSQL バージョン 18 での適応型 autovacuum の機能強化」を参照してください。

これらのパラメータが RDS で変更されるのは、新しい値で autovacuum による排除率が高くなる場合に限られます。パラメータは、DB インスタンスのメモリで変更されます。パラメータグループの値は変更されません。現在のメモリ内の設定を確認するには、PostgreSQL の SHOW SQL コマンドを使用します。

これらの自動バキュームパラメータのいずれかが Amazon RDS で変更されると、影響を受ける DB インスタンスでイベントが生成されます。このイベントは、AWS マネジメントコンソール や Amazon RDS API を介して表示できます。RDS イベントの詳細については、「Amazon RDS のイベントカテゴリとイベントメッセージ」を参照してください。これらのイベントが発生したときに通知を受け取るには、「Amazon RDS イベント通知にサブスクライブする」を参照してください。CloudWatch メトリクス `MaximumUsedTransactionIDs` がしきい値より低い値に戻ると、Amazon RDS はメモリ内の自動バキューム関連のパラメータをリセットして、パラメータグループで指定されている値に戻します。次に、この変更に対応する別のイベントが生成されます。

データベース内のテーブルにバキューム処理が必要かどうかの判別

次のクエリを使用して、データベース内のフリーズしていないトランザクションの数を表示できます。データベースの `datfrozenxid` 行の `pg_database` 列は、そのデータベースに表示されている正常なトランザクション ID の下限です。この列は、データベース内のテーブルあたりの `relfrozenxid` 値の最小数です。

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

例えば、前述のクエリの実行結果は以下のようになります。

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

データベースのトランザクション ID 数が 20 億に達すると、トランザクション ID (XID) の循環が発生し、データベースは読み取り専用になります。このクエリを使用してメトリクスを生成し、1 日に数回実行できます。デフォルトでは、autovacuum は保持するトランザクション数が 200,000,000 以下になるように設定されます ()。autovacuum_freeze_max_age

サンプルモニタリング戦略は次のようになります。

- autovacuum_freeze_max_age の値を 2 億トランザクションに設定します。
- テーブルのフリーズしていないトランザクション数が 5 億に達すると、重要度が低いアラームがトリガーされます。これは無効な値ではありませんが、autovacuum が遅れていることを示している場合があります。
- テーブルのトランザクション数が 10 億に達した場合は、対処を要するアラームとして扱う必要があります。一般的に、パフォーマンス上の理由から、トランザクション数は autovacuum_freeze_max_age に近い値にしてください。以下の推奨事項を使用して調査することをお勧めします。
- テーブルのバキューム処理されていないトランザクション数が 15 億に達すると、重要度が高いアラームがトリガーされます。データベースでトランザクション ID をどれだけ速く使用するかによりますが、このアラームは、システムに autovacuum を実行する時間がないことを示している場合があります。この場合は、この問題を早急に解決することをお勧めします。

テーブルのサイズがこれらのしきい値を頻繁に超える場合は、自動バキュームパラメータをさらに変更します。デフォルトでは、手動で VACUUM (コストベースの遅延が無効) を使用するほうが、デフォルトの autovacuum を使用するより排除率が高くなりますが、システム全体に与える負担が増えます。

次の構成を推奨します。

- この場合、最も古いトランザクションの経過時間を認識できるように、モニタリングメカニズムをオンにしてください。

  トランザクション ID の循環について警告するプロセスを作成する方法については、AWS のデータベースブログの記事「Amazon RDS for PostgreSQL でトランザクション ID の循環に早期警告システムを実装する」を参照してください。
- 処理の多いテーブルでは、autovacuum の使用に加えて、メンテナンスウィンドウ中に手動でバキュームフリーズを定期的に実行してください。手動バキュームフリーズの実行については、「手動バキュームフリーズの実行」を参照してください。

現在 autovacuum の対象となっているテーブルの判別

多くの場合、1 つ以上のテーブルにバキューム処理が必要です。`relfrozenxid` の値が `autovacuum_freeze_max_age` のトランザクション数を超えているテーブルは、常に autovacuum の処理対象となります。それ以外の場合、前回の VACUUM 以降「古い」とされたタプルの数が「バキュームしきい値」を超えると、テーブルがバキューム処理されます。

autovacuum しきい値は、次のように定義されます。

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

ここで、`vacuum base threshold` は `autovacuum_vacuum_threshold`、`vacuum scale factor` は `autovacuum_vacuum_scale_factor`、`number of tuples` は `pg_class.reltuples` です。

データベースに接続しているときに、次のクエリを実行し、自動バキュームがバキューム処理の対象と見なしているテーブルのリストを表示します。

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
```

```
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Autovacuum が現在実行されているかどうかと実行されている時間の判別

テーブルを手動でバキューム処理する必要がある場合、必ず自動バキュームが現在実行されているかどうか判別してください。実行されている場合、さらに効率的に実行されるようにパラメータを調整するか、自動バキュームを一時的にオフに切り替えて VACUUM を手動で実行できるようにする必要がある場合があります。

次のクエリを使用して、autovacuum が実行中か、どのくらいの時間実行中か、また別のセッションの待機中かを判別します。

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

クエリが実行されると、次のような出力が表示されます。

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
         |          |       |        |            |                         | FROM
 pg_stat_activity
           +
         |          |       |        |            |                         | WHERE
 query like '%VACUUM%'
          +
         |          |       |        |            |                         | ORDER BY
 xact_start;
       +
```

いくつかの問題が原因で autovacuum セッションの実行が長期間 (複数日) に渡る場合があります。最もよくある問題は、maintenance_work_mem パラメータ値で設定されたテーブルのサイズまたは更新速度が小さすぎることです。

次の計算式を使用して、maintenance_work_mem パラメータ値を設定することをお勧めします。

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

実行時間が短い autovacuum セッションは、以下の問題を示している可能性もあります。

- ワークロード用の autovacuum_max_workers が十分ではないことを示している場合があります。この場合は、ワーカーの数を指定する必要があります。
- インデックスの破損を示している場合があります (自動バキュームがクラッシュし、同じリレーションで再起動されますが進行はありません)。この場合は、手動 vacuum freeze verbose *table* を実行して正確な原因を確認します。

手動バキュームフリーズの実行

バキュームプロセスが既に実行されているテーブルで、手動バキュームを実行できます。これは、トランザクション数が 20 億に近づいている (または、モニタリングしているしきい値を上回った) テーブルに気付いた場合に役立ちます。

次の手順はガイドラインであり、プロセスにはいくつかのバリエーションがあります。例えば、テスト時に、maintenance_work_mem パラメータの設定値が小さすぎて、テーブルに早急な対処が必要であることに気づいたとします。ただし、今はインスタンスをバウンスしたくない場合があります。前のセクションのクエリを使用することで、問題のあるテーブルを判別し、長時間実行されている autovacuum セッションを確認できます。maintenance_work_mem パラメータ設定の変更が必要であることがわかっていても、すぐに対処して問題のテーブルにバキューム処理を実行する必要があります。このような場合、次の手順で対応します。

バキュームフリーズを手動で実行するには

1. バキュームを実行するテーブルを含むデータベースへのセッションを 2 つ開きます。2 番目のセッションで、接続が中断された場合にセッションを維持する「screen」または他のユーティリティを使用します。
2. セッション 1 で、テーブルで実行されている自動バキュームセッションのプロセス ID (PID) を取得します。

   次のクエリを実行し、autovacuum セッションの PID を取得します。

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```

3. セッション 2 で、このオペレーションに必要なメモリの量を計算します。この例では、このオペレーションに最大 2GB のメモリを使用できると決めたため、現在のセッションの maintenance_work_mem を 2GB に設定します。

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```

4. セッション 2 で、テーブルに対して vacuum freeze verbose コマンドを発行します。現在のところ PostgreSQL には進行状況レポートがないため、verbose 設定はアクティビティを確認するのに役立ちます。

```
\timing on
Timing is on.
vacuum freeze verbose pgbench_branches;
```

```
INFO:  vacuuming "public.pgbench_branches"
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. セッション 1 で、自動バキュームがバキュームセッションをブロックしていた場合、pg_stat_activity は、バキュームセッションを待機中であること (T) を示します。この場合は、次のように自動バキュームプロセスを終了します。

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Amazon RDS の一部の下位バージョンでは、前述のコマンドを使用して自動バキュームプロセスを終了できず、次のエラーで失敗します: ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227。

この時点で、セッションがスタートされます。このテーブルは作業リストの一番上にあると思われるため、自動バキュームは即座に再開します。

6. セッション 2 で vacuum freeze verbose コマンドを開始し、セッション 1 で自動バキュームプロセスを終了します。

autovacuum の実行中にテーブルのインデックスを再作成する

インデックスが破損した場合、autovacuum はテーブルの処理を続けますが失敗します。この状況で手動バキュームを試みると、次のようなエラーメッセージが表示されます。

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

インデックスが破損しているときに、自動バキュームをテーブルで実行しようとすると、既に実行中の自動バキュームセッションと競合します。「REINDEX」コマンドを発行する場合は、テーブルに対する排他ロックを取り除きます。書き込みオペレーションがブロックされ、この特定のインデックスを使用する読み込みオペレーションもブロックされます。

autovacuum がテーブルに対して実行されているときにテーブルのインデックスを再作成するには

1. バキュームを実行するテーブルを含むデータベースへのセッションを 2 つ開きます。2 番目のセッションで、接続が中断された場合にセッションを維持する「screen」または他のユーティリティを使用します。
2. セッション 1 で、テーブルを実行している autovacuum セッションの PID を取得します。

   次のクエリを実行し、autovacuum セッションの PID を取得します。

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. セッション 2 で、reindex コマンドを発行します。

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   REINDEX
     Time: 9.966 ms
   ```

4. セッション 1 で、自動バキュームがプロセスをブロックしていた場合、pg_stat_activity で、バキュームセッションの [waiting] (待機) が「T」であることを確認できます。この場合、自動バキュームプロセスを終了します。

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

   この時点で、セッションがスタートされます。このテーブルは作業リストの一番上にあると思われるため、autovacuum が即座に再開される点に注意することが重要です。
5. セッション 2 で コマンドを開始し、セッション 1 で自動バキュームプロセスを終了します。

大きなインデックスを使った autovacuum の管理

操作の一環として、autovacuum はテーブル上で実行している間にいくつかのバキュームフェーズを実行します。テーブルをクリーンアップする前に、まずすべてのインデックスがバキューム処理されます。複数の大きなインデックスを削除する場合、このフェーズではかなりの時間とリソースを消費します。したがって、ベストプラクティスとして、テーブル上のインデックスの数を制御し、未使用のインデックスを削除してください。

このプロセスでは、まずインデックス全体のサイズを確認します。次に、次の例に示すように、削除できるインデックスがあるかどうかを確認します。

テーブルとそのインデックスのサイズを確認するには

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

この例では、インデックスのサイズはテーブルよりも大きくなっています。この違いにより、インデックスが肥大化したり使用されなかったりするため、パフォーマンスの問題が発生し、自動バキュームや挿入オペレーションに影響する可能性があります。

未使用のインデックスを確認するには

pg_stat_user_indexes ビューを使用すると、idx_scan 列でインデックスがどのくらいの頻度で使用されているかを確認できます。次の例では、未使用のインデックスに 0 の idx_scan 値があります

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+---------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

> **Note**
>
> これらの統計情報は、統計がリセットされた時点から増加します。例えば、あるビジネス四半期末にのみ使用される、または特定のレポートにのみ使用されるインデックスがあるとします。統計がリセットされてから、このインデックスが使用されていない可能性があります。詳細については、「統計関数」を参照してください。一意性を保証するために使用されるインデックスはスキャンされないため、未使用のインデックスとして識別しないでください。未使用のインデックスを特定するには、アプリケーションとそのクエリに関する深い知識が必要です。

データベースの統計が最後にリセットされた日時を確認するには、pg_stat_database を使用してください。

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

テーブルをできるだけ早くバキューム処理する

RDS for PostgreSQL 12 以上

大きなテーブルにインデックスが多すぎる場合、DB インスタンスがトランザクション ID ラップアラウンド (XID) に近づいている可能性があります。これは XID カウンターが 0 にラップアラウンドするタイミングです。チェックを外したままにすると、この状況では、データが失われる可能性があります。ただし、インデックスをクリーンアップせずにテーブルをすばやくバキューム処理できます。RDS for PostgreSQL 12 以上では、INDEX_CLEANUP 句で VACUUM を使用することができます。

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
```

```
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

自動バキュームセッションが既に実行されている場合、手動 VACUUM を開始するにはセッションを終了する必要があります。手動バキュームフリーズの実行については、「手動バキュームフリーズの実行」を参照してください。

> **Note**
>
> インデックスのクリーンアップを定期的にスキップすると、インデックスが肥大化し、スキャンのパフォーマンスが低下します。インデックスはデッド行を保持し、テーブルはデッドラインポインタを保持します。その結果、pg_stat_all_tables.n_dead_tup は autovacuum またはインデックスクリーンアップを含む手動 VACUUM が実行されるまで増加します。ベストプラクティスとして、この手順は、トランザクション ID の循環を防ぐためにのみ使用してください。

RDS for PostgreSQL 11 以降

ただし、RDS for PostgreSQL 11 以前のバージョンでは、バキューム処理をより速く完了させる唯一の方法は、テーブルのインデックス数を減らすことです。インデックスを削除すると、クエリプランに影響する可能性があります。未使用のインデックスを最初に削除し、XID の循環が間近になったらインデックスを削除することをお勧めします。バキューム処理が完了したら、これらのインデックスを再作成できます。

autovacuum に影響を与えるその他のパラメータ

次のクエリは、autovacuum とその動作に直接影響を与えるパラメータのいくつかについて値を表示します。autovacuum パラメータの詳細については、PostgreSQL のドキュメントを参照してください。

```
SELECT name, setting, unit, short_desc
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
```

```
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

これらはすべて autovacuum に影響を与えますが、最も重要なものは以下のとおりです。

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

テーブルレベルの autovacuum パラメータを設定する

自動バキューム関連のストレージパラメータをテーブルレベルで設定できます。これは、データベース全体の動作を変更するより適切である場合があります。大きなテーブルでは、極端な設定にする必要が生じる場合がありますが、autovacuum がすべてのテーブルに対してそのように動作するわけではありません。

次のクエリは、現在テーブルレベルのオプションが設定されているテーブルを表示します。

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

これが役立つ可能性がある例として、残りのテーブルよりかなり大きいテーブルがあります。1 個の 300 GB のテーブルと、他の 30 個の 1 GB 未満のテーブルがあるとします。この場合、システム全体の動作を変更しないで、大きなテーブルのいくつかの特定のパラメータを設定できます。

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

これを行うと、このテーブルでコストベースの自動バキューム遅延がなくなりますが、システムでのリソース使用量が多くなります。通常、自動バキュームは autovacuum_cost_limit に達するた

びに autovacuum_vacuum_cost_delay で一時停止します。詳細については、「PostgreSQL ドキュメント」の「cost-based vacuuming」(コストベースのバキューム処理) を参照してください。

自動バキュームおよびバキュームアクティビティのログ記録

自動バキュームアクティビティに関する情報は、rds.force_autovacuum_logging_levelパラメータで指定したレベルに基づいて postgresql.log に送信されます。このパラメータで指定できる値、およびその値がデフォルトで設定されている PostgreSQL バージョンは次のとおりです。

- disabled (PostgreSQL 10、PostgreSQL 9.6)
- debug5, debug4, debug3, debug2, debug1
- info (PostgreSQL 12、PostgreSQL 11)
- notice
- warning (PostgreSQL 13 以降)
- error、ログ、fatal、panic

rds.force_autovacuum_logging_level では log_autovacuum_min_duration パラメータが使用されます。log_autovacuum_min_duration パラメータの値はしきい値 (ミリ秒単位) です。このしきい値を超過すると、自動バキュームアクションがログに記録されます。-1 に設定するとログに何も記録されませんが、0 に設定するとすべてのアクションが記録されます。rds.force_autovacuum_logging_level と同様に、log_autovacuum_min_duration のデフォルト値はバージョンによって次のように異なります。

- 10000 ms – PostgreSQL 14、PostgreSQL 13、PostgreSQL 12、および PostgreSQL 11
- (empty) – PostgreSQL 10 と PostgreSQL 9.6 の場合、デフォルト値はありません

rds.force_autovacuum_logging_level を WARNING に設定することをお勧めします。log_autovacuum_min_duration についても 1,000 ~ 5,000 の値に設定することをお勧めします。5,000 に設定すると、5,000 ミリ秒を超える長さのアクティビティがログに記録されます。-1 以外の設定では、ロックの競合または同時に削除されたリレーションが原因で自動バキュームアクションがスキップされた場合にも、メッセージがログに記録されます。詳細については、「PostgreSQL のドキュメント」の「Automatic Vacuuming」(自動バキューム処理) を参照してください。

問題のトラブルシューティングを行うために、rds.force_autovacuum_logging_level パラメータを debug1 から debug5 までのデバッグレベルの 1 つに変更し、最も詳しい情報を取得しま

す。デバッグ設定は、短期間かつトラブルシューティングの目的でのみ使用することをお勧めします。詳細については、「PostgreSQL のドキュメント」の「When to log」(ログ記録のタイミング) を参照してください。

> **Note**
>
> PostgreSQL では、`rds_superuser` アカウントが `pg_stat_activity` 内の autovacuum セッションを表示できます。例えば、コマンドの実行をブロックしている autovacuum セッション、あるいは手動で発行される vacuum コマンドよりも実行スピードが遅い autovacuum セッションを特定して終了することもできます。

無効なデータベースでの自動バキュームの動作を理解する

新しい値 `-2` が `pg_database` カタログの `datconnlimit` 列に導入され、DROP DATABASE 操作の途中で中断されたデータベースが無効であることが示されます。

この新しい値は、次の RDS for PostgreSQL バージョンで使用できます。

- 15.4 以降のすべてのバージョン
- 14.9 以降のバージョン
- 13.12 以降のバージョン
- 12.16 以降のバージョン
- 11.21 以降のバージョン

無効なデータベースは、有効なデータベースに対する自動バキュームのフリーズ機能には影響しません。自動バキュームは無効なデータベースを無視します。そのため、通常のバキューム操作は、PostgreSQL 環境内のすべての有効なデータベースに対して正常かつ効率的に機能し続けます。

トピック

- トランザクション ID のモニタリング
- モニタリングクエリの調整
- 無効なデータベースの問題の解決

トランザクション ID のモニタリング

age(datfrozenxid) 関数は、通常、トランザクション ID のラップアラウンドを防ぐため、データベースのトランザクション ID (XID) の経過時間をモニタリングするために使用されます。

無効なデータベースは自動バキュームから除外されるため、そのトランザクション ID (XID) カウンターは最大値 2 billion に達すると、- 2 billion にラップアラウンドして、このサイクルを無限に続けることができます。トランザクション ID のラップアラウンドをモニタリングする一般的なクエリは次のようになります。

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

ただし、datconnlimit に -2 値を導入した場合、無効なデータベースによってこのクエリの結果が歪む可能性があります。これらのデータベースは有効ではなく、定期的なメンテナンスチェックの一部に含めるべきではないため、誤検出が発生し、age(datfrozenxid) が実際よりも大きいという誤解を招く可能性があります。

モニタリングクエリの調整

正確なモニタリングを行うには、モニタリングクエリを調整して無効なデータベースを除外する必要があります。次の推奨クエリに従ってください。

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

このクエリにより、有効なデータベースのみが age(datfrozenxid) 計算で考慮され、PostgreSQL 環境全体のトランザクション ID の経過時間が正確に反映されます。

無効なデータベースの問題の解決

無効なデータベースに接続しようとすると、次のようなエラーメッセージが表示されることがあります。

```
postgres=> \c db1
```

```
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

さらに、log_min_messages パラメータが DEBUG2 以上に設定されている場合、自動バキュームプロセスが無効なデータベースをスキップしていることを示す次のログエントリが表示されることがあります。

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

この問題を解決するには、接続の試行中に提供される HINT に従ってください。RDS マスターアカウントまたは rds_superuser ロールを持つデータベースアカウントを使用して有効なデータベースに接続し、無効なデータベースを削除します。

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

RDS for PostgreSQL で積極的なバキュームのブロック要因を特定して解決する

PostgreSQL でデータベースの正常な状態を維持するためにはバキューム処理が不可欠です。バキューム処理によって、ストレージの再利用が可能になり、トランザクション ID の循環に関する問題を回避できます。しかし、バキューム処理が目的どおりに動作しなくなることもあります。これにより、パフォーマンスの低下やストレージの肥大化が生じ、トランザクション ID の循環によって DB インスタンスの可用性にも影響する場合があります。したがって、データベースのパフォーマンスと可用性を最適化するには、これらの問題を特定して解決することが不可欠です。「Understanding autovacuum in Amazon RDS for PostgreSQL environments」で自動バキュームの詳細について確認してください。

postgres_get_av_diag() 関数は、積極的なバキュームの進行を妨げたり遅らせたりしている問題を特定するのに役立ちます。推奨事項が提示され、問題が特定可能な場合はそれを解決するためのコマンドが、問題を特定できない場合は詳細な診断のためのガイダンスが得られます。積極的なバキュームのブロック要因は、経過時間が RDS の適応型自動バキュームのしきい値である 5 億トランザクション ID を超えた場合に報告されます。

トランザクション ID の経過時間とは

トランザクション ID の age() 関数は、データベース (pg_database.datfrozenxid) またはテーブル (pg_class.relfrozenxid) の最も古いフリーズしていないトランザクション ID 以降に発生したトランザクションの数を計算します。この値は、前回の積極的なバキューム操作以降のデータベースアクティビティを示し、今後の VACUUM プロセスのワークロードについての見通しを示します。

積極的なバキュームとは

積極的な VACUUM 操作では、通常の VACUUM で省略されるページも含め、テーブル内のすべてのページが包括的にスキャンされます。この徹底的なスキャンは、最大経過時間に近づいているトランザクション ID を「フリーズ」することを目指しており、トランザクション ID の循環と呼ばれる状況を効果的に防止します。

postgres_get_av_diag() で報告されるブロック要因は、5 億トランザクション以上古いものとなります。

トピック

- RDS for PostgreSQL に自動バキュームのモニタリングツールと診断ツールをインストールする
- RDS for PostgreSQL の postgres_get_av_diag() の関数
- RDS for PostgreSQL での識別可能なバキュームブロック要因の解決
- RDS for PostgreSQL での識別不能なバキュームブロック要因の解決
- RDS for PostgreSQL でバキュームのパフォーマンスに関する問題を解決する
- RDS for PostgreSQL の NOTICE メッセージの説明

RDS for PostgreSQL に自動バキュームのモニタリングツールと診断ツールをインストールする

postgres_get_av_diag() 関数は現在、次の RDS for PostgreSQL バージョンで使用できます。

- 17.2 以降の 17 バージョン

- 16.7 以降の 16 バージョン
- 15.11 以降の 15 バージョン
- 14.16 以降の 14 バージョン
- 13.19 以降の 13 バージョン

postgres_get_av_diag() を使用するには、rds_tools 拡張機能を作成します。

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

拡張機能がインストールされていることを確認します。

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+-------------------------------------------------------------
 rds_tools |   1.8   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

関数が作成されていることを確認します。

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

RDS for PostgreSQL の postgres_get_av_diag() の関数

postgres_get_av_diag() 関数は、RDS for PostgreSQL データベースで妨害または遅延が生じている自動バキュームプロセスに関する診断情報を取得します。正確な結果を得るには、最も古い

トランザクション ID を持つデータベースでクエリを実行する必要があります。最も古いトランザクション ID を持つデータベースを使用する方法の詳細については、「トランザクション ID の経過時間が最も古いデータベースに接続していない」を参照してください。

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

postgres_get_av_diag() 関数は次の情報を含むテーブルを返します。

blocker

バキュームをブロックしているデータベースアクティビティのカテゴリを指定します。

- アクティブなステートメント
- トランザクションでのアイドル状態
- 準備済みトランザクション
- 論理レプリケーションスロット
- 物理レプリケーションスロットを使用するリードレプリカ
- ストリーミングレプリケーションを使用するリードレプリカ
- 一時テーブル

database

該当しサポートされている場合にデータベースの名前を指定します。これは、アクティビティが進行中で、自動バキュームをブロックしている、またはこれからブロックするデータベースです。これは、接続してアクションを実行する必要があるデータベースです。

blocker_identifier

自動バキュームをブロックしている、またはこれからブロックするアクティビティの識別子を指定します。識別子は、プロセス ID に SQL ステートメント、準備済みトランザクション、リードレプリカの IP アドレス、および論理または物理レプリケーションスロットの名前を加えたものとなります。

wait_event

ブロックしているセッションの待機イベントを指定し、次のブロック要因に適用されます。

- アクティブなステートメント
- トランザクションでのアイドル状態

autovacum_lagging_by

バックログ作業で自動バキュームが停滞しているトランザクションの数をカテゴリごとに指定します。

suggestion

ブロック要因を解決するための推奨事項を指定します。これらの手順には、アクティビティが存在するデータベースの名前 (該当する場合)、セッションのプロセス ID (PID)(該当する場合)、および実行するアクションが含まれます。

suggested_action

ブロック要因を解決するために実行する必要があるアクションを提案します。

RDS for PostgreSQL での識別可能なバキュームブロック要因の解決

自動バキュームは積極的なバキュームを実行し、トランザクション ID の経過時間が RDS インスタンスの `autovacuum_freeze_max_age` パラメータで指定されたしきい値に達しないようにします。この経過時間は、Amazon CloudWatch メトリクス `MaximumUsedTransactionIDs` を使用して追跡できます。

Amazon RDS インスタンスの `autovacuum_freeze_max_age` の設定 (デフォルトは 2 億トランザクション ID) を確認するには、次のクエリを使用します。

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
```

```
    name = 'autovacuum_freeze_max_age';
```

postgres_get_av_diag() では、経過時間が Amazon RDS の適応型自動バキュームのしきい値である 5 億トランザクション ID を超えた場合にのみ、積極的なバキュームのブロック要因をチェックすることに注意してください。postgres_get_av_diag() で検出されるブロック要因は、5 億トランザクション以上古いものとなります。

postgres_get_av_diag() 関数は、次のタイプのブロック要因を識別します。

トピック

- アクティブなステートメント
- トランザクションでのアイドル状態
- 準備済みトランザクション
- 論理レプリケーションスロット
- リードレプリカ
- 一時テーブル

アクティブなステートメント

PostgreSQL において、アクティブなステートメントとは、データベースによって現在実行されている SQL ステートメントです。これには、クエリ、トランザクション、または進行中のすべてのオペレーションが含まれます。pg_stat_activity を使ってモニタリングする場合、状態列には、対応する PID を持つプロセスがアクティブであることが示されます。

postgres_get_av_diag() 関数がアクティブなステートメントであるステートメントを識別すると、次のような出力を表示します。

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

推奨されるアクション

suggestion 列のガイダンスに従って、ユーザーはアクティブなステートメントが存在するデータベースに接続できます。suggested_action 列で指定されているように、セッションを終了するオプションを慎重に検討することが推奨されます。終了しても安全な場合は、pg_terminate_backend( ) 関数を使用してセッションを終了できます。このアクションは、管理者 (RDS マスターアカウントなど) または必要な pg_terminate_backend( ) 権限を持つユーザーが実行できます。

> **Warning**
>
> 終了したセッションは、行われた変更を元に戻します (ROLLBACK)。要件に応じて、ステートメントを再度実行できます。ただし、自動バキュームプロセスが積極的なバキューム操作を完了した後にのみ実行することをお勧めします。

トランザクションでのアイドル状態

トランザクションステートメントのアイドル状態とは、明示的なトランザクションを開き (BEGIN ステートメントの発行など)、何らかの作業を実行したセッションで、クライアントがさらに作業を渡すか、COMMIT、ROLLBACK、END (暗黙的な COMMIT になります) を発行してトランザクションの終了を通知するのを待っているセッションを指します。

postgres_get_av_diag( ) 関数が idle in transaction ステートメントをブロック要因として識別すると、次のような出力を表示します。

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

推奨されるアクション

suggestion 列に示されているように、トランザクションセッションでのアイドル状態が存在するデータベースに接続し、pg_terminate_backend( ) 関数を使用してセッションを終了できます。ユーザーは、管理者 (RDS マスターアカウント) ユーザーでも、 pg_terminate_backend( ) 権限を持つユーザーでもかまいません。

> **Warning**
>
> 終了したセッションは、行われた変更を元に戻します (ROLLBACK)。要件に応じて、ステートメントを再度実行できます。ただし、自動バキュームプロセスが積極的なバキューム操作を完了した後にのみ実行することをお勧めします。

準備済みトランザクション

PostgreSQL では、準備済みトランザクションと呼ばれる 2 相コミット戦略の一部であるトランザクションを使用できます。これらは、max_prepared_transactions パラメータをゼロ以外の値に設定することで有効になります。準備済みトランザクションは、データベースのクラッシュ、再起動、またはクライアントの切断後も、トランザクションが持続し、引き続き使用できるように設計されています。通常のトランザクションと同様に、トランザクション ID が割り当てられ、自動バキュームに影響を与える場合があります。準備状態のままにすると、自動バキュームはフリーズを実行できず、トランザクション ID の循環につながる可能性があります。

トランザクションマネージャによって解決されることなく無期限に準備状態となったトランザクションは、孤立した準備済みトランザクションになります。これを修正する唯一の方法は、COMMIT PREPARED コマンドまたは ROLLBACK PREPARED コマンドを使用してトランザクションをコミットするかロールバックすることです。

> **Note**
>
> 準備済みトランザクションで作成されたバックアップには、復元後もそのトランザクションが含まれることに注意してください。このようなトランザクションを見つけて閉じる方法については、以下の情報を参照してください。

postgres_get_av_diag() 関数が準備済みトランザクションをブロック要因として識別すると、次の出力を表示します。

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
```

```
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

推奨されるアクション

suggestion 列で説明されているように、準備済みトランザクションがあるデータベースに接続します。suggested_action 列に基づいて、COMMIT と ROLLBACK のどちらを実行するかを慎重に検討し、適切なアクションを選択します。

準備済みトランザクション全般をモニタリングするために、PostgreSQL には pg_prepared_xacts というカタログビューが用意されています。次のクエリを使用して、準備済みトランザクションを検索できます。

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

論理レプリケーションスロット

レプリケーションスロットの目的は、ターゲットサーバーにレプリケートされるまで、未使用の変更を保持することです。詳細については、PostgreSQL の「Logical replication」を参照してください。

論理レプリケーションスロットには 2 つのタイプがあります。

非アクティブな論理レプリケーションスロット

レプリケーションが終了すると、未使用のトランザクションログは削除されず、レプリケーションスロットは非アクティブになります。非アクティブな論理レプリケーションスロットは、現在、サブスクライバーによって使用されていませんが、サーバーには残るため、WAL ファイルが保持され、古いトランザクションログが削除されなくなります。システムは LSN 情報が上書きされないよう保持する必要があるため、ディスク使用量が増加し、具体的には、自動バキュームで内部カタログテーブルをクリーンアップできなくなる可能性があります。放置するとカタログの肥大化とパフォーマンスの低下を招き、循環バキュームのリスクが増大して、トランザクションのダウンタイムが発生する可能性があります。

アクティブだが遅い論理レプリケーションスロット

論理レプリケーションのパフォーマンスの低下により、カタログのデッドタプルの削除が遅れることがあります。このレプリケーションの遅延により、catalog_xmin の更新が遅れ、カタログの肥大化や循環バキュームが発生する可能性があります。

postgres_get_av_diag() 関数が論理レプリケーションスロットをブロック要因として検出すると、次のような出力を表示します。

```
blocker               | Logical replication slot
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

推奨されるアクション

この問題を解決するには、レプリケーション設定で、適用プロセスを終了している可能性のあるターゲットスキーマまたはデータの問題を確認します。最も一般的な理由を次に示します。

- 列の欠落
- 互換性のないデータ型
- データの不一致
- データの欠落

問題がインフラストラクチャの問題に関連している場合

- ネットワークの問題 - 互換性のないネットワーク状態にある Amazon RDS DB の問題を解決するにはどうすればよいですか?
- データベースまたは DB インスタンスが、次の理由により使用できない
  - レプリカインスタンスのストレージが不足している - 「Amazon RDS DB インスタンスのストレージが不足したときに発生する問題を解決する方法を教えてください」を参照して、ストレージの追加について確認してください。

- 互換性のないパラメータ - 「互換性のないパラメータステータスのままになっている Amazon RDS DB インスタンスを修正するにはどうすればよいですか?」を参照して、問題を解決する方法の詳細を確認してください。

インスタンスが AWS ネットワーク外または AWS EC2 上にある場合は、可用性またはインフラストラクチャ関連の問題の解決方法について管理者に問い合わせてください。

非アクティブなスロットの削除

> **Warning**
>
> 注意: レプリケーションスロットを削除する前に、レプリケーションが進行中ではないこと、レプリケーションスロットが非アクティブで回復不可能な状態であることを慎重に確認してください。スロットを途中で削除すると、レプリケーションが中断されたり、データが失われたりする可能性があります。

レプリケーションスロットが不要になったことを確認したら、削除して自動バキュームを続行できるようにします。条件 `active = 'f'` を指定することで、非アクティブなスロットのみが削除されます。

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

リードレプリカ

Amazon RDS リードレプリカで `hot_standby_feedback` 設定が有効になっている場合、リードレプリカで実行されているクエリで引き続き必要になる可能性のあるデッド行は、プライマリデータベースの自動バキュームで削除されません。これは、レプリケーションスロットを使用して管理されているかどうかにかかわらず、すべてのタイプの物理リードレプリカに影響します。スタンバイレプリカで実行されているクエリでは、クエリの競合やキャンセルを防ぐために、これらの行をプライマリで利用できる状態に保つ必要があるため、こうした動作が必要になります。

物理レプリケーションスロットを使用するリードレプリカ

物理レプリケーションスロットを使用するリードレプリカでは、RDS for PostgreSQL でのレプリケーションの信頼性と安定性が大幅に強化されます。これらのスロットにより、プライマリデータベースはレプリカが処理するまで重要なログ先行書き込みファイルを保持し、ネットワークの中断中もデータ整合性を維持できます。

RDS for PostgreSQL バージョン 14 以降、すべてのレプリカでレプリケーションスロットが使用されます。以前のバージョンでは、クロスリージョンレプリカのみでレプリケーションスロットが使用されていました。

postgres_get_av_diag() 関数が物理レプリケーションスロットを使用するリードレプリカをブロック要因として検出すると、次のような出力を表示します。

```
blocker               | Read replica with physical replication slot
database              |
blocker_identifier    | rds_us_west_2_db_xxxxxxxxxxxxxxxxxxxx
wait_event            | Not applicable
autovacuum_lagging_by | 554,080,689
suggestion            | Run the following query on the replica
 "rds_us_west_2_db_xxxxxxxxxxxxxxxxxxx" to find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 757989377;

                      | Review carefully and you may consdier terminating the query on
 read replica using suggested_action. For more information, see Working with PostgreSQL
 autovacuum in the Amazon RDS User Guide.                                  +
          |
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 757989377;","
                                                                           +
                      | [OR]

                                                +
                      | ","Disable hot_standby_feedback","

                                                +
                      | [OR]

                                                +
                      | ","Delete the read replica if not needed"}
```

ストリーミングレプリケーションを使用するリードレプリカ

Amazon RDS では、バージョン 13 までの古いバージョンで、物理レプリケーションスロットを使用せずにリードレプリカを設定できます。このアプローチでは、プライマリが WAL ファイルをより積極的にリサイクルできるようにすることでオーバーヘッドを軽減します。これは、ディスク容量が制限された環境においてメリットがあり、ときどき発生するレプリケーションの遅延も許容できます。

ただし、スロットを使用しない場合、WAL ファイルが欠落しないように、スタンバイは同期したままにする必要があります。Amazon RDS は、レプリカが遅れをとった場合にアーカイブされた WAL ファイルを使用して遅れを解消しますが、このプロセスには慎重なモニタリングが必要となり、時間がかかることがあります。

postgres_get_av_diag() 関数がストリーミングリードレプリカをブロック要因として検出すると、次のような出力を表示します。

```
blocker               | Read replica with streaming replication slot
database              | Not applicable
blocker_identifier    | xx.x.x.xxx/xx
wait_event            | Not applicable
autovacuum_lagging_by | 610,146,760
suggestion            | Run the following query on the replica "xx.x.x.xxx" to find the
 long running query:
                                                                                +
                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 348319343;

               +
                      | Review carefully and you may consdier terminating the query on
 read replica using suggested_action. For more information, see Working with PostgreSQL
 autovacuum in the Amazon RDS User Guide.                                       +
                      |
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 348319343;","

                       +
                      | [OR]

                                                                                +
                      | ","Disable hot_standby_feedback","

                                                                                +
                      | [OR]

                                                                                +
                      | ","Delete the read replica if not needed"}
```

推奨されるアクション

suggested_action 列で推奨されているように、以下のオプションを慎重に検討して自動バキュームのブロックを解除します。

- クエリを終了する – suggestion 列のガイダンスに従って、suggested_action 列で指定されているようにリードレプリカに接続できます。セッションを終了するオプションは慎重に検討することをお勧めします。終了しても安全であると判断した場合は、pg_terminate_backend( ) 関数を使用してセッションを終了できます。このアクションは、管理者 (RDS マスターアカウントなど) または必要な pg_terminate_backend() 権限を持つユーザーが実行できます。

  リードレプリカで次の SQL コマンドを実行すると、プライマリのバキュームによる古い行のクリーンアップを妨げているクエリを終了できます。backend_xmin の値は、関数の出力で報告されます。

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- ホットスタンバイフィードバックを無効にする – hot_standby_feedback パラメータがバキュームの大幅な遅延の原因となっている場合は、これを無効にすることを検討します。

  hot_standby_feedback パラメータを使用すると、リードレプリカはクエリアクティビティについてプライマリに通知し、プライマリがスタンバイで使用されているテーブルや行をバキューム処理できないようにします。これによりスタンバイでのクエリの安定性が確保されますが、プライマリでのバキューム処理が大幅に遅延する可能性があります。この機能を無効にすると、プライマリはスタンバイが追いつくのを待たずにバキューム処理を進めることができます。ただし、プライマリによってバキューム処理された行にアクセスしようとすると、スタンバイでクエリのキャンセルや失敗が発生する可能性があります。
- 不要になったリードレプリカを削除する – リードレプリカが不要になった場合は、削除できます。これにより、関連するレプリケーションオーバーヘッドが解消され、プライマリがレプリカによって妨げられることなくトランザクションログをリサイクルできるようになります。

一時テーブル

TEMPORARY キーワードを使用して作成された一時テーブルは、pg_temp_xxx などの一時スキーマにあり、それらを作成したセッションのみがアクセスできます。一時テーブルは、セッションが終了すると削除されます。ただし、このようなテーブルは PostgreSQL の自動バキュームプロセスには表

示されず、テーブルを作成したセッションによって手動でバキューム処理する必要があります。別のセッションから一時テーブルのバキューム処理を試みても効果はありません。

異常な状況下では、テーブルを所有するアクティブなセッションがない状態で一時テーブルが存在します。致命的なクラッシュ、ネットワークの問題、または同様のイベントが原因でテーブルを所有するセッションが予期せず終了した場合、一時テーブルはクリーンアップされず、「孤立した」テーブルとして残される可能性があります。PostgreSQL 自動バキュームプロセスで孤立した一時テーブルが検出されると、次のメッセージがログに記録されます。

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

postgres_get_av_diag() 関数が一時テーブルをブロック要因として識別すると、次のような出力を表示します。この関数で一時テーブルに関連する出力を正しく表示するには、それらのテーブルが存在するのと同じデータベース内で関数を実行する必要があります。

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

推奨されるアクション

出力の suggestion 列に示されている手順に従って、自動バキュームの実行を妨げている一時テーブルを特定して削除します。次のコマンドを使用して、postgres_get_av_diag() で報告された一時テーブルを削除します。postgres_get_av_diag() 関数で提示された出力に基づいてテーブル名を置き換えます。

```
DROP TABLE my_temp_schema.my_temp_table;
```

次のクエリを使用して、一時テーブルを識別できます。

```
SELECT
    oid,
    relname,
```

```
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

RDS for PostgreSQL での識別不能なバキュームブロック要因の解決

このセクションでは、バキューム処理の進行を妨げる可能性のあるその他の理由について説明します。以下の問題は、現在、postgres_get_av_diag() 関数によって直接識別できません。

トピック

- 無効なページ
- インデックスの不整合
- トランザクションレートが極めて高い

無効なページ

無効なページエラーは、PostgreSQL がページへのアクセス中にページのチェックサムの不一致を検出した場合に発生します。コンテンツが読み取れないため、自動バキュームでタプルがフリーズされません。これにより、クリーンアッププロセスが実質的に停止します。次のエラーが PostgreSQL のログに書き込まれます。

```
WARNING:  page verification failed, calculated checksum YYYYY but expected XXXX
ERROR:  invalid page in block ZZZZZ of relation base/XXXXX/XXXXX
CONTEXT:  automatic vacuum of table myschema.mytable
```

オブジェクトタイプを判断する

```
ERROR: invalid page in block 4305910 of relation base/16403/186752608
WARNING: page verification failed, calculated checksum 50065 but expected 60033
```

エラーメッセージのパス base/16403/186752608 は、次の情報を提供しています。

- 「base」は PostgreSQL データディレクトリのディレクトリ名です。
- 「16403」はデータベース OID であり、pg_database システムカタログで検索できます。

- 「186752608」は relfilenode であり、pg_class システムカタログでスキーマとオブジェクト名を検索するために使用できます。

影響を受けるデータベースで次のクエリの出力をチェックすることで、オブジェクトタイプを判断できます。次のクエリは、oid: 186752608 のオブジェクト情報を取得します。OID を、発生したエラーに関連する OID に置き換えます。

```
SELECT
    relname AS object_name,
    relkind AS object_type,
    nspname AS schema_name
FROM
    pg_class c
    JOIN pg_namespace n ON c.relnamespace = n.oid
WHERE
    c.oid = 186752608;
```

詳細については、PostgreSQL のドキュメント「pg_class」で、pg_class の relkind 列で示されるサポート対象のすべてのオブジェクトタイプを参照してください。

ガイダンス

この問題の最も効果的な解決策は、特定の Amazon RDS インスタンスの設定と、整合性のないページの影響を受けるデータの種類によって異なります。

オブジェクトタイプがインデックスの場合

インデックスを再構築することをお勧めします。

- **CONCURRENTLY** オプションの使用 – PostgreSQL バージョン 12 より前のバージョンでは、インデックスを再構築するには、排他的テーブルロックによってテーブルへのアクセスを制限する必要がありました。PostgreSQL バージョン 12 以降のバージョンでは、CONCURRENTLY オプションにより行レベルのロックが可能になり、テーブルの可用性が大幅に向上しています。コマンドは以下のとおりです。

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  CONCURRENTLY はそれほど破壊的ではありませんが、ビジー状態のテーブルでは時間がかかる場合があります。可能であれば、トラフィックが少ない時間帯にインデックスを構築することを検討してください。

詳細については、PostgreSQL ドキュメントの「REINDEX」を参照してください。

- **INDEX_CLEANUP FALSE** オプションの使用 – インデックスが大きく、完了までにかなりの時間がかかると予想される場合は、インデックスを除外しながら手動 VACUUM FREEZE を実行して自動バキュームのブロックを解除できます。この機能は PostgreSQL バージョン 12 以降で使用できます。

  インデックスを無視すると、整合性のないインデックスのバキュームプロセスを省略して、循環の問題を軽減できます。ただし、無効なページの根本的な問題は解決されません。無効なページの問題に完全に対処し、これを解決するには、インデックスを再構築する必要があります。

オブジェクトタイプがマテリアライズドビューの場合

マテリアライズドビューで無効なページエラーが発生した場合は、影響を受けるデータベースにログインし、更新して無効なページを解決します。

マテリアライズドビューを更新します。

```
REFRESH MATERIALIZED VIEW schema_name.materialized_view_name;
```

更新に失敗した場合は、再作成を試みます。

```
DROP MATERIALIZED VIEW schema_name.materialized_view_name;
CREATE MATERIALIZED VIEW schema_name.materialized_view_name AS query;
```

マテリアライズドビューを更新または再作成すると、基盤となるテーブルデータに影響を与えずにマテリアライズドビューが復元されます。

他のすべてのオブジェクトタイプの場合

他のすべてのオブジェクトタイプについては、AWS サポートに問い合わせてください。

インデックスの不整合

論理的に整合性のないインデックスによって、自動バキュームの進行が妨げられる場合があります。次のエラーまたは同様のエラーは、インデックスのバキュームフェーズ中、または SQL ステートメントによってインデックスがアクセスされるときに口グに記録されます。

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

ガイダンス

手動 VACUUM FREEZE で INDEX_CLEANUP を使用してインデックスを再構築するか、インデックスを省略します。インデックスを再構築する方法の詳細については、「オブジェクトタイプがインデックスの場合」を参照してください。

- CONCURRENTLY オプションの使用 – PostgreSQL バージョン 12 より前のバージョンでは、インデックスを再構築するには、排他的テーブルロックによってテーブルへのアクセスを制限する必要がありました。PostgreSQL バージョン 12 以降のバージョンでは、CONCURRENTLY オプションにより行レベルのロックが可能になり、テーブルの可用性が大幅に向上しています。コマンドは以下のとおりです。

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  CONCURRENTLY はそれほど破壊的ではありませんが、ビジー状態のテーブルでは時間がかかる場合があります。可能であれば、トラフィックが少ない時間帯にインデックスを構築することを検討してください。詳細については、PostgreSQL ドキュメントの「REINDEX」を参照してください。
- INDEX_CLEANUP FALSE オプションの使用 – インデックスが大きく、完了までにかなりの時間がかかると予想される場合は、インデックスを除外しながら手動 VACUUM FREEZE を実行して自動バキュームのブロックを解除できます。この機能は PostgreSQL バージョン 12 以降で使用できます。

  インデックスを無視すると、整合性のないインデックスのバキュームプロセスを省略して、循環の問題を軽減できます。ただし、無効なページの根本的な問題は解決されません。無効なページの問題に完全に対処し、これを解決するには、インデックスを再構築する必要があります。

トランザクションレートが極めて高い

PostgreSQL では、トランザクションレートが高いと自動バキュームのパフォーマンスに大きな影響を与え、デッドタプルのクリーンアップが遅くなり、トランザクション ID の循環のリスクが高まります。トランザクションレートは、2 つの期間 (通常は 1 秒ごと) の `max(age(datfrozenxid))` の差を測定することでモニタリングできます。さらに、RDS Performance Insights の次のカウンター

メトリクスを使用して、トランザクションの合計数であるトランザクションレート (xact_commit と xact_rollback の合計) を測定できます。

Counter	タイプ	単位	メトリクス
xact_commit	トランザクション	1 秒あたりのコミット数	db.Transactions.xact_commit
xact_rollback	トランザクション	1 秒あたりのロールバック数	db.Transactions.xact_rollback

急激な増加は、トランザクション負荷が高いことを示しており、自動バキュームが過負荷になり、肥大化、ロック競合、および潜在的なパフォーマンスの問題が発生する可能性があります。これにより、以下のような点で自動バキュームプロセスに悪影響が及ぶ可能性があります。

- テーブルアクティビティ: バキューム処理されている特定のテーブルで大量のトランザクションが発生し、遅延が発生する可能性があります。
- システムリソース: システム全体が過負荷になっている可能性があるため、自動バキュームが効率的に機能するために必要なリソースにアクセスすることが困難になります。

自動バキュームをより効果的に動作させ、遅延なくタスクに対処するために、次の戦略を検討してください。

1. 可能であれば、トランザクションレートを下げます。可能な場合は、類似したトランザクションをバッチ処理またはグループ化することを検討してください。
2. オフピークの時間帯は、毎晩、毎週、または隔週の手動 `VACUUM FREEZE` オペレーションで頻繁に更新されるテーブルをターゲットにします。
3. インスタンスクラスをスケールアップして、大量のトランザクションと自動バキュームを処理するためにより多くのシステムリソースを割り当てることを検討してください。

RDS for PostgreSQL でバキュームのパフォーマンスに関する問題を解決する

このセクションでは、バキュームのパフォーマンスの低下を招く要因と、それらの問題に対処する方法について説明します。

トピック

- 大規模なインデックスのバキューム処理
- バキューム処理対象のテーブルまたはデータベースが多すぎる
- (循環を防ぐための) 積極的なバキューム処理が実行されている

大規模なインデックスのバキューム処理

VACUUM は、初期化、ヒープスキャン、インデックスとヒープバキューム、インデックスクリーンアップ、ヒープ切り捨て、最終クリーンアップのシーケンシャルフェーズで動作します。ヒープスキャン中、プロセスはページを除外し、デフラグしてフリーズします。ヒープスキャンが完了すると、VACUUM はインデックスをクリーンアップし、空のページがオペレーティングシステムに返されて、空き領域マップのバキューム処理や統計の更新などの最終的なクリーンアップタスクを実行します。

`maintenance_work_mem` (または `autovacuum_work_mem`) がインデックスの処理に不十分な場合は、インデックスのバキューム処理に複数のパスが必要になることがあります。PostgreSQL 16 以前では、デッドタプル ID を保存するために 1 GB のメモリ制限があり、大きなインデックスでは、多くの場合複数のパスが必要となっていました。PostgreSQL 17 では、単一の割り当て配列を使用する代わりにメモリを動的に割り当てる `TidStore` が導入されています。これにより、1 GB の制約がなくなり、メモリをより効率的に使用でき、インデックスごとに複数のインデックススキャンを行う必要が軽減されます。

使用可能なメモリがインデックス処理全体を一度に処理できない場合、大きなインデックスには PostgreSQL 17 で複数のパスが必要になることがあります。通常、大きなインデックスには、複数のパスを必要とするデッドタプルが多く含まれます。

低速バキュームオペレーションの検出

`postgres_get_av_diag()` 関数は、メモリ不足が原因でバキューム操作の実行が遅いタイミングを検出できます。この関数の詳細については、「RDS for PostgreSQL に自動バキュームのモニタリングツールと診断ツールをインストールする」を参照してください。

この `postgres_get_av_diag()` 関数は、使用可能なメモリが 1 回のパスでインデックスのバキューム処理を完了するのに十分でない場合、次の通知を発行します。

`rds_tools` 1.8

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is "XXX" and might not be
 sufficient. Consider increasing the setting, and if necessary, scaling up the Amazon
 RDS instance class for more memory.
       Additionally, review the possibility of manual vacuum with exclusion of indexes
 using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) table_name;).
```

`rds_tools` 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
       memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
 TRUE) table_name;). For more information, see
       Working with PostgreSQL autovacuum in the Amazon Amazon RDS User Guide
       .
```

> **Note**
>
> postgres_get_av_diag() 関数では、pg_stat_all_tables.n_dead_tup を使用してインデックスのバキューム処理に必要なメモリ量を推定します。

postgres_get_av_diag() 関数が、autovacuum_work_mem が不十分なために複数のインデックススキャンを必要とするスローバキュームオペレーションを特定すると、次のメッセージが生成されます。

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
       For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

ガイダンス

手動 VACUUM FREEZE を使用して次の回避策を適用し、テーブルのフリーズにかかる時間を短縮できます。

バキューム処理のためのメモリを増やす

postgres_get_av_diag() 関数で提案されているように、インスタンスレベルで潜在的なメモリの制約に対応するために、autovacuum_work_mem パラメータを増やすことをお勧めします。autovacuum_work_mem は動的パラメータですが、新しいメモリ設定を有効にするには、自動バキュームデーモンがワーカーを再起動する必要があることに注意してください。これを行うには、以下の手順を使用します。

1. 新しい設定が指定されていることを確認します。
2. 自動バキュームを現在実行しているプロセスを終了します。

このアプローチにより、調整されたメモリ割り当てが新しい自動バキューム操作に適用されます。

より迅速な結果を得るには、セッション内で maintenance_work_mem 設定を増やし、手動で VACUUM FREEZE 操作を実行することを検討してください。

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Amazon RDS を使用していて、maintenance_work_mem または autovacuum_work_mem のより高い値をサポートするために追加のメモリが必要であると判断した場合は、より多くのメモリを持つインスタンスクラスにアップグレードすることを検討してください。これにより、手動バキューム操作と自動バキューム操作の両方を強化するために必要なリソースが提供され、バキューム処理とデータベースの全体的なパフォーマンスが向上します。

INDEX_CLEANUP を無効にする

PostgreSQL バージョン 12 以降の手動 VACUUM ではインデックスのクリーンアップフェーズを省略できますが、PostgreSQL バージョン 14 以降の緊急自動バキュームでは、vacuum_failsafe_age パラメータに基づいてこのフェーズが自動的に行われます。

> **Warning**
>
> インデックスのクリーンアップを省略すると、インデックスが肥大化し、クエリのパフォーマンスに悪影響を及ぼす可能性があります。これを軽減するには、メンテナンスウィンドウ

で、影響を受けるインデックスに対してインデックスの再作成またはバキューム処理を行うことを検討してください。

大きなインデックスの処理に関するその他のガイダンスについては、「大きなインデックスを使った autovacuum の管理」のドキュメントを参照してください。

インデックスの並列バキューム処理

PostgreSQL 13 以降では、手動 VACUUM を使用して、各インデックスに 1 つのバキュームワーカープロセスを割り当て、デフォルトで複数のインデックスのバキューム処理とクリーンアップを並列して行うことができます。ただし、バキューム操作が並列実行の対象となるかどうかを PostgreSQL が判断するには、特定の基準を満たす必要があります。

- 少なくとも 2 つのインデックスが必要です。
- max_parallel_maintenance_workers パラメータを 2 以上に設定する必要があります。
- インデックスサイズが min_parallel_index_scan_size の制限 (デフォルトは 512KB) を超えている必要があります。

Amazon RDS インスタンスで使用可能な vCPU の数とテーブルのインデックスの数に基づいて max_parallel_maintenance_workers 設定を調整し、バキューム処理のターンアラウンド時間を最適化できます。

詳細については、「Parallel vacuuming in Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL」を参照してください。

バキューム処理対象のテーブルまたはデータベースが多すぎる

PostgreSQL の「The Autovacuum Daemon」ドキュメントで説明されているように、自動バキュームデーモンは複数のプロセスで動作します。このプロセスには、システム内の各データベースの自動バキュームワーカープロセスを開始する、永続的な自動バキュームランチャーが含まれます。ランチャーは、データベースあたり約 autovacuum_naptime 秒ごとにこれらのワーカーを開始するようにスケジュールします。

N 個のデータベースでは、新しいワーカーはおおよそ [autovacuum_naptime/N 秒] ごとに開始されます。ただし、同時ワーカーの合計数は autovacuum_max_workers 設定によって制限されます。バキューム処理を必要とするデータベースまたはテーブルの数がこの制限を超えると、ワーカーが利用可能になり次第すぐに次のデータベースまたはテーブルが処理されます。

多数の大きなテーブルやデータベースで同時にバキューム処理が必要な場合、使用可能なすべての自動バキュームワーカーが長時間占有され、他のテーブルやデータベースのメンテナンスに遅延が生じる可能性があります。トランザクションレートが高い環境では、このボトルネックがすぐに増大し、Amazon RDS インスタンス内で循環バキュームの問題が発生する可能性があります。

postgres_get_av_diag() が多数のテーブルまたはデータベースを検出すると、次の推奨事項が提示されます。

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

ガイダンス

autovacuum_max_workers を増やす

バキューム処理を迅速化するために、autovacuum_max_workers パラメータを調整して同時実行の自動バキュームワーカーを増やすことをお勧めします。パフォーマンスのボトルネックが続く場合は、Amazon RDS インスタンスをより多くの vCPU を持つクラスにスケールアップすることを検討してください。これにより、並列処理機能をさらに向上させることができます。

(循環を防ぐための) 積極的なバキューム処理が実行されている

PostgreSQL のデータベースの経過時間 (MaximumUsedTransactionIDs) は、(循環を防ぐための) 積極的なバキューム処理が正常に完了した場合にのみ減少します。このバキューム処理が終了するまで、トランザクションレートに応じて経過時間は増加し続けます。

postgres_get_av_diag() 関数が積極的なバキュームを検出すると、次の NOTICE を生成します。ただし、この出力は、バキュームが少なくとも 2 分間アクティブになった後にのみトリガーされます。

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

積極的なバキュームの詳細については、「When an aggressive vacuum is already running」を参照してください。

次のクエリを使用して、積極的なバキュームが進行中かどうかを確認できます。

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

出力の query 列をチェックすることで、(循環を防ぐための) 積極的なバキュームであるかどうかを判断できます。「to prevent wraparound」という語句は、それが積極的なバキュームであることを示しています。

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

例えば、トランザクション経過時間が 10 億の時点でブロック要因があり、同じトランザクション経過時間での循環を防ぐために積極的なバキューム処理を必要とするテーブルがあるとします。

さらに、トランザクション経過時間が 7 億 5,000 万の時点で別のブロック要因もあります。トランザクション経過時間 10 億でのブロック要因をクリアしても、トランザクション経過時間はすぐに 7 億 5,000 万には低下しません。積極的なバキューム処理を必要とするテーブル、または経過時間が 7 億 5,000 万を超えるトランザクションが完了するまで、高いままとなります。この間、PostgreSQL クラスターのトランザクション経過時間は増加し続けます。バキューム処理が完了すると、トランザクションの経過時間は 7 億 5,000 万に低下しますが、さらにバキューム処理が完了するまで再び増加し始めます。このサイクルは、最終的にトランザクション経過時間が `autovacuum_freeze_max_age` で指定された Amazon RDS インスタンスの設定レベルに低下するまで、これらの条件が存続する限り続きます。

RDS for PostgreSQL の NOTICE メッセージの説明

`postgres_get_av_diag()` 関数は、次の NOTICE メッセージを提供します。

経過時間がまだモニタリングしきい値に達していない場合

ブロック要因を識別するための `postgres_get_av_diag()` のモニタリングしきい値は、デフォルトで 5 億トランザクションです。`postgres_get_av_diag()` で次の NOTICE が生成された場合は、トランザクション経過時間がまだこのしきい値に達していないことを示します。

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

トランザクション ID の経過時間が最も古いデータベースに接続していない

`postgres_get_av_diag()` 関数は、トランザクション ID の経過時間が最も古いデータベースに接続したときに、最も正確な出力を提供します。`postgres_get_av_diag()` によって報告されたトランザクション ID の経過時間が最も古いデータベースが、「my_database」とは異なる場合があります。正しいデータベースに接続していない場合、次の NOTICE が生成されます。

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

トランザクション経過時間が最も古いデータベースに接続することは、次の理由で重要です。

- 一時テーブルのブロック要因の識別: 一時テーブルのメタデータは各データベースに固有のため、通常、一時テーブルは作成されたデータベースにあります。ただし、一時テーブルが上位のブロック要因となり、最も古いトランザクションを持つデータベースに存在する状況では、

誤解が生じる可能性があります。適切なデータベースに接続することで、一時テーブルのブロック要因を正確に識別できます。

- 遅いバキュームの診断: インデックスメタデータとテーブル数の情報はデータベース固有であり、バキュームが遅い問題の診断に必要です。

トランザクションの経過時間が最も古いデータベースが、rdsadmin または template0 データベースにある

場合によっては、rdsadmin または template0 データベースが、トランザクション ID の経過時間が最も古いデータベースとして識別される場合があります。このような場合、postgres_get_av_diag() で次の NOTICE が発行されます。

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

リストされたブロック要因がこれら 2 つのデータベースのいずれからも発生していないことを確認します。ブロック要因が rdsadmin または template0 のいずれかに存在すると報告された場合は、サポートに問い合わせてください。ユーザーはこれらのデータベースにはアクセスできず、サポートが必要です。

rdsadmin と template0 データベースのいずれかに上位のブロック要因が含まれている可能性はほとんどありません。

積極的なバキュームがすでに実行されている場合

postgres_get_av_diag() 関数は、積極的なバキューム処理が実行されているときに報告を行うように設計されていますが、この出力はバキュームが少なくとも 1 分間アクティブになった後にのみトリガーされます。この意図的な遅延によって、誤検出の可能性が低くなります。待機することで、有効で重要なバキュームのみが報告され、バキュームアクティビティのより正確で信頼性の高いモニタリングが可能になります。

postgres_get_av_diag() 関数は、進行中の 1 つ以上の積極的なバキュームを検出すると、次の NOTICE を生成します。

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

NOTICE に示されているように、バキュームのパフォーマンスを引き続きモニタリングします。積極的なバキュームの詳細については、「(循環を防ぐための) 積極的なバキューム処理が実行されている」を参照してください。

自動バキュームがオフの場合

データベースインスタンスで自動バキュームが無効になっている場合、postgres_get_av_diag() 関数は次の NOTICE を生成します。

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

自動バキュームは、RDS for PostgreSQL DB インスタンスの重要な機能であり、スムーズなデータベース操作を実現します。古い行バージョンを自動的に削除し、ストレージ領域を再利用して、テーブルの肥大化を防止することで、テーブルとインデックスの効率が維持され、パフォーマンスが最適化されます。さらに、Amazon RDS インスタンスのトランザクションを停止する可能性のある、トランザクション ID の循環も防止します。自動バキュームを無効にすると、データベースのパフォーマンスと安定性が長期的に低下する可能性があるため、常に有効にしておくことをお勧めします。詳細については、「Understanding autovacuum in RDS for PostgreSQL environments」を参照してください。

> **Note**
>
> 自動バキュームをオフにしても、積極的なバキュームは停止しません。積極的なバキュームは、テーブルが autovacuum_freeze_max_age しきい値に達すると実行されます。

残っているトランザクションの数が非常に少ない

postgres_get_av_diag() 関数は、循環バキュームが差し迫った場合に次の NOTICE を生成します。この NOTICE は、Amazon RDS インスタンスが新しいトランザクションを拒否するまであと 1 億トランザクションに差し迫った場合に発行されます。

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

データベースのダウンタイムを回避するために、直ちにアクションが必要です。バキューム操作を注意深くモニタリングし、トランザクションの失敗を防ぐために、影響を受けるデータベースで VACUUM FREEZE を手動で開始することを検討する必要があります。

PostgreSQL バージョン 18 での適応型 autovacuum の機能強化

RDS for PostgreSQL バージョン 18 以降、Amazon RDS は適応型 autovacuum メカニズムを強化し、DB インスタンスがトランザクション ID の循環に近づくと、autovacuum_max_workers を動的にスケーリングします。以前の PostgreSQL バージョンでは、autovacuum_max_workers を変更するには再起動が必要でした。PostgreSQL 18 では、autovacuum_max_workers が動的なパラメータとなり、Amazon RDS が再起動せずに、これを調整できるようになりました。

PostgreSQL 18 では、サーバー起動時に autovacuum ワーカー用にバックエンドプロセスのスロットを予約する新しいパラメータ autovacuum_worker_slots も導入されています。このパラメータは、同時に実行できる autovacuum ワーカーの数の上限を設定します。autovacuum_max_workers はこの値を超えることはできません。autovacuum_max_workers とは異なり、autovacuum_worker_slots を変更するには再起動が必要です。詳細については、PostgreSQL ドキュメントの「autovacuum_worker_slots」を参照してください。

適応型 autovacuum がワーカーをスケーリングする方法

PostgreSQL 18 インスタンスで MaximumUsedTransactionIDs CloudWatch メトリクスが 10 億を超えると、Amazon RDS は次の式を使用して autovacuum_max_workers を autovacuum_worker_slots 値まで増加させます。これは autovacuum_worker_slots パラメータのデフォルトの式でもあります。

```
LEAST(GREATEST({DBInstanceClassMemory/32185783296}, 16), 32)
```

中小規模のインスタンス (最大 512 GiB のメモリ) の場合、適応型 autovacuum は 16 のワーカーにスケールされます。インスタンスが大きい場合、ワーカーの数はメモリに比例して最大 32 までスケールされます。例えば、db.m5.4xlarge インスタンス (64 GiB) のデフォルトの autovacuum_max_workers は 3 です。MaximumUsedTransactionIDs が 10 億を超えると、Amazon RDS は 16 ワーカーにスケーリングします。

Amazon RDS は、既に設定されている値を減らすことはありません。設定した autovacuum_max_workers が計算値より既に高い場合、Amazon RDS はその値を変更しません。

各 autovacuum ワーカーは、autovacuum_work_mem 設定までのメモリを使用します。適応型 autovacuum がワーカー数を増やすと、autovacuum が消費するメモリの総量は比例して増加します。例えば、autovacuum_work_mem を 1 GB に設定し、適応型 autovacuum が 3 から 16 ワーカーにスケールする場合、autovacuum ワーカーが使用する最大メモリは 3 GB から 16 GB に増加します。

> **Warning**
>
> ワーカープロセスとメモリの組み合わせが、autovacuum に割り当てるメモリの合計と一致していることを確認してください。

適応型 autovacuum の変更のモニタリング

`autovacuum_max_workers` またはその他の autovacuum パラメータが Amazon RDS で変更されると、影響を受ける DB インスタンスでイベントが生成されます。これらのイベントは、AWS マネジメントコンソールまたは Amazon RDS API を通じて表示できます。RDS イベントの詳細については、「Amazon RDS のイベントカテゴリとイベントメッセージ」を参照してください。これらのイベントが発生したときに通知を受け取るには、「Amazon RDS イベント通知にサブスクライブする」を参照してください。

現在のメモリ内 autovacuum 設定を表示するには、DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW autovacuum_max_workers;
```

パラメータグループの値は変更されません。Amazon RDS は、これらのパラメータを DB インスタンスのメモリでのみ変更します。`MaximumUsedTransactionIDs` がしきい値を下回ると、Amazon RDS はパラメータをパラメータグループの値にリセットし、別のイベントを生成します。

Amazon RDS for PostgreSQL での高いオブジェクト数の管理

PostgreSQL の制限は理論上のものですが、データベース内のオブジェクト数が非常に多いと、さまざまなオペレーションにおいてパフォーマンスに顕著な影響が及びます。このドキュメントでは、合計数が多い場合にさまざまな影響を及ぼす可能性のある、いくつかの一般的なオブジェクトタイプについて説明します。

次の表は、オブジェクトタイプとその潜在的な影響の概要を示しています。

オブジェクトタイプと潜在的な影響

オブジェクトのタイプ	autovacuum	論理レプリケーション	メジャーバージョンアップグレード	pg_dump / pg_restore	全般的なパフォーマンス	インスタンスの再起動
リレーション	x		x	x	x	
一時テーブル。	x				x	
ログに記録されないテーブル		x				x
パーティション					x	
一時ファイル					x	
シーケンス		x				
ラージオブジェクト		x	x			

リレーション

PostgreSQL データベース内のテーブル数に特定のハード制限はありません。理論上の制限は非常に高いですが、データベース設計時に留意する必要がある実用的な制限が他にもあります。

影響: autovacuum が遅れる

> autovacuum は、作業量と比較してワーカーが不足しているため、トランザクション ID の増加やテーブルの肥大化に対応しきれない可能性があります。
>
> 推奨されるアクション: autovacuum をチューニングして、特定の数のテーブルと特定のワークロードに適切に対応させるには、いくつかの要素を考慮する必要があります。適切な autovacuum 設定を決定する方法については、「PostgreSQL autovacuum を使用するためのベストプラクティス」を参照してください。postgres_get_av_diag ユーティリティを使用して、トランザクション ID の増加に関する問題をモニタリングします。

影響: メジャーバージョンのアップグレード/pg_dump と復元

Amazon RDS は、pg_upgrade の実行中に「--link」オプションを使用して、データファイルのコピーを作成する必要がないようにしますが、スキーマメタデータは新しいバージョンのデータベースに復元する必要があります。並列 pg_restore を使用しても、リレーションの数が多いとダウンタイムが増加します。

影響: 一般的なパフォーマンスの低下

カタログサイズによる一般的なパフォーマンスの低下。各テーブルとそれに関連する列は、pg_attribute、pg_class、および通常のデータベースオペレーションで頻繁に使用される pg_depend テーブルに追加されます。特定の待機イベントは表示されませんが、共有バッファの効率に影響します。

推奨されるアクション: これらの特定のテーブルに対してテーブルの肥大化を定期的にチェックし、必要に応じてこれらの特定のテーブルに VACUUM FULL を実行します。カタログテーブルの VACUUM FULL では ACCESS EXCLUSIVE ロックが必要です。これは、オペレーションが完了するまで他のクエリがそれらにアクセスできないことを意味します。

影響: ファイル記述子の枯渇

エラー: 「ファイル記述子不足: システムで開いているファイルが多すぎます。ファイルを解放して再試行してください」。PostgreSQL のパラメータ max_files_per_process は、各プロセスが開くことができるファイルの数を決定します。多数のテーブルを結合する接続の数が多い場合、この制限に達する可能性があります。

推奨されるアクション:

- パラメータ max_files_per_process の値を下げると、このエラーを軽減できる可能性があります。各プロセスとサブプロセス (並列クエリなど) は、この数のファイルを開くことができ、クエリが複数のテーブルを結合している場合は、この制限を超える可能性があります。
- 接続の総数を減らし、Amazon RDS Proxy などの接続プーラー、または PgBouncer などの他のソリューションを使用します。詳細については、PGBouncer のウェブサイトを参照してください。

影響: Inode の枯渇

エラー: 「デバイスに空き容量がありません」。ストレージの空き領域が大量にある場合にこれが観察された場合は、inode が不足していることが原因です。Amazon RDS 拡張モニタリングは、使用中の inode とホストで使用できる最大数を可視化します。

概算しきい値: 数百万

一時テーブル

一時テーブルの使用は、テストデータまたは中間結果に役立ち、多くのデータベースエンジンで見られる一般的なパターンです。PostgreSQL での大量使用の影響を理解して、いくつかの落とし穴を回避する必要があります。一時テーブルの作成と削除ごとに、システムカタログテーブルに行が追加され、肥大化すると、一般的なパフォーマンスの問題が発生します。

影響: autovacuum が遅れる

一時テーブルは autovacuum によってバキューム処理されませんが、存在中はトランザクション ID を保持し、削除しないと循環が発生する可能性があります。

推奨されるアクション: 一時テーブルは、それらを作成したセッション期間中存続するか、手動で削除できます。ベストプラクティスとして、一時テーブルを使用する際に長時間実行されるトランザクションを避けることで、これらのテーブルが使用済みトランザクション ID の最大値の増加に寄与するのを防ぐことができます。

影響: 一般的なパフォーマンスの低下

カタログサイズによる一般的なパフォーマンスの低下。セッションが継続的に一時テーブルを作成および削除すると、通常のデータベースオペレーションで頻繁に使用される `pg_attribute`、`pg_class`、`pg_depend` テーブルにテーブルが追加されます。特定の待機イベントは表示されませんが、共有バッファの効率に影響します。

推奨されるアクション:

- これらの特定のテーブルに対してテーブルの肥大化を定期的にチェックし、必要に応じてこれらの特定のテーブルに `VACUUM FULL` を実行します。カタログテーブルの `VACUUM FULL` では `ACCESS EXCLUSIVE` ロックが必要です。これは、オペレーションが完了するまで他のクエリがそれらにアクセスできないことを意味します。
- 一時テーブルが頻繁に使用される場合は、メジャーバージョンアップグレードの前に、ダウンタイムを短縮するために、これらの特定のカタログテーブルの `VACUUM FULL` を強くお勧めします。

一般的なベストプラクティス:

- 共通テーブル式を使用して中間結果を生成することで、一時テーブルの使用を減らします。これらは、必要なクエリを複雑にすることがありますが、上記の影響を排除できます。

- TRUNCATE コマンドを使用して一時テーブルを再利用し、ドロップ/作成ステップを実行する代わりに内容をクリアします。これにより、一時テーブルによるトランザクション ID の増加の問題も排除されます。

概算しきい値: 数万

ログに記録されないテーブル

ログに記録されないテーブルは、WAL 情報を生成しないため、パフォーマンスを向上させることができます。これらはデータベースのクラッシュ回復中に切り捨てられるため、耐久性がなく、慎重に使用する必要があります。これは PostgreSQL ではコストの高いオペレーションであり、ログに記録されていない各テーブルは順次切り捨てられます。このオペレーションはログに記録されないテーブルの数が少ない場合は高速ですが、数千のテーブルの数になると、起動中に顕著な遅延が発生し始める可能性があります。

影響: 論理レプリケーション

論理レプリケーションでは WAL を利用して変更をキャプチャおよび転送するため、ログに記録されないテーブルは通常、ブルー/グリーンデプロイを含む論理レプリケーションに含まれません。

影響: 復旧中のダウンタイムが長引く

フェイルオーバーによるマルチ AZ 再起動、Amazon RDS ポイントインタイムリカバリ、Amazon RDS メジャーバージョンアップグレードなどのデータベースのクラッシュ回復を含むデータベース状態においては、ログに記録されないテーブルを切り捨てるシリアル化されたオペレーションが発生します。これにより、予想よりもはるかに高いダウンタイムが発生する可能性があります。

推奨されるアクション:

- ログに記録されないテーブルの使用は、データベースのクラッシュ回復オペレーション中に失われても許容できるデータのみに限定するようにしてください。
- シリアル切り捨ての現在の動作により、データベースの起動にかなりの時間がかかる可能性があるため、ログに記録されないテーブルの使用は最小限に抑えてください。

一般的なベストプラクティス:

- ログに記録されないテーブルはクラッシュ時に安全性が確保されません。クラッシュ回復を伴うポイントインタイムリカバリの開始は、各テーブルを切り捨てるシリアルプロセスであるため、PostgreSQL ではかなりの時間がかかります。

概算しきい値: 数千

パーティション

パーティション分割により、クエリのパフォーマンスが向上し、データの論理的な整理が可能になります。理想的なシナリオでは、パーティション分割は、クエリの計画と実行時にパーティションプルーニングを使用できるように構成されます。パーティションが多すぎると、クエリのパフォーマンスとデータベースのメンテナンスに悪影響を及ぼす可能性があります。クエリの計画と実行のパフォーマンスは、不十分な設計によって悪影響を受ける可能性があるため、テーブルを分割する方法の選択は慎重に行う必要があります。パーティション分割の詳細については、PostgreSQL のドキュメントを参照してください。

影響: 一般的なパフォーマンスの低下

場合によっては、計画時間のオーバーヘッドが増加し、クエリの実行計画がより複雑になり、チューニングの機会を特定することが難しくなる場合があります。18 より前の PostgreSQL バージョンでは、ワークロードの高いパーティションが多いと、`LWLock:LockManager` 待機につながる可能性があります。

推奨されるアクション: データの整理を完了させると同時に、クエリの実行パフォーマンスも向上させるために必要な最小限のパーティション数を決定します。

影響: メンテナンスの複雑さ

パーティションの数が非常に多いと、事前作成や削除などのメンテナンスの問題が発生します。autovacuum はパーティションを通常のリレーションとして扱い、定期的なクリーンアップを実行する必要があるため、タスクを完了するのに十分なワーカーが必要です。

推奨されるアクション:

- 新しいパーティションが必要になったとき (月単位のパーティションなど) や古いパーティションがロールオフされたときにワークロードがブロックされないように、パーティションを事前に作成してください。
- すべてのパーティションの通常のクリーンアップメンテナンスを実行するのに十分な autovacuum ワーカーがあることを確認します。

概算しきい値: 数百

一時ファイル

上記の一時テーブルとは異なり、一時ファイルは、複雑なクエリが複数のソート操作またはハッシュ操作を同時に実行する可能性がある場合に PostgreSQL によって作成されます。各オペレーションは、インスタンスメモリを使用して work_mem パラメータで指定された値までの結果を保存します。インスタンスメモリが不足すると、結果を保存する一時ファイルが作成されます。一時ファイルの詳細については、「一時ファイルの管理」を参照してください。ワークロードがこれらのファイルを大量に生成する場合、いくつかの影響が生じる可能性があります。

影響: ファイル記述子の枯渇

エラー: 「ファイル記述子不足: システムで開いているファイルが多すぎます。ファイルを解放して再試行してください」。PostgreSQL のパラメータ max_files_per_process は、各プロセスが開くことができるファイルの数を決定します。多数のテーブルを結合する接続の数が多い場合、この制限に達する可能性があります。

推奨されるアクション:

- パラメータ max_files_per_process の値を下げると、このエラーを軽減できる可能性があります。各プロセスとサブプロセス (並列クエリなど) は、この数のファイルを開くことができ、クエリが複数のテーブルを結合している場合は、この制限を超える可能性があります。
- 接続の総数を減らし、Amazon RDS Proxy などの接続プーラーや PgBouncer などの他のソリューションを使用します。詳細については、PGBouncer のウェブサイトを参照してください。

影響: Inode の枯渇

エラー: 「デバイスに空き容量がありません」。ストレージの空き領域が大量にある場合にこれが観察された場合は、inode が不足していることが原因です。Amazon RDS 拡張モニタリングは、使用中の inode とホストで使用できる最大数を可視化します。

一般的なベストプラクティス:

- Performance Insights を使用して一時ファイルの使用状況をモニタリングします。
- 大量の一時ファイルを生成しているクエリを調整して、一時ファイルの合計数を減らすことができるかどうかを確認します。

概算しきい値: 数千

シーケンス

シーケンスは、PostgreSQL の列の自動増分に使用される基盤となるオブジェクトであり、データの一意性とキーを提供します。これらは、論理レプリケーションを 1 つ例外として、通常のオペレーション中に特に影響を与えることなく、個々のテーブルで使用できます。

PostgreSQL では、論理レプリケーションは現在、シーケンスの現在の値をサブスクライバーにレプリケートしません。詳細については、PostgreSQL のドキュメントの制約事項ページを参照してください。

影響: スイッチオーバー時間の延長

任意のタイプの設定変更またはアップグレードに、Amazon RDS ブルー/グリーンデプロイを使用する予定がある場合は、多数のシーケンスがスイッチオーバーに与える影響を理解することが重要です。スイッチオーバーの最後のフェーズの 1 つとして、シーケンスの現在の値を同期させる処理が行われますが、その数が数千に及ぶ場合、スイッチオーバー全体の所要時間が長くなります。

推奨されるアクション: データベースワークロードがテーブルごとのシーケンスアプローチではなく共有 UUID の使用を許可する場合、スイッチオーバー中に同期ステップを削減できます。

概算しきい値: 数千

ラージオブジェクト

ラージオブジェクトは pg_largeobject という名前の単一のシステムテーブルに保存されます。各ラージオブジェクトには、システムテーブル pg_largeobject_metadata にもエントリがあります。これらのオブジェクトは、標準リレーションとは大幅に異なる方法で作成、変更、クリーンアップされます。ラージオブジェクトは autovacuum では処理されないため、vacuumlo と呼ばれる別のプロセスで定期的にクリーンアップする必要があります。ラージオブジェクトの管理の例については、「lo モジュールを使用したラージオブジェクトの管理」を参照してください。

影響: 論理レプリケーション

現在、PostgreSQL では論理レプリケーション時にラージオブジェクトはレプリケートされません。詳細については、PostgreSQL のドキュメントの制約事項ページを参照してください。ブルー/グリーン設定では、ブルー環境内のラージオブジェクトはグリーン環境にレプリケートされません。

影響: メジャーバージョンアップグレード

数百万のラージオブジェクトがあり、アップグレード中にインスタンスがそれらを処理できない場合、アップグレードはメモリ不足になり、失敗する可能性があります。PostgreSQL メジャーバージョンのアップグレードプロセスは、大きく分けて 2 つのフェーズで構成されます。pg_dump によるスキーマダンプと、pg_restore による復元です。データベースに数百万のラージオブジェクトがある場合は、アップグレード中に pg_dump と pg_restore を処理するために十分なメモリがインスタンスにあることを確認し、より大きなインスタンスタイプにスケールする必要があります。

一般的なベストプラクティス:

- vacuumlo ユーティリティを定期的に使用して、孤立したラージオブジェクトを削除します。
- ラージオブジェクトをデータベースに保存する場合は、BYTEA データ型を使用することを検討してください。

概算しきい値: 数百万

概算しきい値

このトピックで説明されている概算しきい値は、特定のリソースがスケールできる範囲を推定するためにのみ使用されます。これらは、説明されている影響が発生する可能性が高くなる一般的な範囲を示していますが、実際の動作は、特定のワークロード、インスタンスサイズ、および設定によって異なります。これらの予測を超える可能性はあるものの、記載されている影響を避けるためには、適切な管理とメンテナンスを遵守する必要があります。

Amazon RDS for PostgreSQL での TOAST OID 競合の管理

TOAST (オーバーサイズ属性ストレージ技術) は、一般的な 8KB のデータベースブロックサイズを超える大きなデータ値を処理するように設計された PostgreSQL 機能です。PostgreSQL では、物理的な行が複数のブロックにまたがることは許可されません。ブロックサイズは、行サイズの上限として機能します。TOAST は、大きなフィールド値を小さなチャンクに分割することで、この制限を克服します。メインテーブルにリンクされた専用の TOAST テーブルに個別に保存されます。詳細については、「PostgreSQL TOAST storage mechanism and implementation documentation」を参照してください。

トピック

- TOAST オペレーションについて

- パフォーマンスの課題を特定する
- 推奨事項
- モニタリング

TOAST オペレーションについて

TOAST は圧縮を実行し、大きなフィールド値を行外に保存します。TOAST は、TOAST テーブルに保存されているオーバーサイズのデータの各チャンクに一意の OID (オブジェクト識別子) を割り当てます。メインテーブルは TOAST 値 ID とリレーション ID をページに保存し、TOAST テーブルの対応する行を参照します。これにより、PostgreSQL はこれらの TOAST チャンクを効率的に見つけて管理できます。ただし、TOAST テーブルが大きくなると、システムは使用可能な OID を使い果たすリスクがあり、OID の枯渇によりパフォーマンスが低下し、ダウンタイムが発生する可能性があります。

TOAST のオブジェクト識別子

オブジェクト識別子 (OID) は、PostgreSQL がテーブル、インデックス、関数などのデータベースオブジェクトを参照するために使用するシステム全体の一意の識別子です。これらの識別子は PostgreSQL の内部オペレーションで重要な役割を果たし、データベースがオブジェクトを効率的に見つけて管理できるようにします。

トーストの対象となるデータセットを持つテーブルの場合、PostgreSQL は OID を割り当てて、関連付けられた TOAST テーブルに保存されているオーバーサイズデータの各チャンクを一意に識別します。システムは各チャンクを `chunk_id` に関連付けます。これにより、PostgreSQL はこれらのチャンクを TOAST テーブル内で効率的に整理して見つけることができます。

パフォーマンスの課題を特定する

PostgreSQL の OID 管理は、グローバル 32 ビットカウンターに依存しているため、40 億の一意の値を生成した後は最初に戻ります。データベースクラスターがこのカウンターを共有する間、OID 割り当てには TOAST オペレーション中の 2 つのステップが含まれます。

- 割り当てのグローバルカウンター – グローバルカウンターは、クラスター全体に新しい OID を割り当てます。
- 競合のローカル検索 – TOAST テーブルは、新しい OID がその特定のテーブルで既に使用されている既存の OID と競合しないようにします。

パフォーマンスの低下は、次の場合に発生する可能性があります。

- TOAST テーブルはフラグメント化または密な OID 使用率が高く、OID の割り当てが遅れます。
- データチャーンが高い環境や TOAST を多用する幅の広いテーブルがある環境では、システムは OID を頻繁に割り当てて再利用します。

詳細については、「PostgreSQL TOAST table size limits and OID allocation documentation」を参照してください。

グローバルカウンターは OID を生成し、40 億個の値ごとに最初に戻るため、システムは時折、既に使用されている値を再生成します。PostgreSQL はそれを検出し、次の OID で再試行します。TOAST テーブル内で、使用済み OID 値が非常に長く連続し、ギャップがない場合、INSERT が遅くなる可能性があります。これらの課題は、OID スペースがいっぱいになるとより顕著になり、挿入と更新が遅くなります。

問題の特定

- 単純な INSERT ステートメントは、一貫性のないランダムな方法で、通常よりも大幅に時間がかかります。
- 遅延は、TOAST オペレーションを含む INSERT および UPDATE ステートメントに対してのみ発生します。
- システムが TOAST テーブルで使用可能な OID を見つけられない場合、次のログエントリが PostgreSQL ログに表示されます。

```
LOG: still searching for an unused OID in relation "pg_toast_20815"
DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been
 found yet.
```

- Performance Insights は、LWLock:buffer_io および LWLock:OidGenLock 待機イベントに関連付けられた平均アクティブセッション (AAS) の数が多いことを示します。

  次の SQL クエリを実行して、長時間実行される INSERT トランザクションを待機イベントで識別できます。

```
SELECT
    datname AS database_name,
    usename AS database_user,
    pid,
    now() - pg_stat_activity.xact_start AS transaction_duration,
    concat(wait_event_type, ':', wait_event) AS wait_event,
    substr(query, 1, 30) AS TRANSACTION,
```

```
    state
FROM
    pg_stat_activity
WHERE (now() - pg_stat_activity.xact_start) > INTERVAL '60 seconds'
    AND state IN ('active', 'idle in transaction', 'idle in transaction (aborted)',
 'fastpath function call', 'disabled')
    AND pid <> pg_backend_pid()
AND lower(query) LIKE '%insert%'
ORDER BY
    transaction_duration DESC;
```

待機時間が長い INSERT オペレーションを表示するクエリ結果の例。

```
 database_name |  database_user  |  pid  | transaction_duration |     wait_event
 |          transaction           | state
---------------+-----------------+-------+----------------------
+---------------------+---------------------------------+--------
 postgres       | db_admin_user| 70965 | 00:10:19.484061      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 69878 | 00:06:14.976037      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 68937 | 00:05:13.942847      | :                   |
 INSERT INTO "products" (......... | active
```

問題の分離

- 小さな挿入をテストする – toast_tuple_target しきい値より小さいレコードを挿入します。TOAST ストレージの前に圧縮が適用されることに注意してください。これがパフォーマンスの問題なく動作する場合、問題は TOAST オペレーションに関連しています。
- 新しいテーブルをテストする – 同じ構造で新しいテーブルを作成し、toast_tuple_target より大きいレコードを挿入します。問題なく動作する場合、問題は元のテーブルの OID 割り当てに限定されます。

推奨事項

以下のアプローチは、TOAST OID 競合の問題を解決するのに役立ちます。

- データのクリーンアップとアーカイブ – 廃止されたデータや不要なデータを確認して削除し、将来の使用のために OID を解放するか、データをアーカイブします。次の制限事項を考慮してください。
  - 将来のクリーンアップが常に可能とは限らないため、スケーラビリティが制限されます。
  - 長時間実行される VACUUM オペレーションで、生成されたデッドタプルを削除できます。
- 新しいテーブルに書き込む – 将来の挿入用に新しいテーブルを作成し、UNION ALL ビューを使用してクエリの古いデータと新しいデータを結合します。このビューは、古いテーブルと新しいテーブルの両方から組み合わされたデータを表示し、クエリが単一のテーブルとしてそれらにアクセスできるようにします。次の制限事項を考慮してください。
  - 古いテーブルを更新すると、OID が枯渇する可能性があります。
- パーティションまたはシャード – テーブルまたはシャードデータをパーティション化して、スケーラビリティとパフォーマンスを向上させます。次の制限事項を考慮してください。
  - クエリロジックとメンテナンスの複雑さが増し、パーティション化されたデータを正しく処理するためにアプリケーションの変更が必要になる可能性があります。

モニタリング

システムテーブルの使用

PostgreSQL のシステムテーブルを使用して、OID 使用量の増加をモニタリングできます。

> **Warning**
>
> TOAST テーブルの OID の数によっては、完了までに時間がかかる場合があります。影響を最小限に抑えるために、営業時間外にモニタリングをスケジュールすることをお勧めします。

次の匿名ブロックは、各 TOAST テーブルで使用される個別の OID の数をカウントし、親テーブル情報を表示します。

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
```

```
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the number of distinct used OIDs (chunk IDs) from the TOAST table
            EXECUTE 'SELECT COUNT(DISTINCT chunk_id) FROM ' || r.toast_table INTO o;
            -- If there are used OIDs, find the associated parent table and its schema
            IF o <> 0 THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table: %
 | Number of used OIDs: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

TOAST テーブル別に OID 使用状況統計を表示する出力例。

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Number of used OIDs: 45,623,317
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Number of used OIDs: 10,000
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Number of used OIDs: 1,000,000
DO
```

次の匿名ブロックは、空でない TOAST テーブルごとに割り当てられた最大 OID を取得します。

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the max(chunk_id) from the TOAST table
            EXECUTE 'SELECT max(chunk_id) FROM ' || r.toast_table INTO o;
            -- If there's at least one TOASTed chunk, find the associated parent table
 and its schema
            IF o IS NOT NULL THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table:
 % | Max chunk_id: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

TOAST テーブルの最大チャンク ID を表示する出力例。

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Max chunk_id: 45,639,907
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Max chunk_id: 45,649,929
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Max chunk_id: 46,649,935
DO
```

Performance Insights の使用

待機イベント LWLock:buffer_io と LWLock:OidGenLock は、新しいオブジェクト識別子 (OID) の割り当てを必要とするオペレーション中に Performance Insights に表示されます。これらのイベントの高平均アクティブセッション (AAS) は、通常、OID の割り当てとリソース管理中に競合を示します。これは、データチャーンが高い環境、大規模なデータが頻繁に使用される、またはオブジェクトが頻繁に作成される環境で特に一般的です。

LWLock:buffer_io

LWLock:buffer_io は、PostgreSQL セッションが共有バッファの I/O オペレーションの完了を待機しているときに発生する待機イベントです。これは通常、データベースがディスクからメモリにデータを読み取り、変更されたページをメモリからディスクに書き込むときに発生します。BufferIO 待機イベントは、I/O オペレーションの進行中に複数のプロセスが同じバッファにアクセスまたは変更されないようにすることで、一貫性を確保します。この待機イベントの発生率が高い場合は、データベースワークロードのディスクのボトルネックまたは過剰な I/O アクティビティを示している可能性があります。

TOAST オペレーション中。

- PostgreSQL は大きなオブジェクト OID を割り当て、TOAST テーブルのインデックスをスキャンしてその一意性を確保します。
- 大きな TOAST インデックスでは、OID の一意性を検証するために複数のページにアクセスする必要がある場合があります。これにより、特にバッファプールが必要なすべてのページをキャッシュできない場合に、ディスク I/O が増加します。

インデックスのサイズは、これらのオペレーション中にアクセスする必要があるバッファページの数に直接影響します。インデックスが肥大化していない場合でも、特に高同時実行または高チャーン環境では、膨大なサイズによってバッファ I/O が増加する可能性があります。詳細については、「LWLock:BufferIO 待機イベントのトラブルシューティングガイド」を参照してください。

Lwlock:OidGenLock

`OidGenLock` は、PostgreSQL セッションが新しいオブジェクト識別子 (OID) の割り当てを待機しているときに発生する待機イベントです。このロックにより、OID が順番に安全に生成され、一度に 1 つのプロセスのみが OID を生成できます。

TOAST オペレーション中。

- TOAST テーブルのチャンクの OID 割り当て – PostgreSQL は、大きなデータレコードを管理するときに、TOAST テーブルのチャンクに OID を割り当てます。システムカタログの競合を防ぐために、各 OID は一意である必要があります。
- 高い同時実行性 – OID ジェネレーターへのアクセスはシーケンシャルであるため、複数のセッションが同時に OID を必要とするオブジェクトを作成している場合、`OidGenLock` の競合が発生する可能性があります。これにより、OID 割り当てが完了するまでセッションが待機する可能性が高まります。
- システムカタログアクセスの依存関係 – OID を割り当てるには、`pg_class` や `pg_type` などの共有システムカタログテーブルを更新する必要があります。これらのテーブルで大量のアクティビティが発生した場合 (DDL オペレーションが頻繁に発生するため)、`OidGenLock` のロック競合が増加する可能性があります。
- 高い OID 割り当ての需要 – 大量のデータレコードを持つ TOAST 負荷の高いワークロードには、一定の OID 割り当てが必要で、競合が増大します。

OID の競合を増やすその他の要因。

- 頻繁なオブジェクトの作成 – 一時テーブルなどのオブジェクトを頻繁に作成および削除するワークロードは、グローバル OID カウンターでの競合を増幅します。
- グローバルカウンターロック – グローバル OID カウンターは一意性を確保するために連続してアクセスされ、同時実行性の高い環境で単一の競合ポイントを作成します。

RDS for PostgreSQL でサポートされているログ記録メカニズムの使用

いくつかのパラメータ、エクステンション、その他の設定可能な項目を設定して、PostgreSQL DB インスタンスで発生するアクティビティのログを作成できます。これには以下が含まれます。

- `log_statement` パラメータは PostgreSQL データベースのユーザー操作のログを作成するのに使用できます。RDS for PostgreSQL のログ記録とログのモニタリング方法の詳細については、「RDS for PostgreSQL データベースログファイル」を参照してください。

- `rds.force_admin_logging_level` パラメータにより、DB インスタンス上のデータベースでの Amazon RDS 内部ユーザー (rdsadmin) によるアクションをログに記録されます。出力が PostgreSQL エラーログに書き込まれます。指定可能な値は、`disabled`、`debug5`、`debug4`、`debug3`、`debug2`、`debug1`、`info`、`notice`、`warning`、`error` および `panic` です。デフォルト値は `disabled` です。
- `rds.force_autovacuum_logging_level` パラメータを設定して、PostgreSQL エラーログにさまざまな自動バキュームオペレーションをキャプチャすることができます。詳細については、「自動バキュームおよびバキュームアクティビティのログ記録」を参照してください。
- PostgreSQL Audit (pgAudit) 拡張は、セッションレベルまたはオブジェクトレベルでアクティビティをキャプチャするようにインストールおよび設定できます。詳細については、「pgAudit を使用してデータベースのアクティビティを記録する」を参照してください。
- `log_fdw` 拡張を使用すると、SQL を使用してデータベースエンジンのログにアクセスすることができます。詳細については、「SQL を使用した DB ログのアクセスのための log_fdw 拡張機能の使用」を参照してください。
- `pg_stat_statements` ライブラリは、PostgreSQL バージョン 10 以降の RDS で、`shared_preload_libraries` パラメータのデフォルトとして指定されています。実行中のクエリを分析するために使用できるのはこのライブラリです。DB パラメータグループで `pg_stat_statements` が設定されていることを確認してください。このライブラリが提供する情報を使用した RDS for PostgreSQL DB インスタンスのモニタリングの詳細については、「RDS PostgreSQL での SQL 統計」を参照してください。
- `log_hostname` パラメータは、各クライアント接続のホスト名をログに取り込みます。PostgreSQL バージョン 12 以降のバージョンの RDS では、このパラメータはデフォルトで `off` に設定されています。オンにする場合は、必ずセッション接続時間をモニタリングしてください。オンにすると、サービスはドメインネームシステム (DNS) 逆ルックアップリクエストを使用して接続しているクライアントのホスト名を取得し、PostgreSQL ログに追加します。これはセッション接続中に顕著な影響を及ぼします。このパラメータはトラブルシューティングの目的のみでオンにすることをお勧めします。

一般に、ログ記録のポイントは、DBA がモニタリング、パフォーマンスのチューニング、およびトラブルシューティングを実行できるようにすることです。ログの多くは、Amazon CloudWatch または Performance Insights に自動的にアップロードされます。ここでは、DB インスタンスの完全なメトリクスを提供するためにソートおよびグループ化されています。Amazon RDS のモニタリングとメトリクスの詳細については、「Amazon RDS インスタンスでのメトリクスのモニタリング」を参照してください。

PostgreSQL による一時ファイルの管理

PostgreSQL では、複雑なクエリが、インスタンスメモリを使用して work_mem パラメータで指定された値までの結果を保存する、複数のソート操作またはハッシュ操作を同時に実行する場合があります。インスタンスメモリが不足すると、結果を保存する一時ファイルが作成されます。これらは、クエリの実行を完了するためにディスクに書き込まれます。その後、これらのファイルは、クエリが完了すると自動的に削除されます。RDS for PostgreSQL では、これらのファイルはデータボリュームの Amazon EBS に保存されます。詳細については、「Amazon RDS DB インスタンスのストレージ」を参照してください。CloudWatch で提供される FreeStorageSpace メトリクスを常にモニタリングして、DB インスタンスのストレージに十分な空き容量があることを確認できます。詳細については、「FreeStorageSpace 」を参照してください。

複数のクエリが同時に実行され、一時ファイルの使用量が増えるワークロードには、Amazon RDS Optimized Read インスタンスを使用することをお勧めします。これらのインスタンスがローカルの不揮発性メモリエクスプレス (NVMe) ベースのソリッドステートドライブ (SSD) ブロックレベルストレージを使用することで、一時ファイルを配置します。詳細については、「Amazon RDS Optimized Reads による RDS for PostgreSQL のクエリパフォーマンスの向上」を参照してください。

以下のパラメータと関数を使用して、インスタンスの一時ファイルを管理することができます。

- **temp_file_limit** — このパラメータは、temp_files のサイズ (KB 単位) を超えるクエリをすべてキャンセルします。この制限により、クエリが延々と実行され、一時ファイルでディスクスペースが消費されるのを防ぐことができます。log_temp_files パラメータの結果を使用して値を推定できます。ベストプラクティスとして、ワークロードの動作を調べ、推定値に従って制限を設定してください。次の例は、クエリが制限を超えた場合にキャンセルされる様子を示しています。

```
postgres=>select * from pgbench_accounts, pg_class, big_table;
```

```
ERROR: temporary file size exceeds temp_file_limit (64kB)
```

- **log_temp_files** — このパラメータは、セッションの一時ファイルが削除されたときに postgresql.log にメッセージを送信します。このパラメータは、クエリが正常に完了した後にログを生成します。そのため、アクティブで長時間実行されるクエリのトラブルシューティングには役立たない可能性があります。

次の例は、クエリが正常に完了すると、エントリが postgresql.log ファイルに記録され、一時ファイルがクリーンアップされることを示しています。

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir** — この関数は RDS for PostgreSQL 13 以降で使用でき、現在の一時ファイルの使用状況を可視化できます。完了したクエリは、関数の結果には表示されません。次の例では、この関数の結果を表示できます。

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
--------------------------------------------------------------------------------
```

```
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

ファイル名には、一時ファイルを生成したセッションの処理 ID (PID) が含まれます。次の例のような、より高度なクエリでは、各 PID の一時ファイルの合計が実行されます。

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements** — pg_stat_statements パラメータを有効にすると、呼び出しごとの一時ファイルの平均使用量を表示できます。次の例に示すように、クエリの query_id を特定し、それを使用して一時ファイルの使用状況を調べることができます。

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                      388678
(1 row)
```

- **Performance Insights** — Performance Insights ダッシュボードで、temp_bytes と temp_files のメトリクスをオンにすると、一時ファイルの使用状況を確認できます。次に、これら両方のメトリクスの平均と、それらがクエリワークロードにどのように対応しているかを確認できます。Performance Insights 内のビューには、一時ファイルを生成しているクエリが具体的に表示されません。ただし、Performance Insights と `pg_ls_tmpdir` に示されるクエリを組み合わせると、クエリワークロードの変化をトラブルシューティング、分析、判断できます。

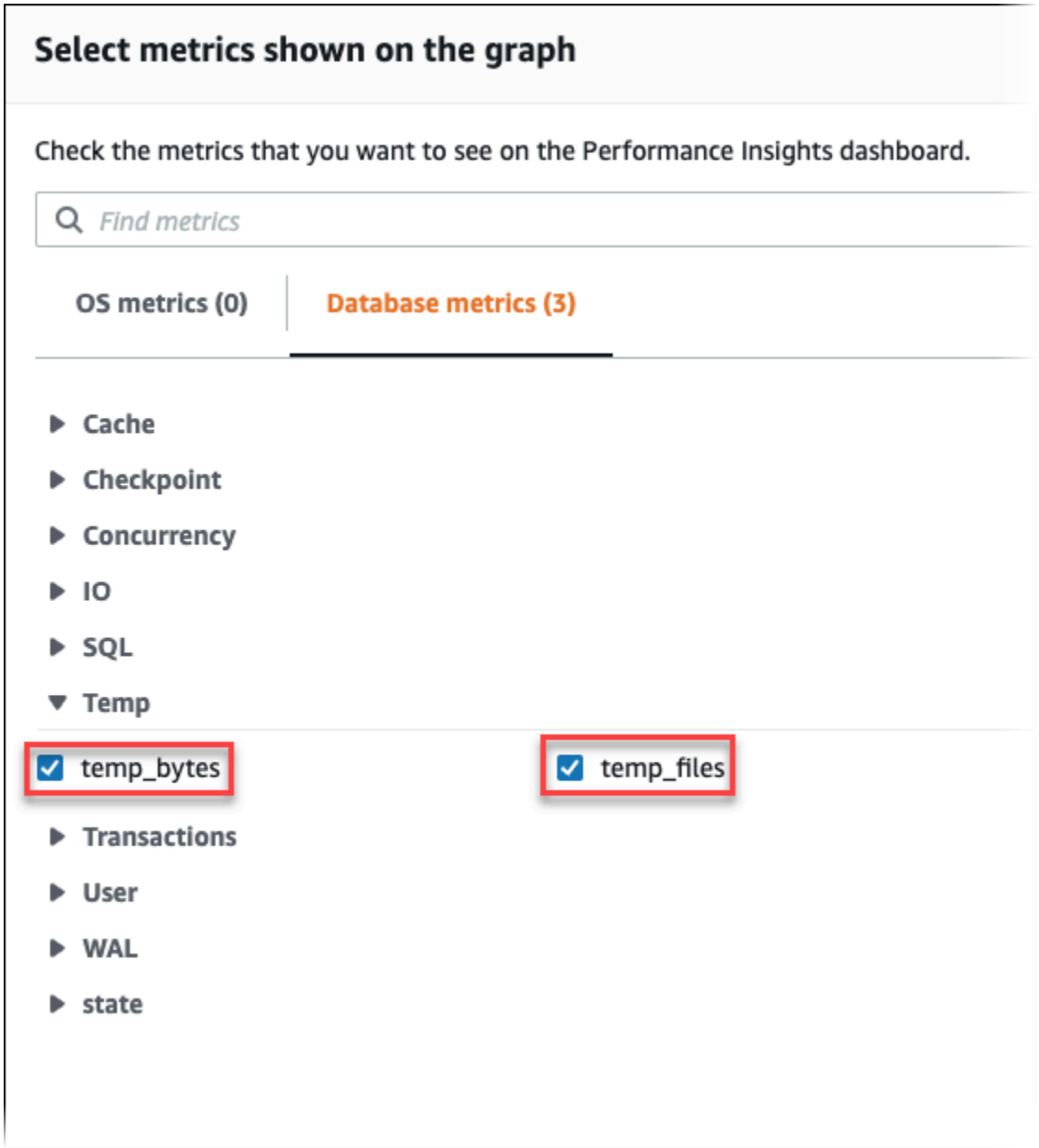
  Performance Insights を使用してメトリクスとクエリを分析する方法については、「Performance Insights ダッシュボードを使用してメトリクスを分析する」を参照してください。

  Performance Insights で一時ファイルの使用状況を表示する例については、「Performance Insights を使用した一時ファイルの使用状況の確認」を参照してください。

Performance Insights を使用した一時ファイルの使用状況の確認

Performance Insights で、temp_bytes と temp_files のメトリクスをオンにすると、一時ファイルの使用状況を確認できます。Performance Insights のビューには、一時ファイルを生成する特定のクエリは表示されませんが、Performance Insights を `pg_ls_tmpdir` に示されているクエリと組み合わせると、クエリワークロードの変更をトラブルシューティング、分析、判断できます。

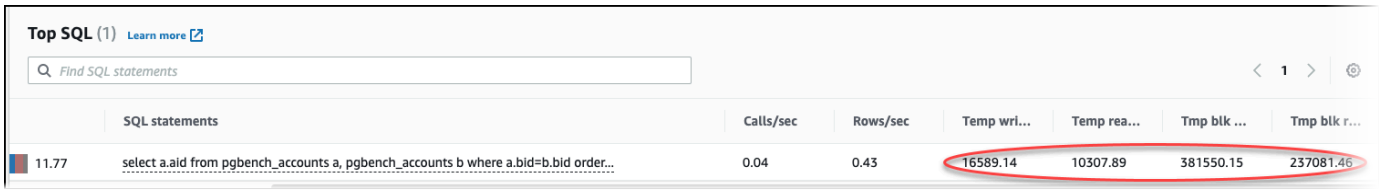
1. Performance Insights ダッシュボードで、[メトリクスを管理] を選択します。
2. 次の画像に示すように、[データベースメトリクス] を選択して、[temp_bytes] と [temp_files] を選択します。



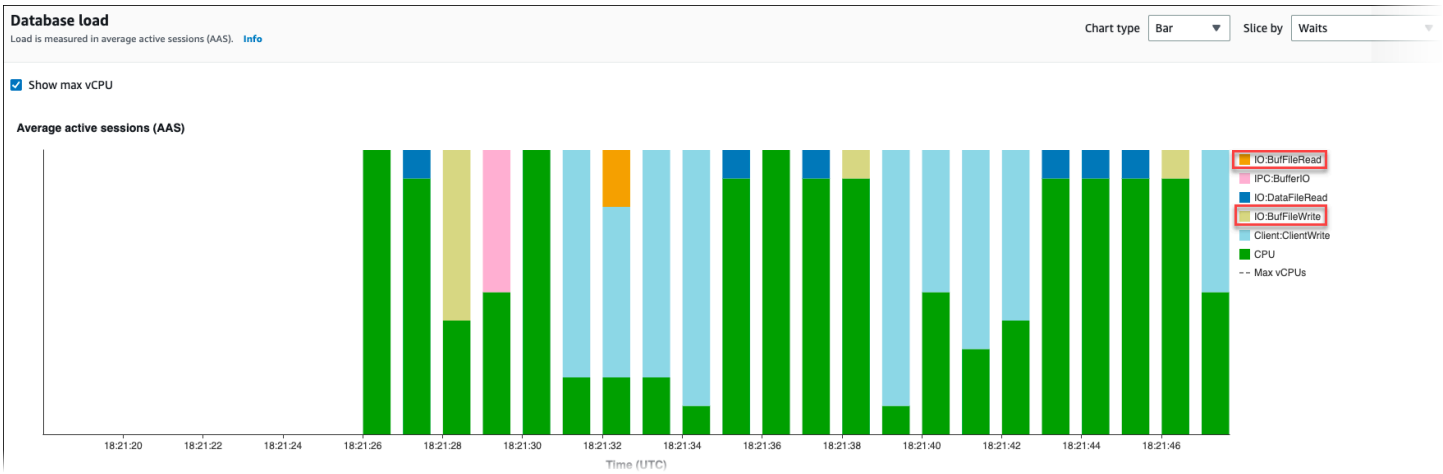
3. [トップ SQL] タブで、[設定] アイコンを選択します。
4. [設定] ウィンドウで、[トップ SQL] タブに以下の統計が表示されるようにして、[続行] を選択します。

   - Temp writes/sec
   - Temp reads/sec
   - Tmp blk write/call
   - Tmp blk read/call

5. 次の例に示すように、pg_ls_tmpdir で示したクエリと組み合わせると、この一時ファイルが抜き出されます。



IO:BufFileRead と IO:BufFileWrite イベントは、ワークロードの上位クエリが一時ファイルを頻繁に作成するときに発生します。Performance Insights を使用することで、「データベース負荷」と「上位 SQL」セクションの「平均アクティブセッション (AAS)」を参照して、IO:BufFileRead と IO:BufFileWrite を待機中の上位クエリの特定が可能になります。



Performance Insights を使用して上位クエリを分析して、待機イベント別にロードする方法については、「[トップ SQL] タブの概要」を参照してください テンポラリファイルの使用量と関連する待機イベントの増加の原因となるクエリを特定し、調整する必要があります。これらの待機イベントと修復方法の詳細については、「IO:BufFileRead および IO:BufFileWrite」「」を参照してください。

> **Note**
>
> work_mem パラメータは、ソート操作がメモリ不足になり、結果が一時ファイルに書き込まれるタイミングを制御します。このパラメータの設定をデフォルト値より高く変更しないことをお勧めします。変更すると、すべてのデータベースセッションでより多くのメモリを消費することになります。また、複雑な結合とソートを実行する単一セッションでは、それぞれの処理がメモリを消費する並列処理を実行できます。
> ベストプラクティスとして、複数の結合とソートを含む大規模なレポートがある場合は、SET work_mem コマンドを使用してこのパラメータをセッションレベルで設定します。

そうすれば、変更は現在のセッションにのみ適用され、値がグローバルに変更されることはありません。

pgBadger を使用した PostgreSQL でのログ分析

pgBadger などのログ分析ツールを使用して、PostgreSQL のログを分析できます。pgBadger のドキュメントでは %l パターン (セッションやプロセスに関するログの行) をプレフィックスに含める必要があると説明されています。ただし、最新の RDS `log_line_prefix` をパラメータとして pgBadger に渡すことでも、レポートが作成されます。

例えば、次のコマンドでは、pgBadger を使用して、2014-02-04 の日付の Amazon RDS for PostgreSQL のログファイルを適切にフォーマットします。

```
./pgbadger -f stderr -p '%t:%r:%u@%d:[%p]:' postgresql.log.2014-02-04-00
```

PostgreSQL をモニタリングするために PGSnapper を使用する

PGSnapper を使用すると、Amazon RDS for PostgreSQL のパフォーマンス関連の統計やメトリクスを定期的に収集することができます。詳細については、「PGSnapper を使用して Amazon RDS for PostgreSQL のパフォーマンスをモニタリングする」を参照してください。

RDS for PostgreSQL でのカスタムキャストの管理

PostgreSQL での型キャストは、あるデータ型から別のデータ型に値を変換するプロセスです。PostgreSQL には多くの一般的な変換用の組み込みキャストがありますが、カスタムキャストを作成して、特定の型変換の動作を定義することもできます。

キャストは、あるデータ型から別のデータ型への変換を実行する方法を指定します。例えば、テキスト `'123'` を整数 `123` に変換したり、数値 `45.67` をテキスト `'45.67'` に変換したりします。

PostgreSQL のキャストの概念と構文に関する包括的な情報については、「PostgreSQL CREATE CAST のドキュメント」を参照してください。

RDS for PostgreSQL バージョン 13.23、14.20、15.15、16.11、17.7、および 18.1 以降では、rds_casts 拡張機能を使用して組み込みタイプの追加のキャストをインストールできるだけでなく、カスタムタイプの独自のキャストを作成することもできます。

トピック

- rds_casts 拡張機能のインストールと使用
- サポートされているキャスト
- キャストの作成または削除
- 適切なコンテキスト戦略を使用してカスタムキャストを作成する

rds_casts 拡張機能のインストールと使用

`rds_casts` 拡張機能を作成するには、`rds_superuser` として RDS for PostgreSQL DB インスタンスに接続し、次のコマンドを実行します。

```
CREATE EXTENSION IF NOT EXISTS rds_casts;
```

サポートされているキャスト

カスタムキャストを使用する各データベースに拡張機能を作成します。拡張機能を作成した後、次のコマンドを使用して使用可能なすべてのキャストを表示します。

```
SELECT * FROM rds_casts.list_supported_casts();
```

この関数は、使用可能なキャストの組み合わせ (ソースタイプ、ターゲットタイプ、強制コンテキスト、キャスト関数) を一覧表示します。例えば、`text` から `numeric` への `implicit` キャストを作成する場合です。次のクエリを使用して、キャストを作成できるかどうかを確認できます。

```
SELECT * FROM rds_casts.list_supported_casts()
WHERE source_type = 'text' AND target_type = 'numeric';
 id | source_type | target_type |          qualified_function          |
 coercion_context
----+-------------+-------------+--------------------------------------
+------------------
 10 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | implicit
 11 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | assignment
 13 | text        | numeric     | rds_casts.rds_text_to_numeric_custom | explicit
 20 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | implicit
 21 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | assignment
 23 | text        | numeric     | rds_casts.rds_text_to_numeric_inout  | explicit
```

rds_casts 拡張機能は、キャストごとに 2 種類の変換関数を提供します。

- _inout 関数 - PostgreSQL の標準 I/O 変換メカニズムを使用し、INOUT メソッドで作成されたキャストと同じように動作します。
- _custom 関数 - 変換エラーを避けるために空の文字列を NULL 値に変換するなど、エッジケースを処理する拡張変換ロジックを提供します。

inout 関数は PostgreSQL のネイティブキャスト動作をレプリケートしますが、custom 関数は、空の文字列を整数に変換するなど、標準の INOUT キャストが対応できないシナリオを処理することでこの機能を拡張します。

キャストの作成または削除

次の 2 つの方法を使用して、サポートされているキャストを作成および削除できます。

キャストの作成

方法 1: ネイティブの CREATE CAST コマンドを使用する

```
CREATE CAST (text AS numeric)
WITH FUNCTION rds_casts.rds_text_to_numeric_custom
AS IMPLICIT;
```

方法 2: rds_casts.create_cast 関数を使用する

```
SELECT rds_casts.create_cast(10);
```

create_cast 関数は、list_supported_casts() 出力から ID を取得します。この方法はよりシンプルで、正しい関数とコンテキストの組み合わせを使用していることが確認されます。この ID は、異なる postgres バージョン間で同じままであることが保証されています。

キャストが正常に作成されたことを確認するには、pg_cast システムカタログをクエリします。

```
SELECT oid, castsource::regtype, casttarget::regtype, castfunc::regproc, castcontext,
 castmethod
FROM pg_cast
WHERE castsource = 'text'::regtype AND casttarget = 'numeric'::regtype;
  oid   | castsource | casttarget |              castfunc              | castcontext
 | castmethod
--------+------------+------------+------------------------------------+-------------
+------------
```

```
 356372 | text       | numeric    | rds_casts.rds_text_to_numeric_custom | i
 | f
```

castcontext 列には、EXPLICIT の場合は e、ASSIGNMENT の場合は a、IMPLICIT の場合は i が表示されます。

キャストの削除

方法 1: DROP CAST コマンドを使用する

```
DROP CAST IF EXISTS (text AS numeric);
```

方法 2: rds_casts.drop_cast 関数を使用する

```
SELECT rds_casts.drop_cast(10);
```

drop_cast 関数は、キャストの作成時と同じ ID を使用します。この方法では、対応する ID で作成されたキャストを正確に削除できます。

適切なコンテキスト戦略を使用してカスタムキャストを作成する

整数型に対して複数のキャストを作成する場合、すべてのキャストが IMPLICIT として作成されると、演算子のあいまいさのエラーが発生する可能性があります。次の例は、テキストから異なる整数幅に 2 つの暗黙的なキャストを作成することで、この問題を示しています。

```
-- Creating multiple IMPLICIT casts causes ambiguity
postgres=> CREATE CAST (text AS int4) WITH FUNCTION
 rds_casts.rds_text_to_int4_custom(text) AS IMPLICIT;
CREATE CAST
postgres=> CREATE CAST (text AS int8) WITH FUNCTION
 rds_casts.rds_text_to_int8_custom(text) AS IMPLICIT;
CREATE CAST

postgres=> CREATE TABLE test_cast(col int);
CREATE TABLE
postgres=> INSERT INTO test_cast VALUES ('123'::text);
INSERT 0 1
postgres=> SELECT * FROM test_cast WHERE col='123'::text;
ERROR:  operator is not unique: integer = text
LINE 1: SELECT * FROM test_cast WHERE col='123'::text;
```

```
                                                ^
HINT:  Could not choose a best candidate operator. You might need to add explicit type
 casts.
```

このエラーは、PostgreSQL が整数列とテキスト値を比較するときに使用する暗黙的なキャストを特定できないために発生します。int4 と int8 の両方の暗黙的なキャストが有効な候補であるため、あいまいさが生じます。

この演算子のあいまいさを回避するには、小さい整数幅には ASSIGNMENT コンテキストを使用し、大きい整数幅には IMPLICIT コンテキストを使用します。

```
-- Use ASSIGNMENT for smaller integer widths
CREATE CAST (text AS int2)
WITH FUNCTION rds_casts.rds_text_to_int2_custom(text)
AS ASSIGNMENT;

CREATE CAST (text AS int4)
WITH FUNCTION rds_casts.rds_text_to_int4_custom(text)
AS ASSIGNMENT;

-- Use IMPLICIT for larger integer widths
CREATE CAST (text AS int8)
WITH FUNCTION rds_casts.rds_text_to_int8_custom(text)
AS IMPLICIT;

postgres=> INSERT INTO test_cast VALUES ('123'::text);
INSERT 0 1
postgres=> SELECT * FROM test_cast WHERE col='123'::text;
 col
-----
 123
(1 row)
```

この戦略では、int8 キャストのみが暗黙的であるため、PostgreSQL はどのキャストを使用するかを明確に判断できます。

RDS for PostgreSQL での並列クエリのベストプラクティス

並列クエリ実行は、PostgreSQL の機能であり、単一の SQL クエリをより小さなタスクに分割し、複数のバックグラウンドワーカープロセスによって同時に処理することを可能にします。PostgreSQL は、クエリ全体を単一のバックエンドプロセスで実行する代わりに、スキャン、結

合、集約、ソートなどのクエリの一部を複数の CPU コアに分散させることができます。リーダープロセスは、この実行を調整し、並列ワーカーから結果を収集します。

ただし、ほとんどの本稼働ワークロード、特に高同時実行 OLTP システムでは、自動並列クエリ実行を無効にすることをお勧めします。並列処理は、分析またはレポートワークロードの大規模なデータセットに対するクエリを高速化できますが、負荷の高い本番環境では、多くの場合、メリットを上回る重大なリスクをもたらします。

並列実行では、大きなオーバーヘッドも発生します。各並列ワーカーは完全な PostgreSQL バックエンドプロセスであり、プロセスのフォーク (メモリ構造のコピーとプロセス状態の初期化) と認証 (`max_connections` 制限の接続スロットの消費) が必要です。各ワーカーは、ソートやハッシュ操作のための `work_mem` を含め、独自のメモリも消費します。クエリごとに複数のワーカーがある場合、メモリ使用量は急速に増加します (例: 4 つのワーカー × 64MB `work_mem` = クエリあたり 256MB)。その結果、並列クエリは単一プロセスクエリよりもかなり多くのシステムリソースを消費する可能性があります。正しくチューニングされていない場合は、CPU 飽和 (複数のワーカーが使用可能な処理能力を圧倒する)、コンテキスト切り替えの増加 (オペレーティングシステムが多数のワーカープロセスを頻繁に切り替えるため、オーバーヘッドが増加しスループットが低下する)、または接続の枯渇 (各並列ワーカーが接続スロットを消費するため、4 つのワーカーを含む単一のクエリでは、リーダー 1 つとワーカー 4 つの合計 5 つの接続が使用され、高い同時実行性のもとでは接続プールがすぐに枯渇し、新しいクライアント接続が妨げられ、アプリケーション障害が発生する) につながる可能性があります。これらの問題は、複数のクエリが同時に並列実行を試みる可能性がある同時実行性の高いワークロードでは特に深刻です。

PostgreSQL は、コスト見積もりに基づいて並列処理を使用するかどうかを決定します。場合によっては、プランナーは、実際には理想的でない場合でも、より安価に見える場合、並列プランに自動的に切り替えることがあります。これは、インデックス統計が古くなった場合や、肥大化してシーケンシャルスキャンがインデックス検索よりも魅力的に見える場合に発生する可能性があります。この動作により、自動並列プランによってクエリのパフォーマンスやシステムの安定性にリグレッションを引き起こすことがあります。

RDS for PostgreSQL で並列クエリを最大限に活用するには、ワークロードに基づいて並列クエリをテストおよびチューニングし、システムへの影響をモニタリングし、自動並列プラン選択を無効にして、クエリレベルで制御することが重要です。

設定パラメータ

PostgreSQL は、複数のパラメータを使用して並列クエリの動作と可用性を制御します。これらを理解して調整することは、予測可能なパフォーマンスを達成するために不可欠です。

パラメータ	説明	デフォルト
max_parallel_workers	同時に実行できるバックグラウンドワーカープロセスの最大数	GREATEST($DBInstanceVCPU/2,8)
max_parallel_workers_per_gather	クエリプランノードあたりのワーカーの最大数 (例: Gather あたり)	2
parallel_setup_cost	並列クエリインフラストラクチャを開始するためのプランナーコストの追加	1,000
parallel_tuple_cost	並列モードで処理されたタプルあたりのコスト (プランナーの決定に影響する)	0.1
force_parallel_mode	プランナーに並列プランのテストを強制する (off、on、regress)	off

主な考慮事項

- max_parallel_workers は、並列ワーカーの合計プールを制御します。設定が低すぎると、一部のクエリが直列実行にフォールバックする可能性があります。
- max_parallel_workers_per_gather は、単一のクエリで使用できるワーカーの数に影響します。値を大きくすると、同時実行数は増加しますが、リソースの使用量も増加します。
- parallel_setup_cost と parallel_tuple_cost は、プランナーのコストモデルに影響を与えます。これらの値を下げると、並列プランを選択する可能性が高くなります。
- force_parallel_mode はテストに便利ですが、必要でない限り本番環境では使用しないでください。

> **Note**
>
> max_parallel_workers パラメータのデフォルト値は、式 GREATEST($DBInstanceVCPU/2, 8) を使用してインスタンスサイズに基づいて動的に計算されます。つまり、DB インスタンスをより多くの vCPU でより大きなコンピューティングサイズにスケールすると、使用可能な並列ワーカーの最大数が自動的に増加します。その結果、以前に連続して実行されたクエリや、並列処理が制限されたクエリでは、スケールアップ操作後に突然、より多くの並列ワーカーが使用され、接続使用量、CPU 使用率、メモリ消費量が予期せず増加する可能性があります。コンピューティングスケーリングイベントの後に並列クエリの動作をモニタリングし、予測可能なリソース使用量を維持するために必要に応じて max_parallel_workers_per_gather を調整することが重要です。

並列クエリの使用状況を特定する

クエリは、データの分布または統計に基づいて並列プランに切り替わる場合があります。例えば、次のようになります。

```
SELECT count(*) FROM customers WHERE last_login < now() - interval '6 months';
```

このクエリは、最近のデータにインデックスを使用しますが、履歴データに対しては並列シーケンシャルスキャンに切り替える可能性があります。

auto_explain モジュールをロードすることでクエリ実行計画を記録できます。詳細については、AWS ナレッジセンターの「クエリ実行計画のログ記録」をご覧ください。

CloudWatch Database Insights を使用すると、並列クエリに関連する待機イベントをモニタリングできます。並列クエリに関連する待機イベントの詳細については、「IPC:parallel 待機イベント」を参照してください。

PostgreSQL バージョン 18 以降では、pg_stat_database と pg_stat_statements の新しい列を使用して、並列ワーカーのアクティビティをモニタリングできます。

- parallel_workers_to_launch: 起動が予定されている並列ワーカーの数
- parallel_workers_launched: 実際に起動された並列ワーカーの数

これらのメトリクスは、計画された並列処理と実際の並列処理との間の差異を特定するのに役立ち、リソースの制約や設定の問題を示している可能性があります。次のクエリを使用して、並列実行をモニタリングします。

データベースレベルの並列ワーカーメトリクスの場合。

```
SELECT datname, parallel_workers_to_launch, parallel_workers_launched
FROM pg_stat_database
WHERE datname = current_database();
```

クエリレベルの並列ワーカーメトリクスの場合

```
SELECT query, parallel_workers_to_launch, parallel_workers_launched
FROM pg_stat_statements
ORDER BY parallel_workers_launched;
```

並列処理を制御する方法

クエリの並列処理を制御するにはいくつかの方法があり、それぞれがさまざまなシナリオと要件に合わせて設計されています。

自動並列処理をグローバルに無効にするには、パラメータグループを変更して、次のように設定します。

```
max_parallel_workers_per_gather = 0;
```

永続的なユーザー固有の設定の場合、ALTER ROLE コマンドを使用すると、特定のユーザーの今後のすべてのセッションに適用されるパラメータを設定できます。

例えば、次のようになります。

`ALTER ROLE username SET max_parallel_workers_per_gather = 4;` は、このユーザーがデータベースに接続するたびに、セッションが必要に応じてこの並列ワーカー設定を使用するようにします。

セッションレベルの制御は、現在のデータベースセッションの期間中にパラメータを変更する SET コマンドを使用して実現できます。これは、他のユーザーや今後のセッションに影響を与えることなく、設定を一時的に調整する必要がある場合に特に便利です。設定すると、これらのパラメータは明示的にリセットされるまで、またはセッションが終了するまで有効です。コマンドは簡単です。

```
SET max_parallel_workers_per_gather = 4;
-- Run your queries
RESET max_parallel_workers_per_gather;
```

さらに詳細な制御を行うには、SET LOCAL を使用すると、単一のトランザクションのパラメータを変更できます。これは、トランザクション内の特定のクエリセットの設定を調整する必要がある場合に最適です。その後、設定は自動的に以前の値に戻ります。このアプローチは、同じセッション内の他のオペレーションへの意図しない影響を防ぐのに役立ちます。

並列クエリ動作の診断

EXPLAIN（ANALYZE, VERBOSE）を使用して、クエリが並列実行を使用したかどうかを確認します。

- Gather、Gather Merge、Parallel Seq Scan などのノードを探します。
- 並列処理ありの場合と並列処理なしの場合の計画を比較します。

比較のために並列処理を一時的に無効にするには。

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
RESET max_parallel_workers_per_gather;
```

RDS for PostgreSQL で REPLICA IDENTITY FULL のパフォーマンスの問題を回避する

PostgreSQL 論理レプリケーションでは、サブスクライバーが更新または削除する正しい行を見つけることができるように、発行された各テーブルにレプリカ ID が必要です。デフォルトでは、プライマリキーがレプリカ ID として機能します。テーブルにプライマリキーまたは適切な一意のインデックスがない場合、レプリカ ID を FULL に設定することで、PostgreSQL は行全体をキーとして使用します。

REPLICA IDENTITY FULL はプライマリキーなしでテーブルをレプリケートするという即時の問題を解決しますが、パブリッシャーとサブスクライバーの両方で重大なパフォーマンスの問題が発生する可能性があります。これらの影響を理解することは、ブルー/グリーンデプロイなど、内部で論理レプリケーションに依存する機能を含め、RDS for PostgreSQL で論理レプリケーションを使用するすべての人にとって重要です。

REPLICA IDENTITY FULL が問題を引き起こす理由

パブリッシャーの WAL ボリュームの増加

REPLICA IDENTITY 設定は、更新または削除された行を識別するために PostgreSQL が口グ先行書き込み (WAL) に書き込む情報を制御します。デフォルトのレプリカ ID (プライマリキー) では、キー列のみが古い行 ID として記録されます。FULL を使用すると、PostgreSQL は UPDATE および DELETE のたびに、すべての列の古い値を記録します。これにはいくつかの影響があります。

- WAL のサイズが大幅に増加します。更新の場合、各列の古い値と新しい値の両方が記録されるため、各 WAL レコードのサイズは約 2 倍になります。テーブルに TOAST を使用して保存された大きな値が含まれている場合、更新によって変更されていない場合でも TOAST 化された値を取得して WAL に書き込む必要があるため、増加がさらに大きくなる可能性があります。
- パブリッシャーの I/O および CPU 使用率が高くなっています。追加の WAL 書き込みは、特に書き込み負荷の高いワークロードの場合、より多くのディスク I/O 帯域幅と CPU サイクルを消費します。
- さらに多くのデータがサブスクライバーに送信されます。パブリッシャーは、ネットワーク経由でより大きな WAL レコードを各サブスクライバーに送信する必要があり、その結果、帯域幅の消費量が増加します。

サブスクライバーでの行検索の遅延

サブスクライバーが UPDATE または DELETE ログレコードを受信すると、テーブルのローカルコピーで一致する行を見つける必要があります。REPLICA IDENTITY FULL を使用すると、サブスクライバーは古い行イメージのすべての列値に一致する行を検索します。

PostgreSQL がこの検索を実行する方法は、PostgreSQL メジャーバージョンによって異なります。

- PostgreSQL 16 より前: テーブルにプライマリキーがなく、明示的に設定されたレプリカ ID インデックスがない場合、サブスクライバーは UPDATE または DELETE オペレーションごとにテーブル全体のシーケンシャルスキャンを実行します。大きなテーブルでは、これにより適用パフォーマンスが極めて低下します。
- PostgreSQL 16 以降: サブスクライバーは、そのインデックスがレプリカ ID として明示的に設定されていない場合でも、行検索に Btree インデックスまたはハッシュインデックスを使用できます。ただし、サブスクライバーはどのインデックスが最も効率的かを評価しません。バージョン 16 以降、PostgreSQL は検出した最初の適切なインデックスを選択し、ユーザーはこの選択を制御できません。選択したインデックスの選択性が低い場合 (ブール値列やステータス列のインデッ

クスなど)、行の検索はシーケンシャルスキャンとほぼ同じくらい遅くなる可能性があります。このため、REPLICA IDENTITY FULL を使用した暗黙的なインデックス選択に依存することは信頼性が低く、推奨される設定ではなくフォールバックと考える必要があります。

REPLICA IDENTITY FULL がレプリケーション遅延を引き起こす方法

上記の 2 つの問題、つまりパブリッシャーの WAL が大きいこととサブスクライバーの行検索が遅いことが組み合わさって、レプリケーションの遅延が発生します。

デフォルトでは、PostgreSQL の論理レプリケーションは、サブスクリプションごとに単一の適用ワーカープロセスを使用して、パブリッシャーから変更を受け取り、サブスクライバーのテーブルに適用します。適用ワーカーは、変更をコミット順に、一度に 1 行ずつ連続して処理します。つまり、サブスクライバーのスループットは、個々の変更を適用する速度によって制限されます。

適切なインデックスのないテーブルに REPLICA IDENTITY FULL が設定されている場合、UPDATE および DELETE のたびに、一致する行を見つけるためにテーブル全体のシーケンシャルスキャンが必要です。テーブルに数百万行ある場合、これらの各オペレーションには数秒以上かかることがあります。その結果、連鎖的な問題が発生します。

1. パブリッシャーは、サブスクライバーが適用できるよりも速いペースで変更を生成します。パブリッシャーの書き込みワークロードは通常の速度で継続されますが、サブスクライバーの適用ワーカーは、各行の検索においてシーケンシャルスキャンまたは選択性の低いインデックスによってボトルネックが発生しています。
2. WAL はパブリッシャーに蓄積され、ストレージを使い果たす可能性があります。PostgreSQL は、サブスクライバーが適用したことを確認するまで WAL セグメントを再利用することはできません。サブスクライバーがさらに遅れると、パブリッシャーは WAL をディスクに蓄積します。RDS for PostgreSQL では、これは CloudWatch で OldestReplicationSlotLag が増加する形で現れます。重大なケースでは、使用可能なすべてのストレージを消費し、パブリッシャーが書き込みの受け入れを停止する可能性があります。
3. 遅延は自己強化的です。サブスクライバーが遅れると、サブスクライバーのテーブルはレプリケートされた挿入によって肥大化し、各シーケンシャルスキャンはさらに遅くなります。介入しない場合、遅延は制限なしで大きくなります。

この問題は、頻繁に UPDATE または DELETE オペレーションを受け取るテーブルでは特に深刻です。INSERT オペレーションは、サブスクライバーの行検索を必要としないため、影響を受けません。

> **Note**
>
> PostgreSQL 16 以降、適用ワーカーは大規模なストリーミングトランザクションに並列適用を使用できるようになり、スループットの向上に役立ちます。ただし、インデックスを使用しない `REPLICA IDENTITY FULL` では、基本的な行検索のボトルネックは残ります。これは、個々の行を特定するためにスキャンが必要となるためです。

ブルー/グリーンデプロイへの影響

Amazon RDS のブルー/グリーンデプロイでは、内部的に論理レプリケーションを使用して、データベースごとに 1 つのサブスクリプションを設定することで、グリーン環境とブルー環境の同期を維持します。グリーン環境の論理的なレプリケーション適用プロセスは、シングルスレッドです。単一の適用ワーカープロセスは、ブルー環境からすべての変更を受け取り、コミット順に一度に 1 つずつ適用します。ブルー/グリーンレプリケーションパスに並列適用はありません。

このシングルスレッド設計は、グリーン環境がブルー環境に対応する能力が、適用ワーカーが個々の変更を処理できる速度に完全に依存することを意味します。テーブルがプライマリキーまたは適切なインデックスなしで `REPLICA IDENTITY FULL` を使用する場合、適用ワーカーへの影響は PostgreSQL バージョンによって異なります。16 より前のバージョンでは、それらのテーブルに対するすべての `UPDATE` および `DELETE` で、適用ワーカーはテーブル全体のシーケンシャルスキャンを実行して一致する行を見つける必要がありました。バージョン 16 以降では、PostgreSQL は適切なインデックスが利用可能であればそれを使用しますが、対象となるインデックスが存在しない場合は、適用ワーカーはシーケンシャルスキャンにフォールバックします。適用ワーカーが 1 行の大きなテーブルをスキャンしている間、すべてのテーブルにわたるその他の保留中の変更はすべてキューに入れられて待機します。

ブルー/グリーンデプロイへの影響は重大です。

- レプリケーションの遅延が継続的に増加します。ブルー環境が単一の適用ワーカーが処理できるよりも速く書き込みトラフィックを生成する場合、グリーン環境はさらに遅れをとるようになります。適用ワーカーはシングルスレッドであるため、キャッチアップを並列化する方法はありません。
- スイッチオーバーはブロックできます。ブルー/グリーンスイッチオーバーでは、グリーン環境をブルー環境と完全に同期する必要があります。レプリケーション遅延が大きすぎる場合、タイムアウト期間内にスイッチオーバーが完了しない場合があります。
- グリーン環境がいつまでもキャッチアップできない可能性があります。`REPLICA IDENTITY FULL` を使用して、インデックスのない大きなテーブルで書き込み負荷の高いワークロードを実行

する場合、適用レートが非常に遅くなり、グリーン環境が永続的に遅れる可能性があるため、最初にレプリカ ID 設定を解決しない限り、スイッチオーバーができない可能性があります。

- WAL はブルー環境に蓄積されます。グリーン環境が遅れている間、ブルー環境はレプリケーションスロットの WAL セグメントを保持します。これにより、ブルー (本番稼働) 環境でのストレージ使用量が増加し、本番稼働のパフォーマンスに影響する可能性があります。

これらの問題を回避するために、ブルー/グリーンデプロイを作成する前に、すべてのテーブルにプライマリキー、または `ALTER TABLE ... REPLICA IDENTITY USING INDEX` を使用してレプリカ ID として明示的に設定された適切な一意のインデックスが存在することを確認します。PostgreSQL 16 以降では、サブスクライバーが選択性の低いインデックスを選択したり、シーケンシャルスキャンにフォールバックしたりする可能性があるため、暗黙的なインデックス選択を伴う `REPLICA IDENTITY FULL` に依存しないでください。代表的な書き込みワークロードを使用してデプロイをテストし、グリーン環境が維持できることを確認します。

ブルー/グリーンデプロイの制限事項の詳細については、「the section called “制約事項と考慮事項”」を参照してください。ベストプラクティスについては、the section called “RDS for PostgreSQL のブルー/グリーンデプロイのベストプラクティス”を参照してください。

REPLICA IDENTITY FULL を使用してテーブルを識別する方法

次のクエリを実行して、`REPLICA IDENTITY FULL` を持つすべてのテーブルを検索します。

```
SELECT n.nspname AS schema, c.relname AS table_name, c.relreplident
FROM pg_catalog.pg_class c
JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
WHERE c.relkind = 'r'
  AND c.relreplident = 'f'
  AND n.nspname NOT IN ('pg_catalog', 'information_schema')
ORDER BY n.nspname, c.relname;
```

`relreplident` 列の値は次のとおりです。

- `d` - デフォルト (プライマリキー)
- `n` - なし
- `f` - full (行全体)
- `i` - 特定のインデックス

回避策とベストプラクティス

可能な限りプライマリキーを追加する

最も効果的な解決策は、プライマリキーがないテーブルにプライマリキーを追加することです。プライマリキーが存在する場合、PostgreSQL はそれをデフォルトのレプリカ ID として使用します。これにより、サブスクライバーでの効率的な行検索が可能になり、パブリッシャーの WAL のオーバーヘッドが最小限に抑えられます。

```
ALTER TABLE my_table ADD COLUMN id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY;
```

> **Important**
>
> このステートメントは、デフォルト値の式が揮発性の `nextval()` を使用しているため、`ACCESS EXCLUSIVE` ロックを取得し、テーブル全体を書き換えます。テーブルへのすべての読み取りと書き込みは、書き換えの間ブロックされます。大きなテーブルの場合、これは大幅なダウンタイムを引き起こす可能性があります。この変更はメンテナンスウィンドウ中に計画するか、最初に列を null 許容として作成し、その後、制約を別のステップでバックフィルして追加するなどの代替アプローチを検討してください。

アプリケーションの制約によりプライマリキーを追加できない場合は、`NOT NULL` 列のセットに一意のインデックスを追加し、それをレプリカ ID として設定することを検討してください。

```
CREATE UNIQUE INDEX my_table_replica_idx ON my_table (col1, col2);
ALTER TABLE my_table REPLICA IDENTITY USING INDEX my_table_replica_idx;
```

> **Note**
>
> インデックスの構築中に書き込みがブロックされないようにするには、CONCURRENTLY 句 `CREATE UNIQUE INDEX CONCURRENTLY my_table_replica_idx ON my_table (col1, col2);` を使用します。

> **Note**
>
> レプリカ ID に使用されるインデックスは一意でなければならず、部分的でなく、遅延可能でない、NOT NULL 制約のある列のみを含める必要があります。

暗黙的なインデックス選択に依存しない (PostgreSQL 16 以降)

PostgreSQL 16 以降、サブスクライバーの適用ワーカーは、レプリカ ID が FULL に設定されている場合、レプリカ ID として明示的に設定されていない場合でも、行検索に Btree インデックスまたはハッシュインデックスを使用できます。これにより、場合によってはシーケンシャルスキャンが防止されますが、この暗黙的な動作に依存することは、以下の理由からアンチパターンと言えます。

- どのインデックスが選択されるか制御できません。PostgreSQL は、最も選択的または効率的なインデックスではなく、カタログ順で検出される最初の適格なインデックスを選択します。テーブルに複数の適格なインデックスがある場合、選択したインデックスの選択性が低くなり、検索パフォーマンスが低下する可能性があります。
- この動作は脆弱です。インデックスを追加、削除、または再構築すると、適用ワーカーが使用するインデックスが変更され、レプリケーションで予期しないパフォーマンスのリグレッションが発生する可能性があります。
- 根本的な問題を覆い隠します。プライマリキーまたは明示的なレプリカ ID を持たないテーブルは、論理レプリケーションにおいて本質的にリスクがあります。暗黙的なインデックス選択に依存することは、問題を解決するのではなく、問題を先送りするだけです。

代わりに、レプリケートされたすべてのテーブルのレプリカ ID を明示的に設定します。

- 最適なオプション: プライマリキーを追加します。これは、最も信頼性が高く効率的なレプリカ IDです。
- 代替方法: ALTER TABLE ... REPLICA IDENTITY USING INDEX を使用して、NOT NULL 列のみを持つ部分的でない、遅延制約でもない、特定の一意インデックスを指定します。これにより、行の識別に使用される列を明示的に制御できます。

REPLICA IDENTITY FULL は、どちらのオプションも実現できないテーブルのみに使用してください。また、パフォーマンスは直接制御できない要因に依存することを理解します。

レプリケーション遅延のモニタリング

REPLICA IDENTITY FULL を使用する場合は、レプリケーションの遅延を注意深くモニタリングして、サブスクライバーのスローダウンが深刻化する前に検出します。

パブリッシャー側で、現在の WAL 位置とサブスクライバーが確認した位置の間の遅延を確認します。

```
SELECT slot_name, confirmed_flush_lsn, pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lag_bytes
FROM pg_replication_slots
WHERE slot_type = 'logical';
```

lag_bytes の値が着実に増加している場合、サブスクライバーが遅れていることを示しています。pg_stat_replication_slots ビューには、各レプリケーションスロットの使用状況に関する追加の統計が表示されます。

サブスクライバー側では、pg_stat_subscription ビューには、最後に受信および報告された WAL ロケーションを含む各適用ワーカーの状態が表示されます。

```
SELECT subname, received_lsn, latest_end_lsn,
       last_msg_send_time, last_msg_receipt_time
FROM pg_stat_subscription;
```

> **Note**
>
> PostgreSQL 16 以降では、worker_type を選択することで、メインの適用ワーカーと並列適用ワーカーを区別することもできます。

received_lsn と latest_end_lsn の間に大きなギャップがある場合、または last_msg_send_time のタイムスタンプが古い場合は、適用ワーカーが処理に追いついていないことを示している可能性があります。pg_stat_subscription_stats ビューでは、遅延の原因となる可能性のあるエラーや競合も追跡されます。

RDS for PostgreSQL では、元も遅延しているレプリケーションスロットの遅延をバイト単位で追跡する CloudWatch メトリクス OldestReplicationSlotLag をモニタリングすることもできます。値の増加は、レプリケーション遅延の早期警告サインです。

適用中に最適でないインデックスを使用している可能性のあるテーブルを確認する

サブスクライバーでは、適用ワーカーが過剰なヒープ読み取りを実行しているテーブルを特定できます。これは、適用中にテーブルに行検索の効率的なインデックスがないことを示している可能性があります。サブスクライバーに対して次のクエリを実行します。

```
SELECT relname, heap_blks_read, heap_blks_hit,
       idx_blks_read, idx_blks_hit,
       heap_blks_read + heap_blks_hit AS total_heap_access
FROM pg_statio_user_tables
WHERE heap_blks_read > 0
ORDER BY heap_blks_read DESC
LIMIT 10;
```

heap_blks_read に比べて idx_blks_read の値が高いテーブルは、適用ワーカーが UPDATE および DELETE オペレーションのために行検索する際に、効率的なインデックスを使用していないことを示している可能性があります。これは、REPLICA IDENTITY FULL を使用している場合に発生するレプリケーション遅延の一般的な原因です。

> **Note**
>
> このクエリを実行するには、サブスクライバーで track_counts パラメータが有効になっている必要があります。このパラメータはデフォルトでオンになっています。

REPLICA IDENTITY FULL が必要かどうかを評価する

REPLICA IDENTITY FULL を設定する前に、本当に必要かどうかを検討してください。これを使用する一般的な理由は次のとおりです。

- テーブルにプライマリキーまたは一意のインデックスがありません。
- 変更データキャプチャ (CDC) コンシューマーには、変更前の行の完全なイメージが必要です。
- これらの列を変更しない更新の場合、レプリケーションイベントに TOAST 化された列値を含める必要があります。

プライマリキーがないことが唯一の理由がであれば、プライマリキーの追加がほとんどの場合良い方法です。CDC の完全な前イメージが必要な場合は、CDC コンシューマーが外部で状態を維持することで完全な行を再構築できるかどうかを検討してください。これにより、REPLICA IDENTITY FULL の WAL とサブスクライバーのオーバーヘッドを回避できます。

レコメンデーションの概要

シナリオ	推奨事項
テーブルにプライマリキーがある	デフォルトのレプリカ ID を使用する (アクションは不要)
テーブルに一意の NOT NULL インデックスがある	`ALTER TABLE ... REPLICA IDENTITY USING INDEX` を使用して、そのインデックスをレプリカ ID として設定する
テーブルに適切なキーがない (PostgreSQL 16 以降)	プライマリキーまたは一意のインデックスを追加します。暗黙的なインデックス選択での `REPLICA IDENTITY FULL` の使用は信頼性が低く、最後の手段にする必要がある
テーブルに適切なキーがない (PostgreSQL 16 より前)	プライマリキーまたは一意のインデックスを追加する。可能であれば `REPLICA IDENTITY FULL` は避ける
大きな列/TOAST 化された列を含む書き込み負荷の高いワークロード	WAL ボリューム増幅のため `REPLICA IDENTITY FULL` を回避する

RDSfor PostgreSQL の一般的な PostgreSQL パフォーマンス問題の初期トラブルシューティング

このガイドでは、RDS for PostgreSQL データベースに影響する 4 つの最も一般的なパフォーマンスの問題について説明します。テーブルとインデックスの肥大化、並列クエリのリソースの枯渇、接続と認証の負荷増大、autovacuum の調整です。パフォーマンスの低下が発生した場合、より詳細な調査を開始する前に、このガイドを初期診断チェックリストとして活用してください。

各セクションでは、観察される可能性のある症状について説明し、根本原因を確認するための診断クエリを提供し、特定の修復手順を推奨します。

> ⚠ 「何も変更していない」場合のパフォーマンスのリグレッションについて
>
> PostgreSQL ワークロードは、数週間または数か月間問題なく実行された後、アプリケーションコードとクエリパターンに変更がないにもかかわらず、突然パフォーマンスが低下す

ることがよくあります。これは、データベース環境が真に静的ではないために発生します。いくつかの目に見えない要因が時間の経過とともに変化し、計画の変更やリソースの競合を引き起こす可能性があります。

- 肥大の蓄積はワークロードの変化を意味します。PostgreSQL のマルチバージョン同時実行制御 (MVCC) は、autovacuum がそれらを再利用するまで古い行バージョンを保持します。デッドタプルが autovacuum が処理できるよりも速く蓄積されると、テーブルとインデックスが物理的に大きくなります。その後、クエリプランナーは、テーブルサイズが大きくなるにつれてコスト見積もりがシフトするため、効率的なインデックススキャンからシーケンシャルスキャンに切り替える可能性があります。SQL は変更されていませんが、プランナーが参照するデータが変更されています。
- 新しいパラメータ値は、ワークロードの変更を意味します。ある範囲の値に対して優れたパフォーマンスを発揮するパラメータ化されたクエリでも、アプリケーションが別の範囲の使用を開始すると、パフォーマンスが低下する可能性があります。PostgreSQL は、新しい範囲内のデータスキューを考慮しない汎用実行プランを再利用したり、プランナーの統計がそれらの値の分布を正確に反映しない場合があります。肥大化も存在する場合、影響は複雑になります。最適ではない計画では、より多くのデッドデータをスキャンすることになります。
- autovacuum が実行されても、統計は古くなる可能性があります。autovacuum は、データ分散が実質的に変更されたかどうかではなく、挿入または更新された行数に基づいて `ANALYZE` をトリガーします。アプリケーションが別の値範囲または時間枠のクエリに移行した場合、autovacuum が最近実行された場合でも、プランナーのコスト見積もりが不正確になる可能性があります。
- データベース全体の成長は、ワークロードの変化を意味します。テーブルが時間の経過とともに増加すると、クエリがスキャンする必要があるデータページの量が増えます。小さいテーブルで正常に実行されたクエリは、クエリロジックとインデックスが変更されていない場合でも、テーブルサイズが大きくなるにつれてレイテンシーが発生する可能性があります。`FreeStorageSpace` をモニタリングして、ストレージの増加傾向を追跡します。

「何も変更していない」場合のパフォーマンスリグレッションを調査する際には、肥大の蓄積、新しいパラメータ値の範囲、データベース全体の成長、古くなった統計などを根本原因として最も可能性の高いものとして考慮します。このガイドの診断手順を使用して、どの要素が適用されるかを確認します。
詳細については次を参照してください:

- Amazon RDS および Amazon Aurora での PostgreSQL データベースのメンテナンスアクティビティ (AWS 規範ガイダンス)
- PostgreSQL クエリのパフォーマンスの最適化 (AWS 規範ガイダンス)
- Amazon RDS および Amazon Aurora での PostgreSQL パラメータの調整
- Amazon RDS インスタンスレベルのメトリクス (ストレージの増加傾向について FreeStorageSpace をモニタリング)

クイック診断チェックリスト

パフォーマンスの問題を最初に調査するときは、次の順序付けられたトリアージステップを使用します。

1. をチェックします。。**pg_stat_activity**接続数、アイドル状態のトランザクションセッション、長時間実行されるクエリを確認します。詳細については、「RDS for PostgreSQL チューニングの基本概念」を参照してください。
2. 肥大化がないか確認します。n_dead_tup で高い pg_stat_user_tables を探し、正確な測定に pgstattuple を使用することを検討してください。詳細については、「pg_repack を使用したテーブルからの肥大化の削除」を参照してください。
3. をチェックします。。**pg_stat_user_tables**n_dead_tup の値が高いもの、および last_autovacuum のタイムスタンプが古いものを探します。詳細については、「Amazon RDS での PostgreSQL autovacuum の使用」を参照してください。
4. スロークエリの **EXPLAIN ANALYZE** を確認します。並列プランや大きなテーブルに対するシーケンシャルスキャンを探します。詳細については、「RDS for PostgreSQL での並列クエリのベストプラクティス」を参照してください。
5. CloudWatch と Performance Insights のメトリクスを確認します。CPU 使用率、接続数、IOPS、解放可能なメモリを確認します。詳細については、「Amazon RDS のモニタリング」を参照してください。一般的な待機イベントと修正アクションについては、「RDS for PostgreSQL 待機イベント」を参照してください。
6. DB パラメータグループを確認します。max_parallel_workers_per_gather および autovacuum の設定を確認します。詳細については、「Amazon RDS および Amazon Aurora での PostgreSQL パラメータの調整」を参照してください。

テーブルとインデックスの肥大化

テーブルとインデックスの肥大化は、autovacuum がテーブルを再利用できるよりも速いペースで、デッドタプルがテーブルに蓄積されたときに発生します。これにより、時間の経過とともに、クエリのパフォーマンスが徐々に低下し、ストレージ使用量が増加し、最適ではないクエリプランが発生します。

症状

- 数週間または数か月にわたる段階的なクエリパフォーマンスの低下
- データ量は安定しているもののストレージ使用量が増加する
- クエリプランナーが、古い統計のため、インデックススキャンよりもシーケンシャルスキャンを選択する
- テーブル統計における `dead_tuple_count` の値が高い

診断

システムカタログをクエリすることで、すべてのテーブルにおける肥大化を推定できます。このアプローチでは、拡張機能は必要ありません。

```
SELECT schemaname, relname,
       n_dead_tup,
       n_live_tup,
       ROUND(n_dead_tup::numeric / GREATEST(n_live_tup, 1) * 100, 2) AS dead_pct,
       last_autovacuum,
       last_autoanalyze
FROM pg_stat_user_tables
WHERE n_dead_tup > 10000
ORDER BY n_dead_tup DESC
LIMIT 20;
```

肥大化に対処するには、`pg_repack` 拡張機能を使用して、最小限のロックでテーブルとインデックスを再編成できます。詳細については、「pg_repack を使用したテーブルからの肥大化の削除」および「pg_repack を使用した Amazon Aurora および RDS for PostgreSQL からの肥大化の削除」を参照してください。

> **Important**
>
> 手動メンテナンスに依存するのではなく、autovacuum が有効で、ワークロードに合わせて適切に調整されていることを確認してください。チューニングの推奨事項については、「the section called “autovacuum 調整”」を参照してください。

並列クエリのリソースの枯渇

PostgreSQL はクエリを並行して実行し、大規模なシーケンシャルスキャンと集約のパフォーマンスを向上させることができます。ただし、各並列ワーカーは、`max_worker_processes` (およびサブ制限 `max_parallel_workers`) に対してカウントされ、独自の `work_mem` を割り当てる完全なバックエンドプロセスです。4 つの並列ワーカーを含む 1 つのクエリは、数百メガバイトのメモリと大量の CPU を消費する可能性があります。同時実行性が高い場合、過剰な並列処理により CPU とメモリが急速に枯渇する可能性があります。

一般的な症状には、突然の CPU スパイク、クエリあたりのメモリ使用量の増加、アプリケーションの変更がないにもかかわらず CloudWatch で `DatabaseConnections` が上昇することなどがあります。また、`IPC:BgWorkerStartup`、`IPC:ExecuteGather`、`IPC:ParallelFinish` などの待機イベントが発生することもあります。これらの待機イベントの詳細については、「IPC:parallel 待機イベント」を参照してください。

ほとんどの OLTP および高同時実行の本稼働ワークロードでは、DB パラメータグループで `max_parallel_workers_per_gather = 0` に設定して自動並列処理を無効にします。その後、セッションごとまたはロールごとにパラメータを設定することで、特定の分析セッションまたはレポートセッションの並列処理を選択的に有効にできます。

並列クエリの動作の診断と制御に関する詳細なガイダンスについては、「RDS for PostgreSQL での並列クエリのベストプラクティス」を参照してください。

接続および認証の高負荷

接続チャーン (プーリングを行わずにデータベース接続を頻繁に開閉すること) は、認証オーバーヘッドを発生させ、使用可能な接続スロットを枯渇させる可能性があります。アイドル状態のまま開いたままになっている接続も、有用な作業を実行せずにスロットを消費します。

症状

- Performance Insights モニタリングで `total_auth_attempts` の値が増加しています。詳細については、「RDS for PostgreSQL の非ネイティブカウンター」を参照してください。

- 接続確立時間が遅い
- `FATAL: too many connections for role` または `remaining connection slots are reserved` エラー
- 接続チャーンと相関する CPU スパイク

診断

次のクエリを実行して、現在の接続状態を確認します。

```
SELECT
  setting::int AS max_connections,
  (SELECT count(*) FROM pg_stat_activity) AS current_connections,
  (SELECT count(*) FROM pg_stat_activity WHERE state = 'idle') AS idle_connections,
  (SELECT count(*) FROM pg_stat_activity WHERE state = 'idle in transaction') AS
 idle_in_txn
FROM pg_settings
WHERE name = 'max_connections';
```

`max_connections` に対する `idle` または `idle in transaction` 接続の数が多い場合は、接続が適切に解放されていないことを示しています。アイドル状態のトランザクション接続は、ロックを保持し、autovacuum によるデッドタプルの再利用を妨げるため、特に問題があります。

修正

- 接続プーリングをデプロイします。PgBouncer または Amazon RDS Proxy を使用して、データベースへの直接接続の数を減らします。接続プーリングは、リクエストごとに新しい接続を作成するのではなく、既存の接続を再利用します。
- を設定します。。**idle_in_transaction_session_timeout**このパラメータは、指定された期間を超えてトランザクションでアイドル状態のままになっているセッションを自動的に終了させます。これにより、長時間実行されるアイドル状態のトランザクションがロックを保持し、autovacuum をブロックすることを防ぎます。
- アプリケーションの接続処理を確認します。アプリケーションが接続をすぐに閉じ、トランザクションを必要以上に長く開いたままにしないようにします。

> **Note**
>
> 並列クエリワーカーは CPU とメモリを消費します。並列クエリアクティビティと同時にリソースの枯渇が見られる場合は、「the section called “並列クエリのリソースの枯渇”」を参照して、並列ワーカーの使用を制御するためのガイダンスを確認してください。

Performance Insights の待機イベントを使用したトラブルシューティング

Performance Insights は、データベースが時間を費やしている場所を示す待機イベントをキャプチャします。パフォーマンスの問題を調査する場合、待機イベントは、ボトルネックが CPU、I/O、ロック、ネットワーク、またはプロセス間通信のいずれであるかを特定するのに役立ちます。このガイドで説明されている問題発生時の一般的な待機イベントカテゴリは次のとおりです。

- CPU - セッションは CPU でアクティブであるか、CPU を待機中です。CPU 待機イベントの増加は、過剰な並列処理や、肥大化したテーブルをスキャンする非効率的なクエリプランと関連していることがよくあります。
- IPC (プロセス間通信) - `IPC:BgWorkerStartup`、`IPC:ExecuteGather`、`IPC:ParallelFinish` などの待機イベントは、並列クエリ調整のオーバーヘッドを示します。
- IO - `IO:DataFileRead` などの待機イベントは、必要なページが共有メモリにないため、クエリがストレージからデータを読み取っていることを示します。これは、肥大化したテーブルがバッファキャッシュを超えた場合に一般的です。
- ロック - `Lock:transactionid` や `Lock:tuple` などの待機イベントはセッション間の競合を示します。アイドル状態のトランザクション接続は、他のクエリや autovacuum をブロックするロックを保持している可能性があります。
- クライアント - データベースがアプリケーションがデータを送信するのを待っていることを示す `Client:ClientRead` などの待機イベント。クライアント待機イベントが多い場合は、接続チャーンまたはネットワークレイテンシーを示している可能性があります。

パフォーマンスの問題を示す一般的な待機イベントとその推奨される是正措置の完全なリファレンスについては、「RDS for PostgreSQL 待機イベント」を参照してください。

autovacuum 調整

autovacuum は、デッドタプルの再利用、テーブルとインデックスの肥大化の防止、プランナー統計の更新、トランザクション ID の循環に対する保護を行うバックグラウンドプロセスです。デフォル

トの autovacuum 設定は保守的で、小規模なデータベース用に設計されています。書き込みの多い本稼働ワークロードでは、ほとんどの場合、チューニングが必要です。

autovacuum が書き込みワークロードに対応できない場合、肥大化し、プランナー統計が古くなり、トランザクション ID の循環のリスクが高まります。`age(relfrozenxid)` が 20 億に近づくと、データベースはデータ破損を防ぐためにシャットダウンします。

autovacuum のパラメータの調整、バキュームアクティビティのモニタリング、テーブルごとのオーバーライドの設定に関する詳細なガイダンスについては、「Amazon RDS での PostgreSQL autovacuum の使用」を参照してください。

関連情報

- RDS for PostgreSQL での並列クエリのベストプラクティス
- PostgreSQL でのデッド接続の処理
- Amazon RDS での PostgreSQL Autovacuum の使用
- RDS for PostgreSQL の一般的な DBA タスク
- Amazon RDS での PostgreSQL Autovacuum の使用
- Amazon RDS for PostgreSQL 環境における自動バキュームについて

RDS for PostgreSQL DB インスタンスでのパラメータの使用

場合によっては、カスタムパラメータグループを指定せずに RDS for PostgreSQL DB インスタンスを作成することがあります。その場合、DB インスタンスは、選択したバージョンの PostgreSQL のデフォルトのパラメータグループを使用して作成されます。例えば、PostgreSQL 13.3 を使用して PostgreSQL DB インスタンス用の RDS を作成するとします。この場合、DB インスタンスは PostgreSQL 13 リリースのパラメータグループの値 `default.postgres13` を使用して作成されます。

独自のカスタム DB パラメータグループを作成することもできます。RDS for PostgreSQL DB インスタンスの設定をデフォルト値から変更する場合は、これを実行する必要があります。この方法の詳細は、「Amazon RDS のパラメータグループ」を参照してください。

RDS for PostgreSQL DB インスタンスの設定は、いくつかの異なる方法で追跡できます。AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用できます。次のように、インスタンスの PostgreSQL `pg_settings` テーブルから値をクエリすることもできます。

```
SELECT name, setting, boot_val, reset_val, unit
```

```
FROM pg_settings
ORDER BY name;
```

このクエリから返される値の詳細については、PostgreSQL ドキュメントの「pg_settings」を参照してください。

RDS for PostgreSQL DB インスタンスで max_connections と shared_buffers の設定を変更するときは、特に注意してください。例えば、max_connections または shared_buffers の設定を変更し、実際のワークロードに対して高すぎる値を使用するとします。この場合、RDS for PostgreSQL DB インスタンスは起動しません。この場合、postgres.log に次のようなエラーが表示されます。

```
2018-09-18 21:13:15 UTC::@:[8097]:FATAL:  could not map anonymous shared memory: Cannot
 allocate memory
2018-09-18 21:13:15 UTC::@:[8097]:HINT:  This error usually means that PostgreSQL's
 request for a shared memory segment
exceeded available memory or swap space. To reduce the request size (currently
 3514134274048 bytes), reduce
PostgreSQL's shared memory usage, perhaps by reducing shared_buffers or
 max_connections.
```

ただし、デフォルトの RDS for PostgreSQL DB パラメータグループに含まれる設定の値は変更できません。パラメータの設定を変更するには、まずカスタム DB パラメータグループを作成します。次に、そのカスタムグループの設定を変更し、カスタムパラメータグループを RDS for PostgreSQL DB インスタンスに適用します。詳細については Amazon RDS のパラメータグループ を参照してください。

RDS for PostgreSQL には、2 種類のパラメータがあります。

- 静的パラメータ – 静的パラメータでは、変更後に RDS for PostgreSQL DB インスタンスを再起動して、新しい値を有効にする必要があります。
- 動的パラメータ – 動的パラメータでは、設定を変更した後に再起動する必要はありません。

> **Note**
>
> RDS for PostgreSQL DB インスタンスで独自のカスタム DB パラメータグループを使用している場合は、実行中の DB インスタンスの動的パラメータの値を変更できます。これを行うには、AWS マネジメントコンソール、AWS CLI、または Amazon RDS API を使用します。

権限がある場合は、ALTER DATABASE、ALTER ROLE、および SET コマンドを使用して、パラメータ値を変更することもできます。

RDS for PostgreSQL DB インスタンスのパラメータのリスト

RDS for PostgreSQL DB インスタンスで使用可能なパラメータの一部 (すべてではありません) を次の表に示します。使用可能なすべてのパラメータを表示するには、describe-db-parametersAWS CLI コマンドを使用します。例えば、RDS for PostgreSQL バージョン 13 のデフォルトパラメータグループで使用可能なすべてのパラメータのリストを取得するには、以下を実行します。

```
aws rds describe-db-parameters --db-parameter-group-name default.postgres13
```

コンソールを使用することもできます。Amazon RDS メニューから [Parameter groups] (パラメータグループ) を選択し、AWS リージョン で利用できるパラメータグループの中からパラメータグループを選択します。

パラメータ名	Apply_Type	説明
application_name	動的	統計情報とログで報告されるアプリケーション名を設定します。
archive_command	動的	WAL ファイルをアーカイブするために呼び出されるシェルコマンドを設定します。
array_nulls	動的	配列での NULL 要素の入力を有効にします。
authentication_timeout	動的	クライアント認証の実行で許可する最大時間を設定します。
autovacuum	動的	autovacuum サブプロセスを起動します。
autovacuum_analyze_scale_factor	動的	分析する前のタプルの挿入、更新、削除の数 (reltuples の割合として指定)。
autovacuum_analyze_threshold	動的	分析する前のタプルの挿入、更新、削除の最小数。
autovacuum_freeze_max_age	静的	トランザクション ID の循環を防ぐためにテーブルに対して autovacuum を実行する期間。

パラメータ名	Apply_Type	説明
autovacuum_naptime	動的	autovacuum の実行の間で休止状態になっている時間。
autovacuum_max_workers	静的	同時に実行される autovacuum ワーカープロセスの最大数を設定します。
autovacuum_vacuum_cost_delay	動的	autovacuum でのバキューム処理のコスト遅延の値 (ミリ秒単位)。
autovacuum_vacuum_cost_limit	動的	autovacuum でバキューム処理を停止する制限値となるバキューム処理のコスト
autovacuum_vacuum_scale_factor	動的	バキューム処理する前のタプルの更新または削除の数 (reltuples の割合として指定)。
autovacuum_vacuum_threshold	動的	バキューム処理する前のタプルの更新また削除の最小数。
backslash_quote	動的	文字列リテラルでバックスラッシュ (\) を許可するかどうかを指定します。
bgwriter_delay	動的	ラウンド間でのバックグラウンドライターの休止時間。
bgwriter_lru_maxpages	動的	ラウンドあたりのフラッシュするバックグラウンドライター LRU ページの最大数。
bgwriter_lru_multiplier	動的	ラウンドあたりの解放される平均バッファ使用量の倍数。
bytea_output	動的	バイトの出力形式を設定します。
check_function_bodies	動的	CREATE FUNCTION の実行中に関数の本文をチェックします。

パラメータ名	Apply_Type	説明
checkpoint_completion_target	動的	チェックポイント中にダーティバッファのフラッシュにかかった時間 (チェックポイント間隔の割合として指定)。
checkpoint_segments	動的	自動 WAL (ログ先行書き込み) チェックポイント間のログセグメントの最大間隔を設定します。
checkpoint_timeout	動的	自動 WAL チェックポイント間の最大時間を設定します。
checkpoint_warning	動的	チェックポイントセグメントがこの値よりも頻繁に満杯になる場合に警告を出します。
client_connection_check_interval	動的	クエリ実行中の切断を確認する時間間隔を設定します。
client_encoding	動的	クライアントの文字セットエンコードを設定します。
client_min_messages	動的	クライアントへ送信されるメッセージレベルを設定します。
commit_delay	動的	トランザクションのコミットからディスクへの WAL のフラッシュまでの遅延間隔をマイクロ秒単位で設定します。
commit_siblings	動的	commit_delay を実行する前に同時に開いている必要があるトランザクションの最小数を設定します。
constraint_exclusion	動的	クエリを最適化するために、プランナーが制約を使用できるようにします。
cpu_index_tuple_cost	動的	インデックススキャンの実行中に各インデックスエントリを処理する際にかかるコストに対するプランナーの見積もりを設定します。

パラメータ名	Apply_Type	説明
cpu_operator_cost	動的	演算子の呼び出しや関数呼び出しのそれぞれを処理する際にかかるコストに対するプランナーの見積もりを設定します。
cpu_tuple_cost	動的	各タプル (行) の処理にかかるコストに対するプランナーの見積もりを設定します。
cursor_tuple_fraction	動的	取得されるカーソル行の割合に対するプランナーの見積もりを設定します。
datestyle	動的	日付と時刻の値の表示形式を設定します。
deadlock_timeout	動的	デッドロックをチェックするまでロックを待機する時間を設定します。
debug_pretty_print	動的	分析ツリーや計画ツリーの表示をインデントして見やすくします。
debug_print_parse	動的	各クエリの分析ツリーをログに記録します。
debug_print_plan	動的	各クエリの実行計画をログに記録します。
debug_print_rewritten	動的	各クエリの書き直された分析ツリーをログに記録します。
default_statistics_target	動的	デフォルトの統計情報の対象を設定します。
default_tablespace	動的	テーブルとインデックスを作成するためのデフォルトのテーブルスペースを設定します。
default_transaction_deferrable	動的	新しいトランザクションのデフォルトの遅延ステータスを設定します。
default_transaction_isolation	動的	新しい各トランザクションのトランザクション分離レベルを設定します。

パラメータ名	Apply_Type	説明
default_transaction_read_only	動的	新しいトランザクションのデフォルトの読み取り専用ステータスを設定します。
default_with_oids	動的	デフォルトでオブジェクト ID (OID) を使用して新しいテーブルを作成します。
effective_cache_size	動的	ディスクキャッシュのサイズに関するプランナーの予測を設定します。
effective_io_concurrency	動的	ディスクサブシステムで効率的に処理できる同時リクエストの数。
enable_bitmapscan	動的	プランナーがビットマップスキャン計画を使用できるようにします。
enable_hashagg	動的	プランナーがハッシュされた集計計画を使用できるようにします。
enable_hashjoin	動的	プランナーがハッシュ結合計画を使用できるようにします。
enable_indexscan	動的	プランナーがインデックススキャン計画を使用できるようにします。
enable_material	動的	プランナーがマテリアル化を使用できるようにします。
enable_mergejoin	動的	プランナーがマージ結合計画を使用できるようにします。
enable_nestloop	動的	プランナーがネステッドループ結合計画を使用できるようにします。
enable_seqscan	動的	プランナーがシーケンシャルスキャン計画を使用できるようにします。

パラメータ名	Apply_Type	説明
enable_sort	動的	プランナーが明示的なソートステップを使用できるようにします。
enable_tidscan	動的	プランナーが TID スキャン計画を使用できるようにします。
escape_string_warning	動的	通常の文字列リテラルにバックスラッシュ (\) が含まれている場合に警告を出します。
extra_float_digits	動的	浮動小数点値の表示桁数を設定します。
from_collapse_limit	動的	FROM リストのサイズを設定します。この値を超えるとサブクエリが折りたたまれなくなります。
fsync	動的	ディスクへの更新の同期を強制的に行います。
full_page_writes	動的	チェックポイントの後でページに初期の変更を加えた時点で、WAL にすべてのページを書き込みます。
geqo	動的	遺伝的クエリ最適化を有効にします。
geqo_effort	動的	GEQO: 他の GEQO パラメータのデフォルト値を設定するために使用されます。
geqo_generations	動的	GEQO: アルゴリズムの反復の数。
geqo_pool_size	動的	GEQO: 母集団内の個体の数。
geqo_seed	動的	GEQO: 無作為のパスを選択するための初期値。
geqo_selection_bias	動的	GEQO: 母集団内の選択圧。
geqo_threshold	動的	FROM 項目のしきい値を設定します。この値を超えると GEQO が使用されます。

パラメータ名	Apply_Type	説明
gin_fuzzy_search_limit	動的	GIN による完全一致検索で許可される結果の最大数を設定します。
hot_standby_feedback	動的	ホットスタンバイがフィードバックメッセージをプライマリあるいはアップストリーミングスタンバイに送信するかを決定します。
intervalstyle	動的	間隔値の表示形式を設定します。
join_collapse_limit	動的	FROM リストのサイズを設定します。この値を超えると JOIN 構造が平坦化されなくなります。
lc_messages	動的	メッセージを表示する言語を設定します。
lc_monetary	動的	金額の書式のロケールを設定します。
lc_numeric	動的	数値の書式のロケールを設定します。
lc_time	動的	日付と時刻の書式のロケールを設定します。
log_autovacuum_min_duration	動的	autovacuum に関する最小実行時間を設定します。この値を超えると autovacuum アクションがログに記録されます。
log_checkpoints	動的	各チェックポイントをログに記録します。
log_connections	動的	成功した各接続をログに記録します。
log_disconnections	動的	セッションの終了をログに記録します (セッションの有効期間も含まれます)。
log_duration	動的	完了した各 SQL ステートメントの期間をログに記録します。
log_error_verbosity	動的	ログに記録されるメッセージの詳細を設定します。

パラメータ名	Apply_Type	説明
log_executor_stats	動的	実行プログラムのパフォーマンスの統計情報をサーバーログに書き込みます。
log_filename	動的	ログファイルのファイル名のパターンを設定します。
log_file_mode	動的	ログファイルのファイルアクセス許可を設定します。デフォルト値は 0644 です。
log_hostname	動的	接続ログにホスト名を記録します。PostgreSQL 12 以降のバージョンでは、このパラメータはデフォルトで「off」になっています。オンにすると、接続は DNS 逆引き参照を使用してホスト名を取得し、接続ログに取り込まれます。このパラメータをオンにする場合は、接続の確立にかかる時間への影響をモニタリングする必要があります。
log_line_prefix	動的	各ログ行の先頭に付ける情報を制御します。
log_lock_waits	動的	長期間にわたるロックの待機をログに記録します。
log_min_duration_statement	動的	最小実行時間を設定します。この値を超えるとステートメントがログに記録されます。
log_min_error_statement	動的	設定したレベル以上のエラーが発生したすべてのステートメントをログに記録します。
log_min_messages	動的	ログに記録するメッセージレベルを設定します。
log_parser_stats	動的	分析のパフォーマンスの統計情報をサーバーログに書き込みます。
log_planner_stats	動的	プランナーのパフォーマンスの統計情報をサーバーログに書き込みます。

パラメータ名	Apply_Type	説明
log_rotation_age	動的	N 分が経過するとログファイルのローテーションが自動的に発生します。
log_rotation_size	動的	N キロバイトを超えるとログファイルのローテーションが自動的に発生します。
log_statement	動的	ログに記録するステートメントのタイプを設定します。
log_statement_stats	動的	累積処理のパフォーマンスの統計情報をサーバーログに書き込みます。
log_temp_files	動的	指定したサイズ (キロバイト) を超えるテンポラリファイルの使用をログに記録します。
log_timezone	動的	ログメッセージで使用するタイムゾーンを設定します。
log_truncate_on_rotation	動的	ログローテーション中に同じ名前の既存のログファイルを切り捨てます。
logging_collector	静的	サブプロセスを開始して、stderr 出力や csvlogs をログファイルにキャプチャします。
maintenance_work_mem	動的	メンテナンスオペレーションに使用するメモリの最大量を設定します。
max_connections	静的	同時接続の最大数を設定します。
max_files_per_process	静的	各サーバープロセスで同時に開くことができるファイルの最大数を設定します。
max_locks_per_transaction	静的	トランザクションあたりのロックの最大数を設定します。
max_pred_locks_per_transaction	静的	トランザクションあたりの述語ロックの最大数を設定します。

パラメータ名	Apply_Type	説明
max_prepared_transactions	静的	同時に準備できるトランザクションの最大数を設定します。
max_stack_depth	動的	スタックの深度の最大値をキロバイト単位で指定します。
max_standby_archive_delay	動的	ホットスタンバイサーバーがアーカイブされた WAL データを処理しているときにクエリをキャンセルするまでの最大遅延間隔を設定します。
max_standby_streaming_delay	動的	ホットスタンバイサーバーがストリーミングされた WAL データを処理しているときにクエリをキャンセルするまでの最大遅延間隔を設定します。

max_wal_size	動的	チェックポイントをトリガーする WAL サイズ (MB) を設定します。
		• RDS for PostgreSQL 15 以前のバージョンでは、2 GB です。
		• RDS for PostgreSQL 16 以降のバージョンでは、インスタンスに割り当てられたストレージサイズに基づいて RDS for PostgreSQL が自動的に設定されます。
		  • 割り当てられたストレージが 100 GB 以上のインスタンスの場合は 6 GB。
		  • 割り当てられたストレージが 100 GB 未満のインスタンスの場合は 2 GB。
		
		Amazon RDS for PostgreSQL DB インスタンスで以下のコマンドを使用して現在の値を確認します。
		`SHOW max_wal_size;`

パラメータ名	Apply_Type	説明
min_wal_size	動的	WAL を縮小する最小サイズを設定します。PostgreSQL バージョン 9.6 以前の場合、min_wal_size の単位は 16 MB です。PostgreSQL バージョン 10 以降の場合、min_wal_size の単位は 1 MB です。
quote_all_identifiers	動的	SQL フラグメントを生成するときに、すべての識別子に引用符 (") を追加します。
random_page_cost	動的	非連続的に取得されたディスクページのコストに対するプランナーの見積もりを設定します。クエリプラン管理 (QPM) がオンでない限り、このパラメータには値はありません。QPM がオンの場合、このパラメータ 4 はデフォルト値です。
rds.adaptive_autovacuum	動的	トランザクション ID のしきい値を超えるたびに、autovacuum パラメータを自動的に微調整します。
rds.force_ssl	動的	SSL 接続を使用する必要があります。PostgreSQL バージョン 15 の RDS では、デフォルト値は 1 (オン) に設定されています。PostgreSQL のメジャーバージョン 14 以前のその他すべての RDS では、デフォルト値が 0 (オフ) に設定されています。
rds.local_volume_spill_enabled	静的	ローカルボリュームへの論理スピルファイルの書き込みを有効にします。
rds.log_retention_period	動的	n 分より古い PostgreSQL ログは Amazon RDS で削除されるようにログ保持期間を設定します。

パラメータ名	Apply_Type	説明
rds.rds_superuser_reserved_connections	静的	rds_superusers 用に予約されている接続スロットの数を設定します。このパラメータは、バージョン 15 以前でのみ使用できます。詳細については、PostgreSQL ドキュメントの「reserved_connections」を参照してください。
rds.replica_identity_full	動的	このパラメータを on に設定すると、すべてのデータベーステーブルのレプリカ ID 設定が FULL に上書きされます。つまり、REPLICA IDENTITY FULL 設定に関係なく、すべての列値が先書き口グ (WAL) に書き込まれます。 **Note** このパラメータをオンにすると、追加の WAL ログ記録のためにデータベースインスタンスの IOPS が増加する可能性があります。
rds.restrict_password_commands	静的	rds_password ロールを持つユーザーに対して、だれがパスワードを管理するかを制限します。パスワードの制限を有効にするには、このパラメータを 1 に設定します。デフォルトは 0 です。
search_path	動的	スキーマによって修飾されていない名前でスキーマを検索する順序を設定します。
seq_page_cost	動的	連続的に取得されたディスクページのコストに対するプランナーの見積もりを設定します。
session_replication_role	動的	トリガーと再書き込みルールに対するセッション動作を設定します。

パラメータ名	Apply_Type	説明
shared_buffers	静的	サーバーで使用される共有メモリバッファの数を設定します。
shared_preload_libraries	静的	RDS for PostgreSQL DB インスタンスにプリロードする共有ライブラリをリストします。サポートされている値は、auto_explain、orafce、pgaudit、pglogical、pg_bigm、pg_cron、pg_hint_plan、pg_prewarm、pg_similarity、pg_stat_statements、pg_transport、plprofiler、および plrust です。
ssl	動的	SSL 接続を有効にします。
sql_inheritance	動的	さまざまなコマンドにデフォルトでサブテーブルが取り込まれます。
ssl_renegotiation_limit	動的	暗号化キーを再度ネゴシエートする前に送受信されるトラフィックの量を設定します。
standard_conforming_strings	動的	... 文字列をリテラルのバックスラッシュとして扱います。
statement_timeout	動的	すべてのステートメントに許可される最大実行時間を設定します。
synchronize_seqscans	動的	シーケンシャルスキャンの同期を有効にします。
synchronous_commit	動的	現在のトランザクションの同期レベルを設定します。
tcp_keepalives_count	動的	TCP キープアライブを再送信する最大回数。
tcp_keepalives_idle	動的	TCP キープアライブを発行する間隔の時間。
tcp_keepalives_interval	動的	TCP キープアライブを再送信する間隔の時間。

パラメータ名	Apply_Type	説明
temp_buffers	動的	各セッションで使用されるテンポラリバッファの最大数を設定します。
temp_file_limit	動的	テンポラリファイルの最大サイズを KB 単位で設定します。
temp_tablespaces	動的	テンポラリテーブルとソートファイルで使用するテーブルスペースを設定します。
timezone	動的	表示やタイムスタンプの解釈で必要となるタイムゾーンを設定します。 Internet Assigned Numbers Authority (IANA) は年に数回、https://www.iana.org/time-zones で新しいタイムゾーンを公開します。RDS が PostgreSQL の新しいマイナーメンテナンスリリースをリリースするたびに、リリース時の最新のタイムゾーンデータが付属しています。最新の RDS for PostgreSQL バージョンを使用すると、RDS からの最新のタイムゾーンデータが得られます。DB インスタンスに最新のタイムゾーンデータがあることを確認するには、DB エンジンの上位バージョンにアップグレードすることをお勧めします。PostgreSQL DB インスタンスのタイムゾーン テーブルを手動で変更することはできません。RDS は、実行中の DB インスタンスのタイムゾーンデータを変更またはリセットしません。新しいタイムゾーンデータは、データベースエンジンのバージョンアップグレードを実行する場合にのみインストールされます。
track_activities	動的	コマンドの実行に関する情報を収集します。
track_activity_query_size	静的	pg_stat_activity.current_query 用に予約するサイズをバイト単位で設定します。

パラメータ名	Apply_Type	説明
track_counts	動的	データベースアクティビティの統計情報を収集します。
track_functions	動的	データベースアクティビティの関数レベルの統計情報を収集します。
track_io_timing	動的	データベース I/O アクティビティのタイミングに関する統計情報を収集します。
transaction_deferrable	動的	読み取り専用のシリアル化可能なトランザクションを、シリアル化が失敗する可能性がない状況でスタートできるようになるまで延期するかどうかを示します。
transaction_isolation	動的	現在のトランザクションの分離レベルを設定します。
transaction_read_only	動的	現在のトランザクションの読み取り専用ステータスを設定します。
transform_null_equals	動的	expr=NULL を expr IS NULL として扱います。
update_process_title	動的	アクティブな SQL コマンドを表示するようにプロセスのタイトルを更新します。
vacuum_cost_delay	動的	バキューム処理のコスト遅延の値 (ミリ秒単位)。
vacuum_cost_limit	動的	バキューム処理を停止する制限値となるバキューム処理のコスト。
vacuum_cost_page_dirty	動的	バキューム処理によってダーティになったページに対するバキューム処理のコスト。
vacuum_cost_page_hit	動的	バッファキャッシュ内で検出されたページに対するバキューム処理のコスト。

パラメータ名	Apply_Type	説明
vacuum_cost_page_miss	動的	バッファキャッシュ内で検出されなかったページに対するバキューム処理のコスト。
vacuum_defer_cleanup_age	動的	バキューム処理とホットクリーンアップが延期されるトランザクションの数 (存在する場合)。
vacuum_freeze_min_age	動的	バキューム処理でテーブルの行をフリーズする最小期間。
vacuum_freeze_table_age	動的	バキューム処理でテーブル全体をスキャンしタプルをフリーズするための期間。
wal_buffers	静的	WAL 用の共有メモリ内のディスクページバッファの数を設定します。
wal_writer_delay	動的	WAL のフラッシュが行われる間の WAL ライターの休止時間。
work_mem	動的	クエリワークスペースに使用するメモリの最大量を設定します。
xmlbinary	動的	バイナリ値を XML にエンコードする方法を設定します。
xmloption	動的	黙示的な分析とシリアル化オペレーションでの XML データをドキュメントとして見なすか、コンテンツのフラグメントとして見なすかを設定します。

Amazon RDS では、すべてのパラメータについて PostgreSQL のデフォルトの単位を使用します。次の表は、PostgreSQL のパラメータ別のデフォルト単位を示しています。

パラメータ名	Unit
archive_timeout	s

パラメータ名	Unit
authentication_timeout	s
autovacuum_naptime	s
autovacuum_vacuum_cost_delay	ms
bgwriter_delay	ms
checkpoint_timeout	s
checkpoint_warning	s
deadlock_timeout	ms
effective_cache_size	8 KB
lock_timeout	ms
log_autovacuum_min_duration	ms
log_min_duration_statement	ms
log_rotation_age	minutes
log_rotation_size	KB
log_temp_files	KB
maintenance_work_mem	KB
max_stack_depth	KB
max_standby_archive_delay	ms
max_standby_streaming_delay	ms
post_auth_delay	s
pre_auth_delay	s

パラメータ名	Unit
segment_size	8 KB
shared_buffers	8 KB
statement_timeout	ms
ssl_renegotiation_limit	KB
tcp_keepalives_idle	s
tcp_keepalives_interval	s
temp_file_limit	KB
work_mem	KB
temp_buffers	8 KB
vacuum_cost_delay	ms
wal_buffers	8 KB
wal_receiver_timeout	ms
wal_segment_size	B
wal_sender_timeout	ms
wal_writer_delay	ms
wal_receiver_status_interval	s

RDS for PostgreSQL の待機イベントでのチューニング

待機イベントは RDS for PostgreSQL の重要なチューニングツールです。セッションがリソースを待っている理由とその内容がわかれば、ボトルネックを減少できます。このセクションの情報を使用して、考えられる原因と修正措置を見つけることができます。このセクションでは、PostgreSQL の基本的なチューニングの概念についても説明します。

このセクションの待機イベントは RDS for PostgreSQL 固有のものです。

トピック

- RDS for PostgreSQL チューニングの基本概念
- RDS for PostgreSQL 待機イベント
- Client:ClientRead
- クライアント: ClientWrite
- CPU
- IO:BufFileRead および IO:BufFileWrite
- IO:DataFileRead
- IO:WALWrite
- IPC:parallel 待機イベント
- IPC:ProcArrayGroupUpdate
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:BufferMapping (LWLock:buffer_mapping)
- LWLock:BufferIO (IPC:BufferIO)
- LWLock:buffer_content (BufferContent)
- LWLock:lock_manager (LWLock:lockmanager)
- LWLock:pg_stat_statements
- LWLock:SubtransSLRU (LWLock:SubtransControlLock)

- Timeout:PgSleep
- Timeout:VacuumDelay

RDS for PostgreSQL チューニングの基本概念

RDS for PostgreSQL データベースをチューニングする前に、待機イベントは何か、なぜそれが発生するのかを確認してください。RDS for PostgreSQL のベーシックメモリとディスクアーキテクチャも確認します。役立つアーキテクチャの図表については、PostgreSQLウィキブックを参照してください。

トピック

- RDS for PostgreSQL 待機イベント
- RDS for PostgreSQL メモリ
- RDS for PostgreSQL プロセス

RDS for PostgreSQL 待機イベント

待機イベントは、リソースに対して待機しているリソースを示します。例えば、待機イベント `Client:ClientRead` は RDS for PostgreSQL がクライアントからのデータの受信を待っているときに発生します。セッションは通常、次のようなリソースを待ちます。

- バッファへのシングルスレッドアクセス (例えば、セッションがバッファを変更しようとした場合など)
- 別のセッションによって現在ロックされている行
- 読み込まれたデータファイル
- ログファイルの書き込み

例えば、クエリを満たすために、セッションで完全なテーブルスキャンを実行することがあります。データがまだメモリ上にない場合、セッションはディスク I/O が完了するまで待機します。バッファがメモリに読み込まれるときは、他のセッションが同じバッファにアクセスしているため、セッションは待機しなければならないことがあります。データベースは、事前定義された待機イベントを使用して待機を記録します。これらのイベントはカテゴリに分類されます。

待機イベント自体では、パフォーマンスの問題は表示されません。例えば、要求されたデータがメモリ上にない場合は、ディスクからデータを読み出す必要があります。あるセッションが更新のために

行をロックすると、別のセッションはその行を更新できるようにロック解除されるまで待機します。コミットは、ログファイルへの書き込みが完了するまで待機する必要があります。待機は、データベースが正常に機能するために不可欠です。

一方で、待機イベントが発生すると、通常、パフォーマンスの問題を示します。そのような場合、待機イベントデータを使用して、セッションが時間を費やしている場所を特定できます。例えば、通常は数分で実行されるレポートが数時間かかるようになった場合、合計の待機時間に最も寄与している待機イベントを特定できます。上位の待機イベントの原因を特定できる場合は、パフォーマンス向上のための変更を実行できることがあります。例えば、別のセッションによってロックされている行をセッションが待っている場合、ロックセッションを終了させることができます。

RDS for PostgreSQL メモリ

RDS for PostgreSQL メモリは、共有とローカルに分かれています。

トピック

- RDS for PostgreSQL の共有メモリ
- RDS for PostgreSQL のローカルメモリ

RDS for PostgreSQL の共有メモリ

RDS for PostgreSQL では、インスタンスの起動時に共有メモリを割り当てます。共有メモリは複数のサブエリアに分割されています。以下のセクションでは、最も重要なものについて説明します。

トピック

- 共有バッファ
- ログ先行書き込み (WAL) バッファ

共有バッファ

共有バッファプールは、アプリケーション接続によって使用されている、または使用されていたすべてのページを保持する RDS for PostgreSQL メモリ領域です。ページは、ディスクブロックのメモリバージョンです。共有バッファプールは、ディスクから読み込まれたデータブロックをキャッシュします。プールは、ディスクからデータを再読み取りする必要性を減らし、データベースの運用効率を向上させます。

すべてのテーブルとインデックスは、固定サイズのページの配列として格納されます。各ブロックには、行に対応する複数のタプルが含まれています。タプルはどのページにも格納できます。

共有バッファプールには有限メモリがあります。新しいリクエストがメモリにないページを必要とし、メモリがもう存在しない場合、RDS for PostgreSQL は使用頻度の低いページを削除してリクエストに対応します。削除ポリシーは、クロックスイープアルゴリズムによって実装されます。

`shared_buffers`パラメータは、サーバーがデータをキャッシュするメモリ量を決定します。デフォルト値は、DB インスタンスで利用可能なメモリに基づいて `{DBInstanceClassMemory/32768}` バイトに設定されます。

ログ先行書き込み (WAL) バッファ

ログ先行書き込み (WAL) バッファは、RDS for PostgreSQL が後で永続的ストレージに書き込むトランザクションデータを保持します。WAL メカニズムを使用すると、RDS for PostgreSQL は次のことを実行できます。

- 障害発生後のデータリカバリ
- ディスクへの頻繁な書き込みを回避し、ディスク I/O を削減

クライアントがデータを変更すると、RDS for PostgreSQL は WAL バッファに変更内容を書き込みます。クライアントが`COMMIT`を発すると、WAL ライタプロセスはトランザクションデータを WAL ファイルに書き込みます。

`wal_level` パラメータは、WAL に書き込む情報量を決定します。`minimal`、`replica`、`logical` などの値を設定できます。

RDS for PostgreSQL のローカルメモリ

すべてのバックエンドプロセスは、クエリ処理にローカルメモリを割り当てます。

トピック

- ワークメモリ領域
- メンテナンス作業用メモリ領域
- テンポラリバッファ領域

ワークメモリ領域

ワークメモリ領域ソートとハッシュを実行するクエリのテンポラリデータを保持します。例えば、`ORDER BY`文節を持つクエリはソートを実行します。クエリは、ハッシュ結合と集約でハッシュテーブルを使用します。

work_mem パラメータは、一時ディスクファイルに書き込む前に、内部のソートオペレーションとハッシュテーブルで使用するメモリ量 (メガバイト単位) を指定します。デフォルト値は 4 MB です。複数のセッションを同時に実行でき、各セッションでメンテナンスオペレーションを並行して行うことができます。このため、使用されるワークメモリの合計は、work_mem設定の何倍にもなることがあります。

メンテナンス作業用メモリ領域

メンテナンス作業用メモリ領域は、メンテナンスオペレーション用のデータをキャッシュします。これらの操作には、バキューム処理、インデックス作成、外部キーの追加が含まれます。

maintenance_work_mem パラメータは、メンテナンスオペレーションで使用するメモリの最大量 (メガバイト単位) を指定します。デフォルト値は 64 MB です。データベースセッションでは、一度に 1 つのメンテナンスオペレーションしか実行できません。

テンポラリバッファ領域

テンポラリバッファ領域は、データベースセッションごとにテンポラリテーブルをキャッシュします。

各セッションは、指定した制限まで、必要に応じてテンポラリバッファを割り当てます。セッションが終了すると、サーバーはバッファをクリアします。

temp_buffers パラメータは、各セッションで使用する一時バッファの最大数 (メガバイト単位) を設定します。デフォルト値は 8 MB です。セッション内でテンポラリテーブルを初期に使用する前に、temp_buffers値を変更できます。

RDS for PostgreSQL プロセス

RDS for PostgreSQL では複数のプロセスを使用します。

トピック

- Postmaster プロセス
- バックエンドプロセス
- バックグラウンドプロセス

Postmaster プロセス

Postmaster プロセスは、RDS for PostgreSQL を起動したときに初期のプロセスがスタートされます。Postmaster プロセスには、主に次のようなロールがあります。

- バックグラウンドプロセスのフォークとモニタリング
- クライアントプロセスから認証要求を受信し、データベースが要求を処理する前に認証する

バックエンドプロセス

Postmaster がクライアント要求を認証する場合、Postmaster は新しいバックエンドプロセスをフォークします。これは postgres プロセスとも呼ばれます。1 つのクライアントプロセスが 1 つのバックエンドプロセスに接続されます。クライアントプロセスとバックエンドプロセスは、Postmaster プロセスの介入なしに直接通信します。

バックグラウンドプロセス

Postmaster プロセスは、異なるバックエンドタスクを実行するいくつかのプロセスをフォークします。より重要なものとしては、以下のとおりです。

- WALライター

  RDS for PostgreSQL は WAL (ログ先行書き込み) バッファのデータをログファイルに書き込みます。ログ先行書き込みの原理は、データベースがそれらの変更を説明するログレコードをディスクに書き込むまで、データベースがデータファイルに変更を書き込むことができないということです。WAL メカニズムはディスク I/O を削減し、RDS for PostgreSQL が障害後にデータベースを回復するためにログを使用できるようにします。
- バックグラウンドライター

  このプロセスは、メモリバッファからデータファイルにダーティ (変更された) ページを定期的に書き込みます。バックエンドプロセスがメモリ上でページを変更すると、ページがダーティになります。
- オートバキュームデーモン

  最新のデーモンには以下の構成要素があります。
  - オートバキュームランチャー
  - オートバキュームワーカープロセス

  オートバキュームをオンにすると、多数の挿入、更新、または削除されたタプルがあるテーブルを確認します。デーモンには、次のようなロールがあります。
  - 更新または削除された行によって占有されているディスク領域をリカバリまたは再利用する
  - プランナーで使用する統計情報を更新する
  - トランザクション ID のラップアラウンドによる古いデータの損失からの保護

オートバキューム機能は、VACUUMとANALYZEコマンドの実行を自動化するもので、VACUUMにはスタンダードとフルのバリエーションがあります。スタンダードバキュームは、他のデータベースオペレーションと並行して実行されます。 VACUUM FULLは、作業中のテーブルを排他的にロックする必要があります。そのため、同じテーブルにアクセスするオペレーションと並行して実行することはできません。VACUUMは相当量の I/O トラフィックを作成し、他のアクティブなセッションのパフォーマンスが低下する原因となることがあります。

RDS for PostgreSQL 待機イベント

次の表では、パフォーマンスの問題を最もよく示す RDS for PostgreSQL の待機イベントと、最も一般的な原因および修正処置をリストアップしています。

待機イベント	定義
Client:ClientRead	このイベントは、RDS for PostgreSQL がクライアントからのデータ受信を待っているときに発生します。
クライアント: ClientWrite	このイベントは、RDS for PostgreSQL がクライアントへのデータ書き込みを待っているときに発生します。
CPU	この待機イベントは、スレッドが CPU でアクティブであるか CPU の待機中に発生します。
IO:BufFileRead および IO:BufFileWrite	これらのイベントは、RDS for PostgreSQL がテンポラリファイルを作成するときに発生します。
IO:DataFileRead	このイベントは、バックエンドプロセスが必要なページをストレージから読み込む際に、ページが共有メモリで使用できないために接続が待機したときに発生します。
IO:WALWrite	このイベントは、RDS for PostgreSQL が WAL ファイルへの先行書き込みログ (WAL) バッファの書き込みを待機しているときに発生します。

待機イベント	定義
IPC:parallel 待機イベント	これらの待機イベントは、セッションが並列クエリ実行オペレーションに関連するプロセス間通信を待機していることを示します。
IPC:ProcArrayGroupUpdate	このイベントは、セッションが、グループリーダーによるトランザクション終了時のトランザクションステータスの更新を待機しているときに発生します。
Lock:advisory	このイベントは、PostgreSQL アプリケーションがロックを使用して、複数のセッションにわたるアクティビティを調整するときに発生します。
Lock:extend	このイベントは、バックエンドプロセスがリレーション拡張のために ロックするのを待機中に、他のプロセスが同じ目的でそのリレーションを ロックしているときに発生します。
Lock:Relation	このイベントは、他のトランザクションによって現在ロックされているテーブルまたはビューに対するロックを取得するためにクエリが待っているときに発生します。
Lock:transactionid	このイベントは、トランザクションが行レベルロックを待っているときに発生します。
Lock:tuple	このイベントは、バックエンドプロセスがタプルのロック取得を待機中の場合に発生します。
LWLock:BufferMapping (LWLock:buffer_mapping)	このイベントは、セッションがデータブロックを共有バッファプール内のバッファに関連付けるのを待っているときに発生します。
LWLock:BufferIO (IPC:BufferIO)	このイベントは、RDS for PostgreSQL が、ページへの同時アクセスを試みたときに、他のプロセスが入出力 (I/O) オペレーションを完了するのを待っているときに発生します。

待機イベント	定義
LWLock:buffer_content (BufferContent)	このイベントは、セッションがデータページのメモリ内への読み取りまたは書き込みのために待機中、そのデータページが他のセッションで書き込むためにロックされているときに発生します。
LWLock:lock_manager (LWLock:lockmanager)	このイベントは、RDS for PostgreSQL エンジンが、高速パスロックが不可能な場合に共有ロックのメモリ領域を維持し、ロックの割り当て、チェック、および解放を行うときに発生します。
LWLock:SubtransSLRU (LWLock:SubtransControlLock)	このイベントは、プロセスがサブトランザクションのシンプルな最も長い時間使われていない (SLRU) キャッシュへのアクセスを待機しているときに発生します。
LWLock:pg_stat_statements	このイベントは、pg_stat_statements 拡張機能が SQL ステートメントを追跡するハッシュテーブルで排他的ロックを取得したときに発生します。
Timeout:PgSleep	このイベントは、サーバープロセスがpg_sleep機能を呼び出し、スリープタイムアウトを待っているときに発生します。
Timeout:VacuumDelay	このイベントは、推定コスト制限に達したため、バキュームプロセスがスリープ状態になっていることを示しています。

Client:ClientRead

Client:ClientRead イベントは、RDS for PostgreSQL がクライアントからのデータ受信を待っているときに発生します。

トピック

- サポート対象エンジンバージョン

- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL バージョン 10 以降でサポートされています。

Context

RDS for PostgreSQL DB インスタンスは、クライアントからのデータ受信を待っています。RDS for PostgreSQL DB インスタンスは、クライアントにさらにデータを送信する前に、クライアントからデータを受信する必要があります。インスタンスがクライアントからデータを受信する前に待機する時間が `Client:ClientRead` イベントとなります。

待機時間が増加する原因の可能性

`Client:ClientRead`上位待機中に表示されるイベントの一般的な原因には、次のものがあります。

ネットワークレイテンシーの増加

RDS for PostgreSQL DB インスタンスとクライアントの間のネットワークレイテンシーが増加することがあります。ネットワークレイテンシーが高いほど、DB インスタンスがクライアントからデータを受信するために必要な時間が長くなります。

クライアントへの負荷の増大

クライアント側で CPU プレッシャーまたはネットワーク飽和が発生している可能性があります。クライアント側の負荷が増加すると、クライアントから RDS for PostgreSQL DB インスタンスへのデータの転送が遅延する可能性があります。

過剰なネットワークラウンドトリップ

RDS for PostgreSQL DB インスタンスとクライアントの間のネットワークラウンドトリップが多くなると、クライアントから RDS for PostgreSQL DB インスタンスへのデータの転送が遅延する可能性があります。

大規模なコピーオペレーション

コピーオペレーション中、データはクライアントのファイルシステムから RDS for PostgreSQL DB インスタンスに転送されます。DB インスタンスに大量のデータを送信すると、クライアントから DB インスタンスへのデータの転送が遅延する可能性があります。

アイドル状態のクライアントの接続

クライアントが RDS for PostgreSQL DB インスタンスに `idle in transaction` 状態で接続している場合、DB インスタンスは、クライアントがより多くのデータを送信するのを待ったり、コマンドを発したりすることがあります。この状態での接続は、`Client:ClientRead`イベントの増加につながることがあります。

接続プーリングに使用される pgBouncer

pgBouncer には`pkt_buf`という低レベルネットワーク構成設定があり、デフォルトでは 4,096 に設定されています。ワークロードが 4,096 バイトを超えるクエリパケットを pgBouncer を介して 送信する場合は、`pkt_buf`8,192 に設定することをお勧めします。新しい設定で`Client:ClientRead`イベントの数が減らない場合は、`pkt_buf`を16,384 や 32,768 など、より大きな値に設定にすることをお勧めします。クエリテキストが大きい場合は、大きな設定を使用すると特に効果的です。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- クライアントをインスタンスと同じアベイラビリティーゾーンと VPC サブネットに配置します。
- クライアントのスケーリング
- 現行世代のインスタンスを使用
- ネットワーク帯域幅の増加
- ネットワークパフォーマンスの最大値をモニタリングする
- 「トランザクションのアイドル」状態のトランザクションをモニタリングする

クライアントをインスタンスと同じアベイラビリティーゾーンと VPC サブネットに配置します。

ネットワークレイテンシーを減らしてネットワークスループットを向上するには、RDS for PostgreSQL DB インスタンスと同じアベイラビリティーゾーンおよび仮想プライベートクラウド

(VPC) サブネットにクライアントを配置します。クライアントが、DB インスタンスにできる限り地理的に近い場所に配置されていることを確認してください。

クライアントのスケーリング

Amazon CloudWatch またはその他のホストメトリクスを使用して、クライアント側が現在 CPU またはネットワーク帯域幅、またはその両方によって制約を受けているかどうかを判断します。クライアント側が制約を受けている場合は、それに応じてクライアントをスケーリングします。

現行世代のインスタンスを使用

場合によっては、ジャンボフレームをサポートする DB インスタンスクラスを使用していない可能性があります。Amazon EC2 でアプリケーションを実行している場合は、クライアント側に現行世代のインスタンスを使用することを検討してください。また、クライアントのOSで最大送信単位 (MTU) を設定します。この技術では、ネットワークラウンドトリップの数を減らし、ネットワークスループットを向上させることができます。詳細については、「Amazon EC2 ユーザーガイド」の「ジャンボフレーム (9001 MTU)」を参照してください。

DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。Amazon EC2 インスタンスタイプと同等の DB インスタンスクラスを決定するには、db.Amazon EC2 インスタンスタイプの前に配置します。例えば、`r5.8xlarge`Amazon EC2 インスタンスは`db.r5.8xlarge`DB インスタンスクラスと同等です。

ネットワーク帯域幅の増加

`NetworkReceiveThroughput` および `NetworkTransmitThroughput` の Amazon CloudWatch メトリクスを使用して、DB インスタンス上の着信および発信ネットワークトラフィックをモニタリングします。これらのメトリックは、ネットワーク帯域幅がワークロードに十分であるかどうかを判断するのに役立ちます。

ネットワーク帯域幅が十分でない場合は、増加してください。AWSクライアントまたは DB インスタンスがネットワーク帯域幅の制限に達している場合、帯域幅を増やす唯一の方法は、DB インスタンスのサイズを増加することことです。詳細については、「DB インスタンスクラスタイプ」を参照してください。

CloudWatch のメトリクスの詳細については、「Amazon RDS の Amazon CloudWatch メトリクス」を参照してください。

ネットワークパフォーマンスの最大値をモニタリングする

Amazon EC2 クライアントを使用している場合、Amazon EC2 は、集約されたインバウンドとアウトバウンドのネットワーク帯域幅を含む、ネットワークパフォーマンスメトリックの最大値を提供

します。また、パケットが期待どおりに返されることを確認する接続追跡、ドメインネームシステム (DNS) などのサービスへのリンクローカルサービスアクセスも提供します。これらの最大値をモニタリングするには、現在の拡張ネットワークドライバーを使用し、クライアントのネットワークパフォーマンスをモニタリングします。

詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスのネットワークパフォーマンスのモニタリング」および「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスのネットワークパフォーマンスのモニタリング」を参照してください。

「トランザクションのアイドル」状態のトランザクションをモニタリングする

`idle in transaction`接続の数が増えているかどうかをチェックします。これを行うには、`pg_stat_activity`テーブルの`state`列をモニタリングします。次のようなクエリを実行することで、接続出典を特定できる場合があります。

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

クライアント: ClientWrite

`Client:ClientWrite` イベントは、RDS for PostgreSQL がクライアントへのデータ書き込みを待っているときに発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL バージョン 10 以降でサポートされています。

Context

クライアントプロセスは、クラスターがさらにデータを送信する前に、RDS for PostgreSQL DB クラスターから受信したすべてのデータを読み込む必要があります。クライアントにより多くのデータを送信する前にクラスターが待機する時間は、`Client:ClientWrite`イベントになります。

RDS for PostgreSQL DB インスタンスとクライアント間のネットワークスループットが低下すると、このイベントが発生することがあります。クライアントの CPU プレッシャーとネットワークの飽和により、このイベントが発生することがあります。CPU プレッシャーとは、CPU が完全に使用されており、CPU 時間を待っているタスクがあることです。ネットワーク飽和度とは、データベースとクライアント間のネットワークが、処理できるデータ以上のデータを伝送しているときです。

待機時間が増加する原因の可能性

`Client:ClientWrite`上位待機中に表示されるイベントの一般的な原因には、次のものがあります。

ネットワークレイテンシーの増加

RDS for PostgreSQL DB インスタンスとクライアントの間のネットワークレイテンシーが増加することがあります。ネットワークレイテンシーが高いほど、クライアントからデータを受信するために必要な時間が長くなります。

クライアント側への負荷の増加

クライアント側で CPU プレッシャーまたはネットワーク飽和が発生している可能性があります。クライアントの負荷が増加すると、RDS for PostgreSQL DB インスタンスからのデータの受信が遅延します。

クライアントに送信される大量のデータ

RDS for PostgreSQL DB インスタンスがクライアントに大量のデータを送信している可能性があります。クライアントは、クラスターのデータ送信と同じ速度ではデータを受信できない場合があります。大きなテーブルのコピーなどのアクティビティは、`Client:ClientWrite`イベントの増加につながることがあります。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- クライアントをクラスターと同じアベイラビリティーゾーンと VPC サブネットに配置します。
- 現行世代のインスタンスを使用
- クライアントに送信するデータ量を減らします。
- クライアントのスケーリング

クライアントをクラスターと同じアベイラビリティーゾーンと VPC サブネットに配置します。

ネットワークレイテンシーを減らしてネットワークスループットを向上するには、RDS for PostgreSQL DB インスタンスと同じアベイラビリティーゾーンおよび仮想プライベートクラウド (VPC) サブネットにクライアントを配置します。

現行世代のインスタンスを使用

場合によっては、ジャンボフレームをサポートする DB インスタンスクラスを使用していない可能性があります。Amazon EC2 でアプリケーションを実行している場合は、クライアント側に現行世代のインスタンスを使用することを検討してください。また、クライアントのOSで最大送信単位 (MTU) を設定します。この技術では、ネットワークラウンドトリップの数を減らし、ネットワークスループットを向上させることができます。詳細については、「Amazon EC2 ユーザーガイド」の「ジャンボフレーム (9001 MTU)」を参照してください。

DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。Amazon EC2 インスタンスタイプと同等の DB インスタンスクラスを決定するには、db.Amazon EC2 インスタンスタイプの前に配置します。例えば、`r5.8xlarge`Amazon EC2 インスタンスは`db.r5.8xlarge`DB インスタンスクラスと同等です。

クライアントに送信するデータ量を減らします。

可能であれば、RDS for PostgreSQL DB インスタンスがクライアントに送信するデータ量を減らすようにアプリケーションを調整します。このような調整を行うと、クライアントの CPU やネットワークの競合を軽減します。

クライアントのスケーリング

Amazon CloudWatch またはその他のホストメトリクスを使用して、クライアント側が現在 CPU またはネットワーク帯域幅、またはその両方によって制約を受けているかどうかを判断します。クライアント側が制約を受けている場合は、それに応じてクライアントをスケーリングします。

CPU

この待機イベントは、スレッドが CPU でアクティブであるか CPU の待機中に発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、すべての RDS for PostgreSQL のすべてのバージョンに関連しています。

Context

中央処理装置 (CPU)は、命令を実行するコンピュータのコンポーネントです。例えば、CPU 命令は演算処理を実行し、メモリ上でデータを交換します。クエリがデータベースエンジンを介して実行する命令の数が増えると、クエリの実行にかかる時間が長くなります。CPU スケジューリングは、CPU にプロセス時間を与えています。スケジューリングは、OS のカーネルによってオーケストレーションされます。

トピック

- この待機の発生時期を確認する方法
- DbLoadCPU メトリクス
- os.cpuUtilization メトリック
- CPU スケジューリングの原因の可能性

この待機の発生時期を確認する方法

このCPU待機イベントは、バックエンドプロセスが CPU でアクティブであるか、CPU を待っていることを示します。クエリに次の情報が表示されると、発生していることがわかります。

- 「pg_stat_activity.state」 列には値activeがあります。
- pg_stat_activityのwait_event_typeおよびwait_eventの列は、両方ともnullです。

CPU を使用中または待機中のバックエンドプロセスを確認するには、次のクエリを実行します。

```
SELECT *
FROM   pg_stat_activity
```

```
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

DbLoadCPU メトリクス

CPU の Performance Insights のメトリクスは DBLoadCPU です。DBLoadCPUの値は、Amazon CloudWatch メトリクスの値とは異なる場合がありますCPUUtilization。後者のメトリクスは、データベースインスタンスのハイパーバイザーから収集されます。

os.cpuUtilization メトリック

Performance Insights OS のメトリクスは、CPU 使用率に関する詳細情報を提供します。例えば、次のメトリクスを表示できます。

- os.cpuUtilization.nice.avg
- os.cpuUtilization.total.avg
- os.cpuUtilization.wait.avg
- os.cpuUtilization.idle.avg

Performance Insights は、データベースエンジンによる CPU 使用率をos.cpuUtilization.nice.avgのように報告します。

CPU スケジューリングの原因の可能性

オペレーティングシステム (OS) カーネルは、CPU のスケジューリングを処理します。CPU がアクティブな場合、プロセスがスケジューリングされるのを待機する必要がある場合があります。計算の実行中は、CPU はアクティブです。また、実行されていないアイドルスレッド、つまりメモリ I/O の待機中もアクティブになります。このタイプの I/O は、一般的なデータベースワークロードの大部分を占めています。

以下の条件が満たされると、プロセスは CPU でスケジュールされるのを待機する可能性があります。

- CloudWatchCPUUtilizationメトリクスは 100% に近いです。
- 平均ロードは vCPUs の数よりも大きく、ロードが重いことを示しています。このメトリクスは、loadAverageMinutePerformance Insights の OS メトリクスセクションで見ることができます。

待機時間が増加する原因の可能性

CPU 待機イベントが通常よりも頻繁に発生する場合は、パフォーマンスの問題を示している可能性があり、典型的な原因は次のとおりです。

トピック

- 突然のスパイクの原因の可能性
- 長期の高周波の原因の可能性
- コーナーケース

突然のスパイクの原因の可能性

突然のスパイクの原因として最も可能性の高いものは次のとおりです。

- アプリケーションがデータベースへの同時接続を開きすぎています。このシナリオは「接続ストーム」と呼ばれます。
- アプリケーションのワークロードは、次のいずれかの方法で変更されました。
  - 新しいクエリ
  - データセットのサイズの増加
  - インデックスのメンテナンスまたは作成
  - 新しい関数
  - 新しいオペレーター
  - パラレルクエリ実行の増加
- クエリ実行プランが変更されました。場合によっては、変更によってバッファが増加することがあります。例えば、以前はインデックスを使用していたクエリが、現在はシーケンシャルスキャンを使用します。この場合、同じ目標を達成するには、クエリがより多くの CPU を必要とします。

長期の高周波の原因の可能性

長期間にわたって再発するイベントの原因として最も可能性の高いもの:

- CPU で同時に実行されているバックエンドプロセスが多すぎます。これらのプロセスは、パラレルワーカーにすることができます。
- クエリのパフォーマンスは、大量のバッファを必要とするため最適ではありません。

コーナーケース

考えられる原因のいずれも実際の原因ではない場合は、以下のような状況が発生することがあります。

- CPU がプロセスを入れ替えています。
- huge pages 機能が無効になっていると、CPU がページテーブルエントリを管理している可能性があります。このメモリ管理機能は、micro、small、medium 以外のすべての DB インスタンスクラスで、デフォルトでオンになっています。詳細については、「RDS for PostgreSQL の ヒュージページ」を参照してください。

アクション

CPU 待機イベントがデータベースアクティビティを占領している場合でも、必ずしもパフォーマンスの問題を示すわけではありません。パフォーマンスが低下した場合にのみ、このイベントに応答します。

トピック

- データベースが CPU の増加原因かどうかを調べる
- 接続数が増加したかどうかを判断する
- ワークロードの変更に対応

データベースが CPU の増加原因かどうかを調べる

`os.cpuUtilization.nice.avg`Performance Insights のメトリクスを検証します。この値が CPU 使用率よりはるかに小さい場合、データベース以外のプロセスが CPU の主な原因となっています。

接続数が増加したかどうかを判断する

`DatabaseConnections`Amazon CloudWatch のメトリクスを検証します。アクションは、CPU の待機イベントが増加した期間中の数値の増減によって異なります。

接続数が増加した

接続数が増えた場合は、CPU を消費しているバックエンドプロセスの数と vCPUs の数を比較します。以下のシナリオが考えられます。

- CPU を消費するバックエンドプロセスの数が、vCPUs の数より少なくなっています。

この場合、接続数は問題ではありません。ただし、それでも CPU 使用率を下げようとすることがあります。

- CPU を消費するバックエンドプロセスの数が vCPUs の数を超えています。

  このような場合は、以下のオプションを検討します。

  - データベースに接続されているバックエンドプロセスの数を減らします。例えば、RDS Proxy などの接続プーリングソリューションを実装します。詳細についてはAmazon RDS Proxyを参照してください。
  - インスタンスサイズをアップグレードして vCPUs の数を増やします。
  - 一部の読み取り専用ワークロードをリーダーノードにリダイレクトします (該当する場合)。

接続は増加しなかった

`blks_hit`Performance Insights のメトリクスを検証します。`blks_hit`と CPU 使用率の増加の相関関係を探してください。以下のシナリオが考えられます。

- CPU 使用率と`blks_hit`が相関しています。

  この場合、CPU 使用率にリンクされている上位 SQL ステートメントを検索し、プランの変更を検討します。以下のいずれかの対策を使用できます。

  - 計画をマニュアルで説明し、予想される実行プランと比較します。
  - 秒単位のブロックヒット数とローカルブロックヒット数の増加を確認します。Performance Insights ダッシュボードの上位 SQLセクションで、Preferences (設定) を選択します。
- CPU 使用率と`blks_hit`には相関関係がありません。

  このような場合は、次のいずれかに該当するかどうかを判断します。

  - アプリケーションは、データベースとの接続と切断を高速で行っています。

    `log_connections`および`log_disconnections`をオンにして、PostgreSQL のログを分析します。`pgbadger`ログアナライザの使用を検討します。詳細については、「https://github.com/darold/pgbadger」を参照してください。
  - OS はオーバーロード状態です。

    この場合、Performance Insights は、バックエンドプロセスが通常よりも長い時間 CPU を消費していることを示しています。Performance Insights の`os.cpuUtilization`メトリクスサイトまたは`CPUUtilization`CloudWatch のメトリクスでエビデンスを探します。OS がオーバー

ロード状態になっている場合は、拡張モニタリングのメトリックを参照してさらに診断します。具体的には、プロセスリストと各プロセスが消費する CPU の割合を確認します。

- 上位 SQL ステートメントが消費する CPU が多すぎます。

  CPU 使用率とリンクするステートメントを検証し、CPU の使用率を減らせるかどうかを確認します。EXPLAINコマンドを実行し、最も影響が大きいプランノードにフォーカスします。PostgreSQL の実行計画ビジュアライザーの使用を検討してください。このツールを試すには、http://explain.dalibo.com/を参照してください。

ワークロードの変更に対応

ワークロードが変更された場合は、次のタイプの変更を探します。

新しいクエリ

新しいクエリが想定されているかどうかを確認します。その場合は、その実行計画と秒単位の実行数が想定されていることを確認してください。

データセットのサイズの増加

パーティショニングが未実装の場合は、それが役立つかどうかを判断します。この戦略では、クエリで取得する必要があるページ数を減らすことができます。

インデックスのメンテナンスまたは作成

メンテナンスのスケジュールが想定されているかどうかを確認します。ベストプラクティスは、ピークアクティビティ以外のメンテナンスアクティビティをスケジュールすることです。

新しい関数

これらの機能がテスト中に想定したとおりに動作するかどうかを確認します。具体的には、秒単位の実行数が想定されているかどうかを確認します。

新しいオペレーター

テスト中に想定どおりに動作するかどうかを確認します。

パラレルクエリの実行の増加

以下のいずれかの状況が発生するかどうかを確認します。

- 関連する関係やインデックスのサイズが突然大きくなり、min_parallel_table_scan_sizeまたはmin_parallel_index_scan_sizeは大きく異なるようになりました。

- 「parallel_setup_costまたはparallel_tuple_cost」に最近変更が加えられました。
- 「max_parallel_workersまたはmax_parallel_workers_per_gather」に最近変更が加えられました。

IO:BufFileRead および IO:BufFileWrite

IO:BufFileRead と IO:BufFileWrite イベントは、RDS for PostgreSQL がテンポラリファイルを作成するときに発生します。作業メモリパラメータが現在の定義より多くのメモリを必要とするオペレーションは、テンポラリデータを永続的ストレージに書き込みます。この操作は「spilling to disk (ディスクへの流出)」と呼ばれることがあります。一時ファイルとその使用方法の詳細については、「PostgreSQL による一時ファイルの管理」を参照してください。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

IO:BufFileReadそしてIO:BufFileWriteは、作業メモリ領域とメンテナンス作業用メモリ領域に関連します。これらのローカルメモリ領域の詳細については、PostgreSQL ドキュメントの「リソース消費」を参照してください。

デフォルト値は work_mem 4 MB です。一つのセッションがパラレルにオペレーションを実行する場合、パラレル処理を行う各ワーカーは 4 MB のメモリを使用します。このため、work_memを慎重に設定してください。値を大きくしすぎると、多くのセッションを実行しているデータベースがメモリを過剰に消費することがあります。値を低く設定しすぎると、RDS for PostgreSQL はローカルストレージに一時ファイルを作成します。これらのテンポラリファイルのためのディスク I/O により、パフォーマンスが低下する可能性があります。

次のようなイベントが発生する場合、データベースがテンポラリファイルを生成している可能性があります。

1. 可用性の急激な低下
2. 空き領域の高速リカバリ

また、「チェーンソー」のパターンが表示されるかもしれません。このパターンは、データベースが小さなファイルを常に作成していることを示す可能性があります。

待機時間が増加する原因の可能性

一般に、これらの待機イベントは、work_memまたはmaintenance_work_memパラメータが割り当てられるよりも多くのメモリを消費するオペレーションによって発生します。補うために、オペレーションはテンポラリファイルに書き込みます。IO:BufFileReadそしてIO:BufFileWriteイベントの一般的な原因には、次のようなものがあります。

作業用メモリ領域に存在するメモリより多くのメモリを必要とするクエリ

次の特性を持つクエリは、作業メモリ領域を使用します。

- ハッシュ結合
- ORDER BY 句
- GROUP BY 句
- DISTINCT
- ウィンドウ関数
- CREATE TABLE AS SELECT
- REFRESH MATERIALIZED VIEW

メンテナンス作業メモリ領域に存在するメモリより多くのメモリを必要とするステートメント

次のステートメントは、メンテナンス作業メモリ領域を使用します。

- CREATE INDEX
- CLUSTER

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- 問題の特定

- ジョイントクエリを検証する
- ORDER BY クエリと GROUP BY クエリを検証する
- DISTINCT オペレーションの使用を避ける
- GROUP BY 関数の代わりにウィンドウ関数の使用を検討してください。
- マテリアライズドビューと CTAS ステートメントの調査
- インデックスの再構築時に pg_repack を使用する
- テーブルをクラスター化するときに maintenance_work_mem を増やす
- IO:BufFileRead および IO:BufFileWrite を防ぐためにメモリを調整します

問題の特定

一時ファイルの使用状況は、Performance Insights で直接表示できます。詳細については、「Performance Insights を使用した一時ファイルの使用状況の確認」を参照してください。Performance Insights を無効にすると、IO:BufFileRead および IO:BufFileWrite オペレーションが増加することがあります。

問題の原因を特定するには、指定したしきい値 KB を超える一時ファイルを生成するすべてのクエリを口グに記録するように log_temp_files パラメータを設定できます。デフォルトでは、log_temp_files は -1 に設定され、この口ギング機能は無効になります。このパラメータを 0 に設定した場合は、RDS for PostgreSQL はすべての一時ファイルを口グに記録します。値を 1024 に設定した場合、RDS for PostgreSQL は 1 MB を超える一時ファイルを生成するすべてのクエリを口グに記録します。log_temp_filesについての詳細は、PostgreSQL ドキュメントのError reporting and logging を参照してください。

ジョイントクエリを検証する

クエリでは、結合が使用されている可能性があります。例えば、次のクエリは 4 つのテーブルをジョイントします。

```
SELECT *
       FROM "order"
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
```

```
 WHERE customer.id = 1234567890;
```

テンポラリファイル使用量が急増する原因は、クエリ自体の問題の可能性があります。例えば、壊れた節はジョイントを適切にフィルタリングしない可能性があります。次の例では 2 番目の内部ジョイントを考えてみましょう。

```
SELECT *
       FROM "order"
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

前のクエリが誤ってcustomer.idをcustomer.idにジョイントし、すべての顧客とすべての注文の間にデカルト積を生成します。このタイプの偶発的なジョイントは、大きなテンポラリファイルを生成します。テーブルのサイズによっては、デカルトクエリでストレージがいっぱいになることもあります。以下の条件を満たす場合は、アプリケーションにデカルトジョインが生成される場合があります。

- ストレージの可用性が大きく急激に低下し、その後、高速リカバリが起こります。
- インデックスは作成されていません。
- CREATE TABLE FROM SELECTステートメントは発行されていません。
- マテリアライズドビューはリフレッシュされません。

テーブルが適切なキーを使用してジョイントされているかどうかを確認するには、クエリおよびオブジェクト関係マッピングディレクティブを調べます。アプリケーションの特定のクエリは常に呼び出されるわけではなく、一部のクエリは動的に生成されることに注意してください。

ORDER BY クエリと GROUP BY クエリを検証する

場合によっては、ORDER BY節を使用するとテンポラリファイルが過剰になる可能性があります。以下のガイドラインを検討します。

- 順序付けが必要な場合のみ、ORDER BYに列を含めてください。このガイドラインは、数千行を返し、ORDER BY節で多数の列を指定するクエリでは特に重要です。

- ORDER BY節が同じ昇順または降順の列にマッチする場合、高速化するためにインデックスの作成を検討します。パーシャルインデックスのほうが小さいため好ましいです。小さいインデックスは、より迅速に読み込まれ、トラバースされます。
- NULL 値を受け入れることができる列のインデックスを作成する場合は、NULL 値をインデックスの最後に格納するか、先頭に格納するかを検討します。

  可能であれば、結果セットをフィルタリングして、順序付けが必要な行の数を減らします。WITH節ステートメントまたはサブクエリを使用する場合、内部クエリが結果セットを生成し、外部クエリに渡すことに注意してください。クエリが行をより多くフィルタリングすると、クエリが行う必要がある順序付けは減ります。
- 完全な結果セットを取得する必要がない場合は、LIMIT節を使用します。例えば、上位 5 行だけが必要な場合、LIMIT節を使用したクエリは結果を生成し続けることはありません。このように、クエリに必要なメモリとテンポラリファイルが減ります。

GROUP BY 句を使用するクエリは、テンポラリファイルを要求することもできます。GROUP BY クエリは、次のような関数を使用して値を要約します。

- COUNT
- AVG
- MIN
- MAX
- SUM
- STDDEV

GROUP BYクエリをチューニングするには、ORDER BYクエリの推奨事項に従ってください。

DISTINCT オペレーションの使用を避ける

可能であれば、DISTINCTオペレーションを使用して重複した行を削除することは避けてください。クエリが返す行が不要かつ重複していればいるほど、VDISTINCTオペレーションのコストは高くなります。可能であれば、異なるテーブルに対して同じフィルターを使用している場合でも、WHERE節でフィルターを追加してください。クエリをフィルタリングして正しく結合すると、パフォーマンスが向上し、リソースの使用量が削減されます。また、誤ったレポートや結果を防ぐことができます。

DISTINCTを同じテーブルの複数の行に使用する必要がある場合、複合インデックスの作成を検討してください。インデックスに複数の列をグループ化すると、個別の行を評価する時間を短縮できま

す。また、RDS for PostgreSQL バージョン 10 以降を使用している場合は、CREATE STATISTICS コマンドを使用して複数の列間で統計を関連付けられます。

GROUP BY 関数の代わりにウィンドウ関数の使用を検討してください。

GROUP BYを使用すると、結果セットを変更し、集計結果を取得できます。ウィンドウ関数を使用すると、結果セットを変更せずにデータを集計できます。ウィンドウ関数は、OVER句を使用して、クエリによって定義されたセット間で計算を実行し、ある行を別の行に関連付けます。ウィンドウ関数に含まれるすべてのGROUP BY関数は使用できますが、次のような関数も使用可能です。

- RANK
- ARRAY_AGG
- ROW_NUMBER
- LAG
- LEAD

ウィンドウ関数によって生成されるテンポラリファイルの数を最小限に抑えるには、2 つの異なる集計が必要な場合は同じ結果セットの重複を削除してください。次のクエリについて考えます。

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

WINDOW節のクエリは、次のように書き換えることができます。

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

デフォルトでは、RDS for PostgreSQL 実行プランナーは類似したノードを統合し、オペレーションが重複しないようにします。ただし、ウィンドウブロックに明示的な宣言を使用すると、クエリをより簡単に維持できます。また、重複を防止するとパフォーマンスの向上につながることがあります。

マテリアライズドビューと CTAS ステートメントの調査

マテリアライズドビューがリフレッシュされると、クエリが実行されます。このクエリには、GROUP BY、ORDER BY、DISTINCTのような操作を含めることができます。リフレッシュ中に、大量のテン

ポラリファイルや待機イベントIO:BufFileWriteおよびIO:BufFileReadが発生することがあります。同様に、SELECTに基づいてテーブルを作成すると、CREATE TABLEステートメントはクエリを実行します。必要なテンポラリファイルを減らすには、クエリを最適化します。

インデックスの再構築時に pg_repack を使用する

インデックスを作成すると、エンジンは結果セットを順序付けます。テーブルのサイズが大きくなり、インデックスで指定された列の値が多様化していくと、テンポラリファイルはより多くの領域を必要とします。ほとんどの場合、メンテナンス作業のメモリ領域を変更しなければ、大きなテーブルのテンポラリファイルの作成を防ぐことはできません。maintenance_work_mem の詳細については、PostgreSQL のドキュメントの「https://www.postgresql.org/docs/current/runtime-config-resource.html」を参照してください。

大きなインデックスを再作成するときに考えられる回避策としては、pg_repack 拡張機能を使用することが挙げられます。詳細については、「pg_repack のドキュメント」で「最小限のロックで PostgreSQL データベース内のテーブルを再編成する」を参照してください。RDS for PostgreSQL DB インスタンスに対する拡張機能の設定については、「pg_repack 拡張機能を使用して、テーブルやインデックスの膨張を抑制する」を参照してください。

テーブルをクラスター化するときに maintenance_work_mem を増やす

CLUSTERコマンドは、index_nameで指定した既存のインデックスに基づいて、table_nameで指定したテーブルをクラスター化します。RDS for PostgreSQL は、指定されたインデックスの順序に一致するようにテーブルを物理的に再作成します。

磁気ストレージが普及していたころは、ストレージのスループットが限られていたため、クラスター化が一般的でした。今では、SSD ベースのストレージが一般的となり、クラスター化はあまり一般的ではなくなっています。ただし、テーブルをクラスター化すると、テーブルのサイズ、インデックス、クエリなどによってパフォーマンスが多少向上することがあります。

CLUSTERコマンドを実行して、待機イベントIO:BufFileWrite、IO:BufFileReadをモニタリングし、maintenance_work_memをチューニングします。メモリサイズをかなり大きくしてください。高い値は、エンジンがクラスター化オペレーションのためにより多くのメモリを使用できることを意味します。

IO:BufFileRead および IO:BufFileWrite を防ぐためにメモリを調整します

状況によっては、メモリのチューニングが必要です。以下のような適切なパラメータを使用して、消費領域にわたってメモリのバランスを取ることが目的です。

- work_mem 値。

- shared_buffers を割り引いた後の残りのメモリ
- max_connectionsで制限されるオープンおよび使用中の最大接続数

これらのメモリのチューニングの詳細については、PostgreSQL ドキュメントの「リソース消費」を参照してください。

作業メモリ領域のサイズを拡大する

状況によっては、セッションで使用されるメモリを増やすことが唯一の選択肢となることもあります。クエリが正しく記述され、ジョイントに正しいキーを使用している場合は、work_mem値の増加を検討してください。

クエリが生成するテンポラリファイルの数を調べるには、log_temp_files を 0 に設定します。work_mem 値を口グで識別される最大値まで上げると、クエリでテンポラリファイルが生成されるのを防ぎます。ただし、work_memは各接続またはパラレルワーカーにプランノードあたりの最大値を設定します。データベースに 5,000 の接続があり、それぞれが 256 MiB のメモリを使用する場合、エンジンは 1.2 TiB の RAM を必要とします。そのため、インスタンスのメモリが不足する可能性があります。

共有バッファプールに十分なメモリを予約する

データベースでは、作業用メモリ領域だけでなく、共有バッファプールなどのメモリ領域が使用されます。work_memを増加する前に、これらの追加メモリ領域の要件を考慮してください。

例えば、RDS for PostgreSQL インスタンスクラスが db.r5.2xlarge であると仮定します。このクラスには 64 GiB のメモリがあります。デフォルトでは、メモリの 25% が共有バッファプール用に予約されています。共有メモリ領域に割り当てられた量を引くと、16,384 MB が残ります。OS やエンジンもメモリを必要とするため、残りのメモリを作業メモリ領域にのみ割り当てないでください。

work_memに割り当て可能なメモリはインスタンスクラスによって異なります。より大きなインスタンスクラスを使用すると、より多くのメモリが使用できます。ただし、前の例では 16 GiB 以上は使用できません。そうでなければ、メモリ不足に陥ったときにインスタンスが使用できなくなります。インスタンスを利用できない状態から回復するには、RDS for PostgreSQL オートメーションサービスが自動的に再起動します。

接続の数を管理する

データベースインスタンスでの同時接続が 5,000 とします。各接続では、work_memのうち少なくとも 4 MiB を使用します。接続に必要なメモリ消費量が多いと、パフォーマンスが低下する可能性があります。これに対して、次のオプションがあります。

- より大きなインスタンスクラスにアップグレードします。
- 接続プロキシまたはプーラーを使用することで、データベースの同時接続の数を減らします。

プロキシの場合は、アプリケーションに基づいて Amazon RDS プロキシ、pgBouncer、または接続プーラーを検討してください。この解決策は CPU ロードを軽減します。また、すべての接続が作業メモリ領域を必要とする場合のリスクも軽減します。データベース接続数が少ない場合は、work_memの値を増やすことができます。このように、IO:BufFileReadそしてIO:BufFileWrite待機イベントの発生を減らします。また、作業メモリ領域で待っているクエリが大幅に高速化します。

IO:DataFileRead

IO:DataFileReadイベントは、バックエンドプロセスが必要なページを読み込む際に、ページが共有メモリで使用できないため接続が待機したときに発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

すべてのクエリおよびデータ操作 (DML) オペレーションは、バッファプール内のページにアクセスします。読み取りを誘発できるステートメントには、SELECT、UPDATE、DELETEがあります。例えば、UPDATEは、テーブルまたはインデックスからページを読み取ることができます。要求または更新中のページが共有バッファプールにない場合、この読み取りはIO:DataFileReadイベントにつながることがあります。

共有バッファプールは有限のため、いっぱいになる可能性があります。この場合、メモリ上にないページをリクエストすると、データベースは強制的にディスクからブロックを読み取ることになります。IO:DataFileReadイベントが頻繁に発生する場合は、共有バッファプールが小さすぎるとワー

クロードに対応できない可能性があります。この問題は、バッファプールに収まらない多数の行読み取るSELECTクエリでは深刻です。バッファプールの詳細については、PostgreSQL ドキュメントの「リソース消費」を参照してください。

待機時間が増加する原因の可能性

IO:DataFileReadイベントの一般的な原因は以下のとおりです。

接続スパイク

複数の接続で同じ数の IO:DataFileRead 待機イベントが発生することがあります。この場合、スパイク (突然大きく増加) が IO:DataFileRead イベントで発生する可能性があります。

シーケンシャルスキャンを実行する SELECT および DML ステートメント

アプリケーションが新しいオペレーションを実行している可能性があります。または、新しい実行計画のために既存の操作がオペレーションされる可能性があります。このような場合は、seq_scan値より大きいテーブル (特に大きなテーブル) を探します。pg_stat_user_tablesクエリでそれらを探してください。より多くの読み取りオペレーションを生成しているクエリを追跡するには、エクステンションpg_stat_statementsを使用します。

大規模なデータセットの CTAS および CREATE INDEX

CTASはCREATE TABLE AS SELECTステートメントです。大規模なデータセットを出典として使用して CTAS を実行する場合、または大きなテーブルにインデックスを作成する場合は、IO:DataFileReadイベントが発生する可能性があります。インデックスを作成するとき、データベースはシーケンシャルスキャンを使用してオブジェクト全体を読み取る必要があります。CTAS は、ページがメモリ上にないときにIO:DataFileリードを生成します。

複数のバキュームワーカーが同時に実行されている

バキュームワーカーは、マニュアルまたは自動でトリガーできます。積極的なバキューム戦略の採用をお勧めします。ただし、テーブルに多数の更新または削除された行がある場合、IO:DataFileRead待機が増加します。スペース確保後、IO:DataFileReadに費やすバキューム時間が減少します。

大量データの取り込み

アプリケーションで大量のデータを取り込むと、ANALYZEオペレーションが頻繁に発生する可能性があります。ANALYZEプロセスは、オートバキュームランチャーによって、あるいはマニュアルでトリガーすることができます。

ANALYZEオペレーションは、テーブルのサブセットを読み取ります。30 にdefault_statistics_target値を掛けたものがスキャンを要するページ数です。詳細については、PostgreSQL ドキュメントをご参照ください。default_statistics_targetパラメータは 1 ~ 10,000 の範囲の値を指定でき、デフォルトは 100 です。

リソースの枯渇

インスタンスのネットワーク帯域幅や CPU が消費されると、IO:DataFileReadイベントはより頻繁に発生する可能性があります。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- 待機を生成するクエリの述語フィルターをチェックする
- メンテナンス作業の影響を最小化する
- 多数の接続に対応する

待機を生成するクエリの述語フィルターをチェックする

IO:DataFileRead待機イベントを生成する特定のクエリを特定するとします。これらは、次の方法を使用して識別できることがあります。

- Performance Insights
- エクステンションpg_stat_statementsで提供されるようなカタログビュー
- カタログビューpg_stat_all_tablesで、定期的な物理読み取り回数の増加を示す場合
- pg_statio_all_tablesビューで、_readカウンターの増加が示されている場合

これらのクエリの述語 (WHERE 節) でどのフィルターが使用されるかを決定することをお勧めします。次のガイドラインに従ってください:

- EXPLAIN コマンドを実行します。出力では、使用されているスキャンのタイプを特定します。シーケンシャルスキャンは必ずしも問題を示すわけではありません。シーケンシャルスキャンを使用するクエリは、フィルターを使用するクエリと比較して、自然により多くのIO:DataFileReadイベントを生成します。

WHERE節に記載された列がインデックスされているかどうかを確認します。されていない場合、この列のインデックスの作成を検討してください。この方法では、シーケンシャルスキャンを回避し、IO:DataFileReadイベントの発生を減らすことができます。制限付きフィルターがあってもシーケンシャルスキャンが実行される場合は、適切なインデックスが使用されているかどうかを評価します。

- クエリが非常に大きなテーブルにアクセスしているかどうかを確認します。場合によっては、テーブルをパーティション化するとクエリで必要なパーティションのみを読み取ることができ、パフォーマンスが向上することがあります。
- ジョイント操作からカーディナリティ (行の合計数) を検証します。フィルターに渡すWHERE節の値がどれほど制限的であるかに注意してください。可能であれば、クエリをチューニングして、計画の各ステップで渡される行数を減らします。

メンテナンス作業の影響を最小化する

VACUUMやANALYZEのようなメンテナンスオペレーションは重要です。これらのメンテナンス作業に関連するIO:DataFileRead待機イベントを見つけても、それらをオフにしないことをお勧めします。次のようなアプローチにより、これらの操作の影響を最小限に抑えることができます。

- オフピーク時にメンテナンス操作をマニュアルで実行します。この方法では、データベースが自動操作のしきい値に達するのを防ぎます。
- 非常に大きなテーブルの場合は、テーブルのパーティション化を検討してください。この方法により、メンテナンスオペレーションのオーバーヘッドが削減されます。データベースは、メンテナンスが必要なパーティションにのみアクセスします。
- 大量のデータを取り込む場合は、自動分析機能を無効にすることを検討してください。

オートバキューム機能は、次の数式が真の場合、テーブルに対して自動的にトリガーされます。

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

ビューpg_stat_user_tablesとカタログpg_classには複数の行があります。1 行は、テーブル内の 1 つの行に対応できます。この公式は、reltuplesが特定のテーブル用だと仮定しています。パラメータ autovacuum_vacuum_scale_factor (デフォルトは 0.20) と autovacuum_vacuum_threshold (デフォルトでは 50 タプル) は通常、インスタンス全体に対してグローバルに設定されます。ただし、特定のテーブルに対して異なる値を設定できます。

トピック

- 不要な領域を消費しているテーブルを探す
- 不要な領域を消費しているインデックスを探す
- オートバキュームの対象となるテーブルを見つける

不要な領域を消費しているテーブルを探す

不必要に領域を消費しているテーブルを探すには、PostgreSQL pgstattuple 拡張機能の関数を使用できます。この拡張機能 (モジュール) は、すべての RDS for PostgreSQL DB インスタンスにデフォルトで使用でき、次のコマンドを使用してインスタンス化できます。

```
CREATE EXTENSION pgstattuple;
```

この拡張機能の詳細については、PostgreSQL ドキュメントの「pgstattuple」を参照してください。

アプリケーション内のテーブルとインデックスの肥大化をチェックできます。詳細については、「テーブルとインデックスの肥大化の診断」を参照してください。

不要な領域を消費しているインデックスを探す

肥大化したインデックスを探し、読み取り権限のあるテーブルで不必要に消費されている領域の大きさを推定するには、次のクエリを実行します。

```
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
```

```
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
 MAXALIGN
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
            ELSE nulldatawidth::integer%maxalign
          END
      )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
      -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
    FROM (
      SELECT
        i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
 AS table_oid,
        current_setting('block_size')::numeric AS bs, fillfactor,
        CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
          WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
 THEN 8
          ELSE 4
        END AS maxalign,
        /* per page header, fixed size: 20 for 7.X, 24 for others */
        24 AS pagehdr,
        /* per page btree opaque data */
        16 AS pageopqdata,
        /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
        CASE WHEN max(coalesce(s.null_frac,0)) = 0
          THEN 2 -- IndexTupleData size
```

```
          ELSE 2 + (( 32 + 8 - 1 ) / 8)
          -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
 index + 8 - 1 /8)
       END AS index_tuple_hdr_bm,
       /* data len: we remove null values save space using it fractionnal part from
 stats */
       sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
 nulldatawidth,
       max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
 AS is_na
     FROM pg_attribute AS a
       JOIN (
         SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
           idx.reltuples, idx.relpages, idx.relam,
           indrelid, indexrelid, indkey::smallint[] AS attnum,
           coalesce(substring(
             array_to_string(idx.reloptions, ' ')
              from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
         FROM pg_index
           JOIN pg_class idx ON idx.oid=pg_index.indexrelid
           JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
           JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
         WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
       ) AS i ON a.attrelid = i.indexrelid
       JOIN pg_stats AS s ON s.schemaname = i.nspname
         AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
          -- stats from tbl
          OR  (s.tablename = i.idxname AND s.attname = a.attname))
          -- stats from functional cols
       JOIN pg_type AS t ON a.atttypid = t.oid
     WHERE a.attnum > 0
     GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
    ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

オートバキュームの対象となるテーブルを見つける

自動バキュームの対象となるテーブルを見つけるには、次のクエリを実行します。

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
         coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
         as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
```

```
)
ORDER BY age(relfrozenxid) DESC;
```

多数の接続に対応する

Amazon CloudWatch をモニタリングすると、DatabaseConnectionsメトリックスパイクが見つかることがあります。この増加は、データベースへの接続数が増加していることを示します。次のようなアプローチを推奨します。

- アプリケーションが各インスタンスで開くことができる接続の数を制限します。アプリケーションが組み込み接続プール機能を備えている場合は、適切な数の接続を設定します。インスタンス内の vCPUs が効果的にパラレル化できる数値を基準にします。

  アプリケーションで接続プール機能を使用しない場合は、Amazon RDS プロキシまたは代替の使用を検討してください。このアプローチにより、アプリケーションはロードバランサーとの複数の接続を開くことができます。その後、バランサーは、データベースとの制限された数の接続を開くことができます。パラレルで実行される接続が少なくなると、DB インスタンスのカーネル内のコンテキスト切り替えが減少します。クエリの進行が速くなり、待機イベントが減少するはずです。詳細については、「Amazon RDS Proxy」を参照してください。
- 可能であれば、RDS for PostgreSQL のリードレプリカを活用してください。アプリケーションが読み取り専用のオペレーションを実行するときは、これらのリクエストを読み取り専用レプリカに送信します。この方法でプライマリ (ライター) ノードの I/O 負荷を軽減します。
- DB インスタンスのスケールアップを検討します。大容量のインスタンスクラスはより多くのメモリを提供するため、RDS for PostgreSQL ではページを保持するためのより大きな共有バッファプールを提供します。サイズが大きければ、DB インスタンスが接続処理する vCPUs も多くなります。特に、IO:DataFileRead待機イベントを発生させているオペレーションが書き込みの場合、vCPU の増設は有効です。

IO:WALWrite

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL 10 以降のすべてのバージョンでサポートされています。

Context

先行書き込みログデータを生成しているデータベース内のアクティビティは、最初に WAL バッファをいっぱいにし、次に非同期でディスクに書き込みます。待機イベント `IO:WALWrite` は、トランザクションの COMMIT 呼び出しを解放できるように、WAL データのディスクへの書き込みが完了するまで SQL セッションの待機中に生成されます。

待機時間が増加する原因の可能性

この待機イベントが頻繁に発生する場合は、ワークロードが実行する更新の種類とその頻度を確認する必要があります。特に、次のタイプのアクティビティを確認します。

高負荷の DML アクティビティ

データベーステーブルのデータは、すぐに変更されるわけではありません。あるテーブルへの挿入は、別のクライアントからの同じテーブルへの挿入または更新を待つ必要がある場合があります。データ操作言語 (DML) のステートメントによってデータ値 (INSERT、UPDATE、DELETE、COMMIT、ROLLBACK TRANSACTION) を変更したことで競合が発生し、先行書き込みログファイルがバッファのフラッシュを待つ可能性があります。この状況は、以下の Amazon RDS Performance Insights メトリクスにも表れており、高負荷の DML アクティビティを示しています。

- `tup_inserted`
- `tup_updated`
- `tup_deleted`
- `xact_rollback`
- `xact_commit`

これらのメトリクスの詳細については、「Amazon RDS for PostgreSQL の Performance Insights カウンター」を参照してください。

チェックポイントアクティビティの頻度

チェックポイントが頻繁に発生すると、WAL ファイルの数が増えます。RDS for PostgreSQL では、ページ全体の書き込みは常に「オン」になっています。ページ全体への書き込みは、デー

タ損失の防止に役立ちます。ただし、チェックポイントが頻繁に行われると、システム全体のパフォーマンスの問題が発生することがあります。これは、特に 高負荷の DML アクティビティのシステムに当てはまります。場合によっては、postgresql.log に「チェックポイントが頻繁に発生しています」という エラーメッセージが表示されることがあります。

チェックポイントをチューニングする際には、異常なシャットダウンが発生した場合に予想される復旧時間と、パフォーマンスとのバランスを慎重に取ることをお勧めします。

アクション

この待機イベントの数を削減するには、次のアクションをお勧めします。

トピック

- コミットの回数を減らす
- チェックポイントのモニタリング
- IO のスケールアップ
- 専用ログボリューム (DLV)

コミットの回数を減らす

コミット数を減らすには、ステートメントをトランザクションブロックにまとめることができます。Amazon RDS Performance Insights を使用して、実行されているクエリの種類を確認してください。また、大規模なメンテナンスオペレーションをオフピークの時間帯に移動することもできます。例えば、インデックスを作成したり、稼働時間外に pg_repack オペレーションを使用したりします。

チェックポイントのモニタリング

RDS for PostgreSQL DB インスタンスがチェックポイント用に WAL ファイルに書き込む頻度をモニタリングできるパラメータが 2 つあります。

- log_checkpoints – このパラメータはデフォルトで「オン」になっています。これにより、チェックポイントごとにメッセージが PostgreSQL ログに送信されます。これらのログメッセージには、書き込まれたバッファの数、書き込みにかかった時間、特定のチェックポイントで追加、削除、またはリサイクルされた WAL ファイルの数が含まれます。

  このパラメータについての詳細は、PostgreSQL ドキュメントの「エラー報告とログ記録」を参照してください。

- checkpoint_warning – このパラメータは、チェックポイント頻度のしきい値 (秒単位) を設定します。この値を超えると、警告が表示されます。デフォルトでは、このパラメータは RDS for PostgreSQL では設定されていません。このパラメータの値を設定すると、RDS for PostgreSQL DB インスタンスのデータベース変更が WAL ファイルのサイズが処理できない速度で書き込まれたときに警告を受け取ることができます。例えば、このパラメータを 30 に設定したとします。RDS for PostgreSQL インスタンスが 30 秒に 1 回以上の頻度で変更を書き込む必要がある場合、「チェックポイントが頻繁に発生しています」という警告が PostgreSQL ログに送信されます。これは、max_wal_size 値を増やす必要があることを示している可能性があります。

  詳細については、PostgreSQL ドキュメントの「ログ先行書き込み」を参照してください。

IO のスケールアップ

このタイプの入出力 (IO) 待機イベントは、1 秒あたりの入出力オペレーション (IOPS) をスケーリングして IO を高速化することで修正できます。CPU をスケーリングするよりも IO をスケーリングする方が望ましいです。CPU をスケーリングすると、処理量が増えることで IO ボトルネックがさらに悪化するため、IO の競合がさらに増える可能性があります。一般的に、スケーリングを実行する前にワークロードのチューニングを検討することをお勧めします。

専用ログボリューム (DLV)

Amazon RDS コンソール、AWS CLI、または Amazon RDS API を使用して、プロビジョンド IOPS (PIOPS) ストレージを使用する DB インスタンスの専用ログボリューム (DLV) を使用できます。DLV は、PostgreSQL データベーストランザクションログを、データベーステーブルを含むボリュームとは別のストレージボリュームに移動します。詳細については、「専用ログボリューム (DLV)」を参照してください。

IPC:parallel 待機イベント

以下の IPC:parallel wait events は、セッションが並列クエリ実行オペレーションに関連するプロセス間通信を待機していることを示します。

- IPC:BgWorkerStartup - プロセスは、並列ワーカープロセスが起動シーケンスを完了するのを待機しています。これは、並列クエリ実行のためにワーカーを初期化するときに発生します。
- IPC:BgWorkerShutdown - プロセスは、並列ワーカープロセスがシャットダウンシーケンスを完了するのを待機しています。これは、並列クエリ実行のクリーンアップフェーズ中に発生します。

- `IPC:ExecuteGather` - プロセスは、クエリの実行中に並列ワーカープロセスからデータを受信するのを待機しています。これは、リーダープロセスがワーカーから結果を収集する必要がある場合に発生します。
- `IPC:ParallelFinish` - プロセスは、並列ワーカーが実行を終了して最終結果を報告するのを待機しています。これは、並列クエリ実行の完了フェーズ中に発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、Aurora PostgreSQL のすべてのバージョンでサポートされています。

Context

PostgreSQL での並列クエリの実行では、複数のプロセスが連携して単一のクエリを処理します。クエリが並列化に適していると判断されると、リーダープロセスは `max_parallel_workers_per_gather` パラメータ設定に基づいて 1 つ以上の並列ワーカープロセスと連携します。リーダープロセスはワーカー間に作業を分割し、各ワーカーは担当するデータ部分を個別に処理します。結果はリーダープロセスに収集されます。

> **Note**
>
> 各並列ワーカーは、フルユーザーセッションと同様のリソース要件を持つ個別のプロセスとして動作します。並列クエリで 4 つのワーカーが連携する場合、リーダープロセスと各ワーカープロセスの両方が独自のリソース割り当てを維持するため、非並列クエリと比較してリソース (CPU、メモリ、I/O 帯域幅) を最大 5 倍消費する可能性があります。例えば、`work_mem` のような設定は各ワーカーに個別に適用されるため、すべてのプロセスにわたって合計メモリ使用量が乗算される可能性があります。

並列クエリアーキテクチャは、以下の 3 つの主要コンポーネントで構成されます。

- リーダープロセス: 並列オペレーションを開始して、ワークロードを分割し、ワーカープロセスと連携するメインプロセス。
- ワーカープロセス: クエリの一部を並行して実行するバックグラウンドプロセス。
- ギャザー/ギャザーマージ: 複数のワーカープロセスの結果をリーダーに戻してまとめるオペレーション。

並列実行中、プロセスはプロセス間通信 (IPC) メカニズムを介して相互に通信する必要があります。これらの IPC 待機イベントは、以下のさまざまなフェーズで発生します。

- ワーカーの起動: 並列ワーカーを初期化するとき
- データ交換: ワーカーがデータを処理し、結果をリーダーに送信するとき
- ワーカーのシャットダウン: 並列実行が完了し、ワーカーを終了するとき
- 同期ポイント: プロセス間で連携する必要があるとき、または他のプロセスがタスクを完了するまで待機する必要があるとき

これらの待機イベントを理解することは、特に複数の並列クエリが同時に実行中となる可能性が高い同時実行環境において、並列クエリの実行に関連するパフォーマンスの問題を診断するために重要です。

待機時間が増加する原因の可能性

並列関連の IPC 待機イベントの増加には、いくつかの要因が関係します。

並列クエリに伴う高い同時実行性

多数の並列クエリを同時に実行すると、リソースの競合が発生し、IPC オペレーションの待機時間が長くなる可能性があります。これは、トランザクション量が多いシステムや分析ワークロードで特によく見られます。

最適ではない並列クエリプラン

クエリプランナーで非効率的な並列プランを選択すると、ワーカー間での不要な並列化や作業分散の低下につながる可能性があります。その結果、特に `IPC:ExecuteGather` イベントと `IPC:ParallelFinish` イベントで、IPC 待機時間が長くなることがあります。こうしたプランニングの問題は、多くの場合、古い統計情報やテーブル/インデックスの肥大化が原因です。

並列ワーカーの頻繁な起動とシャットダウン

有効期間の短いクエリによる並列ワーカーの頻繁な開始と終了は、IPC:BgWorkerStartup イベントと IPC:BgWorkerShutdown イベントの増加につながる可能性があります。これは、多くの小さな並列化可能なクエリを含む OLTP ワークロードでよく見られます。

リソースの制約

CPU、メモリ、または I/O 容量が制限されると、並列実行でボトルネックが発生し、すべての IPC イベントで待機時間が長くなる可能性があります。例えば、CPU が飽和していると、ワーカープロセスが作業の一部を起動または処理するのに時間がかかることがあります。

複雑なクエリ構造

複数レベルの並列処理 (並列結合の後に並列集約が続くなど) を含むクエリは、特に IPC:ExecuteGather イベントの場合、IPC パターンが複雑化し、待機時間が長くなる可能性があります。

大きな結果セット

結果セットが大きいクエリでは、リーダープロセスがワーカープロセスからの結果の収集と処理により多くの時間を費やすため、IPC:ExecuteGather の待機時間が長くなる可能性があります。

これらの要因を理解することは、Aurora PostgreSQL での並列クエリ実行に関連するパフォーマンスの問題の診断と対処に役立ちます。

アクション

並列クエリに関連する待機が生じた場合は、通常、バックエンドプロセスが並列ワーカープロセスを連携中または待機中であることを意味します。これらの待機は、並列プランの実行中によく発生します。これらの待機の影響を調査して軽減するには、並列ワーカーの使用状況をモニタリングし、パラメータ設定を見直して、クエリの実行とリソース割り当てを調整します。

トピック

- 非効率的な並列処理のクエリプランを分析する
- 並列クエリの使用状況をモニタリングする
- 並列クエリ設定を確認して調整する
- リソース割り当てを最適化する
- 接続管理を調査する

- メンテナンスオペレーションを確認して最適化する

非効率的な並列処理のクエリプランを分析する

並列クエリを実行すると、システムの不安定性、CPU 使用率の急上昇、予測不可能なクエリパフォーマンスの変動が発生しがちです。並列処理によって特定のワークロードが実際に改善されるかどうかを徹底的に分析することが重要です。EXPLAIN ANALYZE を使用して、並列クエリ実行プランを確認してください。

プランの効率を比較するには、セッションレベルで並列処理を一時的に無効にします。

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

並列処理を再度有効にして比較します。

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

並列処理を無効にすると、より良い結果またはより一貫した結果が得られる場合は、SET コマンドを使用してセッションレベルで特定のクエリに対して並列処理を無効にすることを検討してください。より広範な効果を得るには、DB パラメータグループの関連するパラメータを調整して、インスタンスレベルで並列処理を無効にすることもできます。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

並列クエリの使用状況をモニタリングする

以下のクエリを使用して、並列クエリのアクティビティと容量を可視化します。

アクティブな並列ワーカープロセスを確認します。

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
```

```
    backend_type = 'parallel worker';
```

このクエリは、アクティブな並列ワーカープロセスの数を示します。値が高い場合は、`max_parallel_workers` の設定値が高い可能性があるため、値を下げることを検討できます。

同時並列クエリを確認します。

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

このクエリは、並列クエリを起動した個別のリーダープロセスの数を返します。ここで数値が高い場合は、複数のセッションで並列クエリを同時に実行中であることを示しているため、CPU とメモリの需要が増加する可能性があります。

並列クエリ設定を確認して調整する

以下のパラメータがワークロードに適合していることを確認します。

- `max_parallel_workers`: すべてのセッションにわたる並列ワーカーの合計数。
- `max_parallel_workers_per_gather`: クエリあたりの最大ワーカー数。

OLAP ワークロードの場合、これらの値を増やすと、パフォーマンスが向上する可能性があります。OLTP ワークロードの場合は、一般的に値を低く設定することが推奨されます。

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

リソース割り当てを最適化する

CPU 使用率をモニタリングし、値が一貫して高い場合や、アプリケーションが並列クエリのメリットを得られる場合は、vCPU の数を調整することを検討してください。並列オペレーションに十分なメモリが使用可能であることを確認します。

- Performance Insights メトリクスを使用して、システムが CPU の処理能力に制約されているかどうかを判断します。
- 各並列ワーカーは独自の work_mem を使用します。合計メモリ使用量がインスタンスの制限内であることを確認します。

並列クエリは、非並列クエリよりもかなり多くのリソースを消費する可能性があります。各ワーカープロセスは完全に独立したプロセスであり、別のユーザーセッションを追加するのとほぼ同じ影響をシステムにもたらすためです。この設定の値を選択したり、リソース使用率を制御する他の設定 (work_mem など) を構成したりする場合は、この点を考慮に入れる必要があります。詳細については、PostgreSQL ドキュメントをご参照ください。work_mem などのリソース制限は各ワーカーに個別に適用されるため、すべてのプロセスにわたる合計使用率は、単一のプロセスにおける通常の使用率よりも大幅に高くなる可能性があります。

ワークロードを高度に並列化する場合は、vCPU を増やすか、メモリパラメータを調整することを検討してください。

接続管理を調査する

接続が不足している場合は、アプリケーション接続プーリング戦略を見直します。接続プーリングをまだ使用していない場合は、アプリケーションレベルで実装することを検討してください。

メンテナンスオペレーションを確認して最適化する

インデックスの作成やその他のメンテナンスタスクを調整して、リソースの競合を防ぎます。これらのオペレーションは、オフピーク時間帯にスケジュールすることを検討してください。ユーザークエリの負荷が高い間は、負荷の高いメンテナンス (並列インデックスのビルドなど) をスケジュールしないでください。これらのオペレーションは並列ワーカーを消費し、通常のクエリのパフォーマンスに影響を与える可能性があります。

IPC:ProcArrayGroupUpdate

IPC:ProcArrayGroupUpdate イベントは、セッションが、グループリーダーによるオペレーション終了時のトランザクションステータスの更新を待機しているときに発生します。PostgreSQL は通常、IPC タイプの待機イベントを並列クエリオペレーションに関連付けますが、この特定の待機イベントは並列クエリに固有ではありません。

トピック

- サポート対象エンジンバージョン

- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

プロセス配列の理解 – プロセス (proc) 配列は PostgreSQL の共有メモリ構造です。トランザクションの詳細など、実行中のすべてのプロセスに関する情報を保持します。トランザクションの完了時 (`COMMIT` または `ROLLBACK`) に、ProcArray を更新して変更を反映し、配列から transactionID をクリアする必要があります。トランザクションを完了しようとするセッションは、ProcArray で排他的ロックを取得する必要があります。これにより、他のプロセスが共有ロックまたは排他的ロックを取得できなくなります。

グループ更新メカニズム – COMMIT または ROLLBACK の実行中に、バックエンドプロセスが排他的モードで ProcArrayLock を取得できない場合、ProcArrayGroupMember という特別なフィールドを更新します。これにより、終了する予定のセッションのリストにトランザクションが追加されます。このバックエンドプロセスはスリープし、スリープ時間は ProcArrayGroupUpdate 待機イベントとして計測されます。procArrayGroupMember を使用した ProcArray の最初のプロセスはリーダープロセスと呼ばれ、ProcArrayLock を排他的モードで取得します。次に、グループ transactionID のクリアを待機しているプロセスのリストをクリアします。これが完了すると、リーダーは ProcArrayLock をリリースし、このリストのすべてのプロセスを起動して、トランザクションが完了したことを通知します。

待機時間が増加する原因の可能性

実行中のプロセスが多いほど、リーダーが排他的モードで procArrayLock を保持する時間が長くなります。したがって、グループ更新シナリオでより多くの書き込みトランザクションが発生すると、`ProcArrayGroupUpdate` 待機イベントを待っているプロセスが蓄積される可能性があります。Database Insights のトップ SQL ビューでは、COMMIT がこの待機イベントの大部分を持つステートメントであることがわかります。これは予想される動作ですが、実行する特定の書き込み SQL をより詳細に調査して、実行する適切なアクションを決定する必要があります。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。Amazon RDS Performance Insights を使用するか、PostgreSQL システムビュー `pg_stat_activity` をクエリして、`IPC:ProcArrayGroupUpdate` イベントを特定します。

トピック

- トランザクションコミットおよびロールバックオペレーションのモニタリング
- 同時実行数の削減
- 接続プーリングを実装する
- 高速ストレージの使用

トランザクションコミットおよびロールバックオペレーションのモニタリング

コミットとロールバックのモニタリング – コミットとロールバックの数が増えると、ProcArray への負荷が増大する可能性があります。例えば、重複するキー違反の増加が原因で SQL ステートメントが失敗し始めた場合、ロールバックが増加し、ProcArray の競合とテーブルの肥大化が増加する可能性があります。

Amazon RDS Database Insights は PostgreSQL メトリクス `xact_commit` と `xact_rollback` を提供し、1 秒あたりのコミット数とロールバック数を報告します。

同時実行数の削減

トランザクションのバッチ処理 – 可能であれば、コミット/ロールバック操作を減らすために、単一のトランザクションでバッチ操作を行います。

同時実行数の制限 – ProcArray でのロック競合を軽減するために、同時にアクティブなトランザクションの数を減らします。ある程度のテストが必要になりますが、同時接続の合計数を減らすと、競合を減らし、スループットを維持できます。

接続プーリングを実装する

接続プーリングソリューション – 接続プーリングを使用してデータベース接続を効率的に管理し、バックエンドの合計数を減らし、ProcArray のワークロードを削減します。ある程度のテストが必要になりますが、同時接続の合計数を減らすと、競合を減らし、スループットを維持できます。

接続ストームの軽減 – 同様に、頻繁に接続を作成および終了するパターンにより、ProcArray にさらなる負荷がかかります。このパターンを減らすことで、全体的な競合が軽減されます。

高速ストレージの使用

専用ログボリューム – IPC:ProcArrayGroupUpdate 待機イベントに高い IO:WALWrite 待機イベントが伴う場合、専用ログボリュームを設定すると、WAL への書き込みのボトルネックを減らすことができます。これにより、コミットのパフォーマンスが向上します。

詳細については、「専用ログボリューム」を参照してください。

Lock:advisory

Lock:advisoryイベントは、PostgreSQL アプリケーションがロックを使用して複数のセッション全体のアクティビティを調整するときに発生します。

トピック

- 関連するエンジンのバージョン
- Context
- 原因
- アクション

関連するエンジンのバージョン

この待機イベント情報は、RDS for PostgreSQL バージョン 9.6 以降に関連します。

Context

PostgreSQL アドバイザリロックは、ユーザーのアプリケーションコードによって明示的にロックおよびロック解除を実行するアプリケーションレベルの協調的ロックです。アプリケーションは PostgreSQL アドバイザリロックを使用して、複数のセッションにまたがるアクティビティを調整できます。通常のオブジェクトレベルまたは行レベルのロックとは異なり、アプリケーションはロックのライフタイムを完全に制御できます。詳細については、PostgreSQL ドキュメントの Advisory Locks (アドバイザリロック) を参照してください。

アドバイザリロックは、トランザクションが終了する前に解放されるか、トランザクション間のセッションで保持されます。これは、CREATE INDEXステートメントによって取得されたテーブルへのアクセス排他ロックなど、暗黙のうちにシステムで強制されるロックには当てはまりません。

アドバイザリロックの取得 (ロック) およびリリース (ロック解除) に使用される関数の説明については、「PostgreSQL のドキュメント」のアドバイザリロックの関数を参照してください。

アドバイザリロックは、通常の PostgreSQL ロックシステムの上に実装され、pg_locksシステムビューで表示できます。

原因

このロックタイプは、明示的に使用するアプリケーションによって排他的に制御されます。クエリの一部として各行に対して取得されるアドバイザリロックは、ロックの急増や、長期的な蓄積を引き起こすことがあります。

これらの効果は、クエリが返すよりも多くの行でロックを取得する方法でクエリが実行されると発生します。アプリケーションは最終的にすべてのロックを解放する必要がありますが、返されない行でロックが取得された場合、アプリケーションはすべてのロックを見つけることができません。

PostgreSQL のドキュメントの「アドバイザリロック」からの例を紹介します。

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

この例では、LIMIT節がクエリの出力を停止できるのは、内部で行が選択され、その ID 値がロックされた後のみです。これは、データ量の増加により、プランナーが開発中にテストされなかった別の実行プランを選択した場合に突然発生することがあります。この場合の構築アップは、アプリケーションがロックされた各ID値に明示的にpg_advisory_unlockを呼び出すことによって発生します。ただし、この場合、返されなかった行において取得されたロックのセットを見つけることはできません。ロックはセッションレベルで取得されるため、トランザクションの終了時に自動的に解放されません。

ブロックされたロック試行のスパイクは、意図しない競合が原因の可能性があります。このような競合では、アプリケーションの無関係な部分が、誤って同じロック ID スペースを共有します。

アクション

アドバイザリロックのアプリケーション使用状況を確認し、アプリケーションフロー内のいつどこで各タイプのアドバイザリロックが取得および解放されるのか、詳しく説明します。

セッションが取得したロックが多すぎるか、長時間実行しているセッションがロックを早期に解放しないために、ロックの蓄積が遅くなっているかどうかを調べます。pg_terminate_backend(pid)を使用してセッションを終了すると、セッションレベルロックの遅い蓄積を修正できます。

アドバイザリロックを待機中のクライアントがpg_stat_activity、wait_event_type=Lock、wait_event=advisoryに表示されます。同

じpidのpg_locksシステムビューへのクエリを実行し、locktype=advisoryとgranted=fを検索することで、特定のロック値を取得できます。

pg_locksに対してgranted=tを持つ同じアドバイザリロックへのクエリを実行することで、ブロックしているセッションを特定することができます。

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
 blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
 blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

すべてのアドバイザリロック API 関数には、1 つのbigint引数または2つのinteger引数の 2 組の引数があります。

- bigint の引数が 1 つの API 関数では、上位 32 ビットが pg_locks.classid、下位 32 ビットが pg_locks.objid となります。
- integerが2つある API 関数の場合、第 1 引数はpg_locks.classid、第 2 引数はpg_locks.objidとなります。

pg_locks.objsubid値はどの API フォームが使用されたかを示し、1は 1 つのbigint引数、2は 2 つのinteger引数を意味します。

Lock:extend

Lock:extendイベントは、バックエンドプロセスがリレーションを拡張するためにロックするのを待っているときに、別のプロセスが同じ目的でそのリレーションをロックしていると発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

イベントLock:extendは、バックエンドプロセスがリレーションの拡張する間、他のバックエンドプロセスがそのリレーションを拡張するのを待っている間にロックを保持することを示しています。リレーションを拡張できるのは一度に 1 つのプロセスだけなので、システムはLock:extend待機イベントを発生させます。INSERT、COPY、UPDATEのオペレーションでこのイベントを生成することができます。

待機時間が増加する原因の可能性

Lock:extendイベントが通常より頻繁に発生する場合は、パフォーマンスの問題を示していることがあります。典型的な原因は次のとおりです。

同じテーブルへの同時挿入または更新の急増

同じテーブルに挿入または更新するクエリの同時セッションが増加する可能性があります。

ネットワーク帯域幅の不足

DB インスタンスのネットワーク帯域幅が、現在のワークロードのストレージ通信ニーズに対して不十分な可能性があります。これは、Lock:extendイベントの増加を引き起こすストレージレイテンシーの原因となることがあります。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- 同じリレーションへの同時挿入と更新を減らす
- ネットワーク帯域幅の増加

同じリレーションへの同時挿入と更新を減らす

まず、tup_inserted、tup_updatedメトリクスの増加と、それに伴うこの待機イベントの増加があるかどうか判断します。その場合は、挿入および更新オペレーションで競合性が高いリレーションをチェックします。これを判断するには、n_tup_insおよびn_tup_updフィールドでの値のpg_stat_all_tablesビューへのクエリを実行します。pg_stat_all_tablesビューの詳細については、PostgreSQL ドキュメントの「pg_stat_statements」を参照してください。

ブロックおよびブロックされたクエリの詳細については、pg_stat_activity次の例のとおりにクエリを実行します。

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
```

```
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+---------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Lock:extendイベントの増加に寄与するリレーションを特定したら、次の方法を使用して競合を減らします。

- パーティション化によって同じテーブルの競合を減らせるかどうかを調べます。挿入または更新されたタプルを異なるパーティショニングすると、競合を減らすことができます。パーティショニングについては、「pg_partman エクステンションによる PostgreSQL パーティションの管理」を参照してください。
- 待機イベントが主に更新アクティビティによるものである場合は、リレーションのフィルファクタ値を減らすことを検討してください。これにより、更新時に新しいブロックのリクエストを減らすことができます。フィルファクタとは、テーブルページをパッキングするための最大容量を決定する、テーブルの格納パラメータです。これは、ページの総容量に対するパーセンテージで表されます。フィルファクタパラメータの詳細については、PostgreSQL ドキュメントの「CREATE TABLE」を参照してください。

> **Important**
>
> この値を変更すると、ワークロードによってはパフォーマンスに悪影響を及ぼす可能性があるため、フィルファクタを変更する場合は、システムのテストを強くお勧めします。

ネットワーク帯域幅の増加

書き込みレイテンシーが増加しているかどうかを確認するには、WriteLatencyCloudWatch でメトリクスをチェックします。増加している場合は、Amazon CloudWatch の WriteThroughput、ReadThroughput メトリクスを使用し、DB インスタンスのストレージに関するトラフィックをモニタリングしてください。これらのメトリックは、ネットワーク帯域幅がワークロードのストレージアクティビティに十分かどうかを判断するのに役立ちます。

ネットワーク帯域幅が不足している場合は、増加してください。DB インスタンスがネットワーク帯域幅の制限に達している場合、帯域幅を増やす唯一の方法は DB インスタンスのサイズを大きくすることです。

CloudWatch のメトリクスの詳細については、「Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス」を参照してください。DB インスタンスクラスごとの DB エンジンサポートについては、「Amazon RDS の Amazon CloudWatch インスタンスレベルのメトリクス」を参照してください。

Lock:Relation

`Lock:Relation`イベントは、別のトランザクションによって現在ロックされているテーブルまたはビュー (リレーション) のロックを取得するためにクエリが待っているときに発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

ほとんどの PostgreSQL コマンドは、テーブル内のデータへの同時アクセスを制御するために、暗黙のうちにロックを使用します。また、これらのロックは、アプリケーションコード内で`LOCK`コマンドによって明示的に使用することもできます。多くのロックモードは互いに互換性がないため、同じオブジェクトにアクセスしようとしているときにトランザクションをブロックすることがあります。このイベントが発生すると、RDS for PostgreSQL は `Lock:Relation` イベントを生成します。一般的な例をいくつか以下に示します。

- `ACCESS EXCLUSIVE`のような排他的なロックは、すべての同時アクセスをブロックできます。`DROP TABLE`、`TRUNCATE`、`VACUUM FULL`、`CLUSTER`などのデータ定義言語 (DDL) オペレーションは、暗黙のうちに`ACCESS EXCLUSIVE`ロックを取得します。`ACCESS EXCLUSIVE` は、明示的にモードを指定しない `LOCK TABLE` ステートメントのデフォルトのロックモードでもあります。

- テーブル上でCREATE INDEX (without CONCURRENT)を使用すると、ROW EXCLUSIVEロックを取得するデータ操作言語 (DML) ステートメントUPDATE、DELETE、INSERTと競合します。

テーブルレベルのロックと競合するロックモードの詳細については、PostgreSQL ドキュメントの「明示的なロック」を参照してください。

ブロックされたクエリとトランザクションは、通常、次のいずれかの方法でブロックを解除します。

- クエリのブロック: アプリケーションがクエリをキャンセルするか、ユーザーがプロセスを終了できます。また、セッションのステートメントタイムアウトやデッドロック検出メカニズムによって、エンジンがクエリを強制終了させることもできます。
- トランザクションのブロック: トランザクションが ROLLBACK または COMMIT を実行すると、トランザクションはブロックを停止します。ロールバックは、クライアントまたはネットワークの問題によってセッションが切断されたり、終了したときにも自動的に行われます。セッションは、データベースエンジンがシャットダウンされたり、システムがメモリ不足になったりしたときに終了できます。

待機時間が増加する原因の可能性

Lock:Relation イベントが通常よりも頻繁に発生する場合、パフォーマンスの問題を示している可能性があります。代表的な原因としては、以下が挙げられます。

テーブルロックの競合による同時セッションの増加

競合するロックモードで同じテーブルをロックするクエリによる同時セッションの数が増加する可能性があります。

メンテナンスオペレーション

VACUUMやANALYZEのようなヘルスメンテナンスオペレーションは、競合するロックの数を大幅に増加させる可能性があります。VACUUM FULLはACCESS EXCLUSIVEのロックを、ANALYSEはSHARE UPDATE EXCLUSIVEのロックを取得します。どちらのタイプのロックも、Lock:Relation待機イベントを引き起こすことがあります。また、マテリアライズドビューのリフレッシュなどのアプリケーションデータのメンテナンスオペレーションも、ブロックされたクエリとトランザクションを増加することもあります。

リーダーインスタンスをロックする

ライターとリーダーが保持しているリレーションロックの間に矛盾がある可能性があります。現在は、ACCESS EXCLUSIVE リレーションロックのみが、リーダーインスタンスにレプリケート

されます。ただし、ACCESS EXCLUSIVE リレーションロックは、リーダーが保持する ACCESS SHARE リレーションロックと競合します。これにより、リーダーのロックリレーション待機イベントが増加する可能性があります。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- SQL ステートメントのブロックによる影響を軽減
- メンテナンスオペレーションの影響を最小限に抑える

SQL ステートメントのブロックによる影響を軽減

SQL ステートメントのブロックによる影響を軽減するには、可能なところではアプリケーションコードを修正します。ブロックを減らすための 2 つの一般的な方法は以下のとおりです。

- NOWAIT オプションを使用する: SELECT や LOCK ステートメントなど、一部の SQL コマンドはこのオプションをサポートしています。NOWAIT指示文は、ロックをすぐに取得できない場合、ロックへのクエリをキャンセルします。この方法は、ブロックされたセッションが、その後ろにあるブロックされたセッションが積み重なるのを防ぐのに役立ちます。

  例えば、トランザクション A がトランザクション B に保持されているロックを待っているとします。ここで、B がトランザクション C によってロックされているテーブルのロックをリクエストすると、トランザクション C が完了するまでトランザクション A がブロックされる可能性があります。ただし、トランザクション B が C のロックを要求するときにNOWAITを使用する場合、トランザクションBは迅速に失敗し、トランザクション A が無期限に待機する必要がないことを保証できます。
- SET lock_timeout を使用する: lock_timeout 値を設定して、SQL ステートメントがリレーションでロックを取得するのを待機する時間を制限します。指定されたタイムアウト時間内にロックが取得されなかった場合、ロックを要求したトランザクションはキャンセルされます。この値はセッションレベルで設定します。

メンテナンスオペレーションの影響を最小限に抑える

VACUUMやANALYZEのようなメンテナンスオペレーションは重要です。これらのメンテナンス作業に関連するLock:Relation待機イベントを見つけても、それらをオフにしないことをお勧めします。次のようなアプローチにより、これらの操作の影響を最小限に抑えることができます。

- オフピーク時にメンテナンス操作をマニュアルで実行します。
- オートバキュームタスクによるLock:Relation待機をへらすには、必要なオートバキュームチューニングを実行します。オートバキュームのチューニングについては、Amazon RDS ユーザーガイドの「Amazon RDS での PostgreSQL オートバキュームの使用」を参照してください。

Lock:transactionid

Lock:transactionidイベントは、トランザクションが行レベルのロックを待っているときに発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

イベントLock:transactionidは、トランザクションが、同時に実行されているトランザクションにすでに付与された行レベルのロックを取得しようとすると発生します。を示すセッションは、Lock:transactionid待機イベントがこのロックのためにブロックされていることを示します。COMMITまたはROLLBACKステートメントでブロックされているトランザクションの完了後、ブロックされたトランザクションを続行できます。

RDS for PostgreSQL のマルチバージョン同時実行制御セマンティクスは、リーダーがライターを、ライターがリーダーをブロックしないことを保証します。行レベルの競合が発生するには、ブロック

およびブロックされたトランザクションで、次のタイプの競合するステートメントを発行する必要があります。

- UPDATE
- SELECT … FOR UPDATE
- SELECT … FOR KEY SHARE

ステートメントSELECT … FOR KEY SHAREは特殊なケースです。データベースは、レファレンスの整合性のパフォーマンスを最適化するために、FOR KEY SHARE節を使用します。行に対する行レベルロックは、行を参照している他のテーブルのINSERT、UPDATE、DELETEコマンドをブロックできます。

待機時間が増加する原因の可能性

このイベントが通常よりも頻繁に発生する場合、通常はUPDATE、SELECT … FOR UPDATE、またはSELECT … FOR KEY SHAREステートメントが以下の条件と組み合わさることが原因です。

トピック

- 同時実行数が多い
- トランザクションでのアイドル状態
- トランザクションの実行時間が長い

同時実行数が多い

RDS for PostgreSQL は、きめ細かい行レベルのロックセマンティクスを使用できます。以下の条件が満たされると、行レベルの同時実行が発生する可能性が高くなります。

- 同時性の高いワークロードは、同じ行で同時実行します。
- 同時実行数が増加します。

トランザクションでのアイドル状態

時々、pg_stat_activity.state列にはidle in transaction値が表示されます。この値は、トランザクションをスタートしていても、まだCOMMITまたはROLLBACKを発行していないセッションに表示されます。pg_stat_activity.state値がactiveではない場合、pg_stat_activityに表示されるクエリは、実行を終了した最新のクエリになります。ブロッ

クされているセッションは、開いているトランザクションがロックを保持しているため、クエリを積極的に処理しません。

アイドル状態のトランザクションが行レベルロックを取得した場合は、他のセッションがそのロックを取得するのを妨害する可能性があります。この状態は、待機イベントLock:transactionidの頻発につながります。問題を診断するには、pg_stat_activityそしてpg_locksからの出力を検証します。

トランザクションの実行時間が長い

実行時間が長いトランザクションは、長時間ロックされます。これらの長時間のロックは、他のトランザクションの実行をブロックすることがあります。

アクション

行ロックはUPDATE、SELECT … FOR UPDATE、またはSELECT … FOR KEY SHAREステートメント間の競合です。解決策を試す前に、これらのステートメントが同じ行で実行されているかどうかを調べます。この情報をもとに、次のセクションで説明する戦略を選択してください。

トピック

- 同時実行数の多さに対応
- アイドル状態のトランザクションに対応する
- 長時間実行されるトランザクションへの対応

同時実行数の多さに対応

同時実行数が問題になる場合は、以下から 1 つの方法を試行します。

- アプリケーションの同時実行数を減らします。例えば、アクティブなセッションの数を減らします。
- 接続プールを実装します。RDS プロキシを使用して接続をプールする方法については、「Amazon RDS Proxy」を参照してください。
- アプリケーションまたはデータモデルを設計し、UPDATEおよびSELECT … FOR UPDATEステートメントの競合を避けてください。また、SELECT … FOR KEY SHAREステートメントにアクセスされる外部キーの数を減らすこともできます。

アイドル状態のトランザクションに対応する

pg_stat_activity.stateがidle in transactionを示している場合は、以下の方法を使用します。

- 可能な限り、オートコミットをオンにします。この方法では、COMMITまたはROLLBACKを待っている間に、トランザクションが他のトランザクションをブロックすることを防ぎます。
- COMMIT、ROLLBACK、またはENDが不足しているコードパスを検索します。
- アプリケーションの例外処理ロジックに、常に有効なend of transactionへのパスが設定されていることを確認してください。
- COMMITまたはROLLBACKでトランザクションを完了した後、アプリケーションがクエリの結果を処理することを確認します。

長時間実行されるトランザクションへの対応

長時間実行されるトランザクションがLock:transactionidの発生を頻繁に引き起こす場合、以下の方法を試します。

- 長時間実行されるトランザクションが行をロックしないようにします。
- 可能であれば、オートコミットを実装してクエリの長さを制限します。

Lock:tuple

Lock:tuple イベントは、バックエンドプロセスがタプルのロックを取得するのを待っているときに発生します。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

イベントLock:tupleは、バックエンドがタプルのロック取得を待っている間、別のバックエンドが同じタプルで競合するロックを保持していることを示します。次の表では、セッションがLock:tupleイベントを生成するシナリオを示します。

時間	セッション 1	セッション 2	セッション 3
t1	トランザクションをスタートします。		
t2	行 1 を更新します。		
t3		行 1 を更新します。セッションは、タプルの排他ロックを取得してから、セッション1がコミットまたはロールバックによってロックを解放するのを待機します。	
t4			行 1 を更新します。セッションは、セッション 2 がタプルの排他ロックを解放するのを待機します。

または、ベンチマークツールpgbenchを使用してこの待機イベントをシミュレートすることができます。多数の同時セッションを構成して、カスタムSQLファイルでテーブル内の同じ行を更新します。

競合するロックモードの詳細については、「PostgreSQL のドキュメント」の「明示的なロック」を参照してください。pgbenchの詳細については、「PostgreSQL のドキュメント」の「pgbench」を参照してください。

待機時間が増加する原因の可能性

このイベントが通常よりも頻繁に表示される場合、パフォーマンスの問題を示している可能性があり、典型的な原因は次のとおりです。

- UPDATEまたはDELETEステートメントの実行により、同じタプルへの競合するロックを取得しようとしている同時セッションが多数あります。
- 同時実行数の多いセッションでは、FOR UPDATEまたはFOR NO KEY UPDATEロックモードを使用してSELECTステートメントを実行します。
- さまざまな要因により、アプリケーションまたは接続プールが同じオペレーションを実行するため、より多くのセッションを開きます。新しいセッションが同じ行を変更しようとすると、DBロードのスパイクが発生してLock:tupleが表示されることがあります。

詳細については、「PostgreSQL のドキュメント」の「行レベルのロック」を参照してください。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- アプリケーションロジックを調査する
- ブロッカーセッションを見つける
- 同時実行数が多いときにそれを減らす
- ボトルネックのトラブルシューティング

アプリケーションロジックを調査する

ブロッカーセッションが長い間idle in transactionステートメントにあったかどうかを確認します。その場合は、短期的な解決策としてブロッカーセッションの終了を検討してください。pg_terminate_backend 関数を使用することもできます。この関数の詳細については、「PostgreSQL のドキュメント」の「サーバーシグナリング関数」を参照してください。

長期的な解決策としては、以下を実行してください。

- アプリケーションロジックを調整します。
- idle_in_transaction_session_timeout パラメータを使用します。このパラメータは、指定された時間より長くアイドル状態であったオープントランザクションを持つセッションを終了さ

せます。詳細については、PostgreSQL ドキュメントの「Client Connection Defaults (クライアント接続のデフォルト)」を参照してください。

- 可能な限りオートコミットを使用します。詳細については、PostgreSQL ドキュメントの「Client authentication (クライアント認証)」を参照してください。

ブロッカーセッションを見つける

Lock:tupleの待機イベントが発生している間に、どのロックが互いに依存しているかを探して、ブロッカーとブロックされたセッションを特定します。詳細については、PostgreSQL wiki の「依存関係情報のロック」を参照してください。

次の例では、tupleでフィルタリングしてwait_timeで順序付けし、すべてのセッションへのクエリを実行しています。

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
 blocked_locks.transactionid
```

```
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

同時実行数が多いときにそれを減らす

Lock:tupleイベントは、特にワークロードの多い時間帯に、コンスタントに発生する可能性があります。このような状況では、非常にビジーな行への同時実行数を減らすことを検討してください。多くの場合、キューまたはブールロジックを制御する行の数行だけで、これらの行は非常にビジーになります。

ビジネス要件、アプリケーションロジック、およびワークロードタイプに応じて、さまざまなアプローチを使用することで、競合を減らすことができます。例えば、次のオペレーションを実行できます。

- テーブルとデータロジックを再設計し、競合を減らします。
- アプリケーションロジックを変更して、行レベルで高い競合を減らします。
- 行レベルのロックを活用してクエリを再設計します。
- NOWAIT節はリトライオペレーションで使用します。
- 楽観的かつハイブリッドロックロジックの同時実行制御の使用を検討します。
- データベースの隔離レベルの変更を検討してください。

ボトルネックのトラブルシューティング

Lock:tupleは、CPU の枯渇や Amazon EBS 帯域幅の最大使用率などのボトルネックを発生させることがあります。ボトルネックを減らすには、次のアプローチを検討します。

- インスタンスクラスタイプをスケールアップします。
- リソースを大量に消費するクエリを最適化します。
- アプリケーションロジックを変更します。
- ほとんどアクセスされないデータをアーカイブします。

LWLock:BufferMapping (LWLock:buffer_mapping)

このイベントは、セッションがデータブロックを共有バッファプール内のバッファに関連付けるのを待っているときに発生します。

> **Note**
>
> RDS for PostgreSQL のバージョン 13 以降では、このイベントの名前は `LWLock:BufferMapping` になります。RDS for PostgreSQL のバージョン 12 以前では、このイベントの名前は `LWLock:buffer_mapping` になります。

トピック

- サポート対象エンジンバージョン
- Context
- 原因
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL バージョン 9.6 以降に関連します。

Context

共有バッファプールは、プロセスで使用されている、または使用されていたすべてのページを保持する PostgreSQL のメモリ領域です。プロセスがページを必要とするとき、そのページを共有バッファプールに読み取ります。shared_buffersパラメータは、共有バッファサイズを設定し、テーブルとインデックスページを格納するためのメモリ領域を予約します。このパラメータを変更する場合は、必ずデータベースを再起動してください。

`LWLock:buffer_mapping`待機イベントは、以下の場合に発生します。

- プロセスは、バッファテーブルでページを検索し、共有バッファマッピングロックを取得します。
- プロセスは、ページをバッファプールにロードし、排他的バッファマッピングロックを取得します。
- プロセスは、プールからページを削除し、排他バッファマッピングロックを取得します。

原因

このイベントが通常よりも頻繁に発生する場合、パフォーマンスの問題を示していることがあり、データベースは共有バッファプールにページインとアウトページングを行っています。代表的な原因としては、以下が挙げられます。

- 大きなクエリ
- 肥大化したインデックスとテーブル
- フルテーブルスキャン
- ワーキングセットより小さい共有プールサイズ

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。

トピック

- バッファ関連のメトリクスをモニタリングする
- インデックス作成戦略を評価する
- 迅速に確保しなければならないバッファの数を減らす

バッファ関連のメトリクスをモニタリングする

`LWLock:buffer_mapping`がスパイクを待機したら、バッファヒット率を調べます。これらのメトリクスを使用すると、バッファキャッシュで何が起こっているかをより深く理解できます。次のメトリックを検証します。

`blks_hit`

この Performance Insights カウンターメトリクスは、共有バッファプールから取得されたブロックの数を示します。`LWLock:buffer_mapping`の待機イベントが表示された後、`blks_hit`のスパイクを観察することがあります。

`blks_read`

この Performance Insights カウンターメトリクスは、共有バッファプールへの読み取りのため、 I/O を必要とするブロックの数を示します。`LWLock:buffer_mapping`待機イベントまでに`blks_read`のスパイクが観察されることがあります。

インデックス作成戦略を評価する

インデックス作成戦略がパフォーマンスが低下させていないことを確認するには、次を確認してください。

インデックスの肥大化

インデックスとテーブルの肥大化によって、不要なページが共有バッファに読み込まれないようにします。テーブルに未使用の行がある場合は、データをアーカイブし、テーブルから行を削除することを検討してください。その後、サイズ変更されたテーブルのインデックスを再構築できます。

頻繁に使用するクエリのインデックス

最適なインデックスがあるかどうかを判断するには、Performance Insights で DB エンジンのメトリクスをモニタリングします。`tup_returned`メトリクスは、読み込まれた行数を示します。`tup_fetched`メトリクスは、クライアントに返される行数を示します。`tup_returned`が`tup_fetched`を大幅に超える場合、データが適切にインデックスされていない可能性があります。また、テーブルの統計が最新ではない可能性があります。

迅速に確保しなければならないバッファの数を減らす

`LWLock:buffer_mapping`待機イベントを減らすには、迅速に割り当てる必要があるバッファの数を減らしてください。1 つの戦略として、より小規模なバッチオペレーションを実行します。テーブルをパーティション化することで、より小さなバッチを実現できることがあります。

LWLock:BufferIO (IPC:BufferIO)

`LWLock:BufferIO` イベントは、RDS for PostgreSQL が同時にページにアクセスしようとしているときに、他のプロセスが入出力 (I/O) オペレーションの完了を待っているときに発生します。その目的は、同じページを共有バッファに読み込むことです。

トピック

- 関連するエンジンのバージョン
- Context
- 原因
- アクション

関連するエンジンのバージョン

この待機イベント情報は、すべての RDS for PostgreSQL のバージョンに関連しています。RDS for PostgreSQL 12 以前のバージョンでは、この待機イベントは lwlock:buffer_io という名前でしたが、RDS for PostgreSQL 13 バージョンでは lwlock:bufferio という名前です。RDS for PostgreSQL 14 バージョンから、BufferIO 待機イベントは `LWLock` から `IPC` 待機イベントタイプ (IPC:BufferIO) に移動されました。

Context

各共有バッファは、ブロック (またはページ) が共有バッファプールの外部で取得される必要があるたびに、`LWLock:BufferIO`待機イベントに関連付けられた I/O ロックを持ちます。

このロックは、すべての同じブロックへのアクセスを必要とする複数のセッションを処理するために使用されます。このブロックは、`shared_buffers`パラメータで定義された共有バッファプールの外部から読み取る必要があります。

共有バッファプール内でページが読み込まれると、`LWLock:BufferIO`ロックが解除されます。

> **Note**
>
> `LWLock:BufferIO`待機イベントはIO:DataFileRead待機イベントに先行します。`IO:DataFileRead`待機イベントは、データがストレージから読み込まれている間に発生します。

ライトウェイトロックの詳細については、「ロックの概要」を参照してください。

原因

`LWLock:BufferIO`上位待機中に表示されるイベントの一般的な原因には、次のものがあります。

- 複数のバックエンドまたは接続が I/O オペレーションを保留している同じページにアクセスしようとしている
- 共有バッファプール (`shared_buffers`パラメータで定義) のサイズと、現在のワークロードが必要とするバッファ数の比率
- 共有バッファプールのサイズが、他の操作によって削除されるページ数とのバランスが悪い
- エンジンが共有バッファプールに必要以上のページを読み込む必要がある大規模なインデックスまたは肥大化したインデックス

- DB エンジンが強制的に必要以上に多くのページをテーブルから読み取るインデックスの欠落
- チェックポイント発生が頻繁すぎたり、変更されたページをフラッシュする必要が多すぎる
- 同じページで操作を実行しようとするデータベース接続が突然スパイクする

アクション

待機イベントの原因に応じたさまざまなアクションを実行することをお勧めします。

- `BufferCacheHitRatio`の急減と`LWLock:BufferIO`待機イベントの相関関係のため、Amazon CloudWatch メトリクスを観察します。この効果は、共有バッファの設定が小さいことを意味することがあります。増やすか、DB インスタンスクラスをスケールアップする必要がある場合があります。ワークロードをより多くのリーダーノードに分割できます。
- `LWLock:BufferIO`が`BufferCacheHitRatio`のメトリックと一致する場合は、ワークロードのピーク時間に基づいて`max_wal_size`と`checkpoint_timeout`をチューニングしてください。次に、原因となっているクエリを特定します。
- 未使用のインデックスがあるかどうかを確認し、それらを削除します。
- パーティション化されたテーブルを使用します (パーティション化されたインデックスもあります)。これにより、インデックスの並べ替えを低く抑え、その影響を軽減することができます。
- 不必要に列のインデックスを作成しないようにします。
- 接続プールを使用して、突然のデータベース接続スパイクを防ぎます。
- ベストプラクティスとして、データベースへの最大接続数を制限します。

LWLock:buffer_content (BufferContent)

この`LWLock:buffer_content`待機イベントは、セッションがデータページをメモリに読み取るまたは書き込むために待機中、他のセッションがそのページを書き込み用にロックしている場合に発生します。RDS for PostgreSQL 13 以降では、この待機イベントは `BufferContent` と呼ばれます。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

データの読み取りや操作のために、PostgreSQL は共有メモリバッファを介してデータにアクセスします。バッファから読み取るために、プロセスは共有モードでバッファコンテンツに対する軽量ロック (LwLock) を取得します。バッファに書き込むには、排他モードでそのロックを取得します。共有ロックを使用すると、他のプロセスがそのコンテンツの共有ロックを同時に取得できます。排他ロックは、他のプロセスによるいかなるタイプのロック取得も防ぎます。

LWLock:buffer_content(BufferContent) イベントは、複数のプロセスが特定のバッファの内容をロックしようとしていることを示します。

待機時間が増加する原因の可能性

LWLock:buffer_content (BufferContent) イベントが通常より頻繁に発生し、パフォーマンスの問題を示している可能性がある場合、代表的な原因として以下が挙げられます。

同一データに対する同時更新の増加

同じバッファコンテンツを更新するクエリによる同時実行セッションの数が増加する可能性があります。この競合は、インデックスの多いテーブルではより顕著になることがあります。

ワークロードデータがメモリ内に存在しない

アクティブなワークロードが処理しているデータがメモリ上にない場合、これらの待機イベントが増加する可能性があります。この効果は、ロックを保持しているプロセスが、ディスク I/O 操作の実行中にロックを長く維持できるためです。

外部キー制約の過度の使用

外部キー制約により、プロセスがバッファコンテンツロックを保持する時間を増やすことがあります。この効果は、読み取り操作では、そのキーが更新されている間、参照キーに対する共有バッファコンテンツのロックが必要になるためです。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。LWLock:buffer_content(BufferContent)イベントは、Amazon RDS Performance Insights を使用するか、ビューpg_stat_activityのクエリで特定することができます。

トピック

- インメモリ効率の向上
- 外部キー制約の使用を減らす
- 未使用インデックスの削除
- シーケンスを使用する場合は、キャッシュサイズを増やしてください

インメモリ効率の向上

アクティブなワークロードデータがメモリ内に存在する可能性を高めるには、テーブルをパーティション化するか、インスタンスクラスをスケールアップします。DB インスタンスクラスの詳細については、「 DB インスタンスクラス」を参照してください。

外部キー制約の使用を減らす

外部キー制約の使用で、LWLock:buffer_content(BufferContent)待機イベントが多発しているワークロードを調査します。不要な外部キーの制約を削除します。

未使用インデックスの削除

LWLock:buffer_content (BufferContent) 待機イベントが多いワークロードで、未使用のインデックスを特定して削除します。

シーケンスを使用する場合は、キャッシュサイズを増やしてください

テーブルがシーケンスを使用している場合は、キャッシュサイズを増やして、シーケンスページとインデックスページの競合をなくします。各シーケンスは共有メモリ内の 1 ページです。定義済みのキャッシュは接続ごとです。多くの同時セッションによってシーケンス値を取得している場合、これではワークロードを処理するのに不十分な場合があります。

LWLock:lock_manager (LWLock:lockmanager)

このイベントは、RDS for PostgreSQL エンジンが、高速パスロックが不可能な場合に共有ロックのメモリ領域を維持し、ロックの割り当て、チェック、および解放を行うときに発生します。

トピック

- サポート対象エンジンバージョン
- Context

- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL バージョン 9.6 以降に関連します。RDS for PostgreSQL のバージョン 13 より前のリリースでは、この待機イベントの名前は `LWLock:lock_manager` になります。RDS for PostgreSQL のバージョン 13 以降のリリースでは、この待機イベントの名前は `LWLock:lockmanager` になります。

Context

SQL ステートメントを発行すると、RDS for PostgreSQL は同時オペレーション中にデータベースの構造、データ、および整合性を保護するためにロックを記録します。エンジンは、高速パスロックまたは高速ではないパスロックを使用して、この目標を達成できます。高速ではないパスロックは、高速パスロックよりも高価で、オーバーヘッドも多く発生します。

高速パスロック

バックエンドプロセスでは、頻繁にロックされロック解除されるがめったに競合しないロックのオーバーヘッドを減らすために、高速パスロックを使用できます。データベースでは、以下の基準を満たすロックにこのメカニズムを使用します。

- DEFAULT ロック方式を使用します。
- これらは、共有関係ではなく、データベースリレーションに対するロックを表します。
- これらは競合する可能性が低い弱いロックです。
- エンジンは、競合するロックが存在する可能性がないことをすばやく確認できます。

エンジンは、以下のいずれかの条件が true の場合、高速パスロックを使用できません。

- ロックが上記の条件を満たしていません。
- バックエンドプロセスに使用できるスロットはこれ以上ありません。

高速パスロック用にクエリを調整するには、次のクエリを使用できます。

```
SELECT count(*), pid, mode, fastpath
  FROM pg_locks
```

```
 WHERE fastpath IS NOT NULL
 GROUP BY 4,3,2
 ORDER BY pid, mode;
 count | pid  |      mode       | fastpath
-------+------+-----------------+----------
16 | 9185 | AccessShareLock | t
336 | 9185 | AccessShareLock | f
1 | 9185 | ExclusiveLock   | t
```

次のクエリでは、データベース全体の合計のみを表示します。

```
SELECT count(*), mode, fastpath
  FROM pg_locks
 WHERE fastpath IS NOT NULL
 GROUP BY 3,2
 ORDER BY mode,1;
count |      mode       | fastpath
-------+-----------------+----------
16 | AccessShareLock | t
337 | AccessShareLock | f
1 | ExclusiveLock   | t
(3 rows)
```

高速パスロックの詳細については、「PostgreSQL ロックマネージャ README」 のfast pathおよび「PostgreSQL ドキュメント」のpg-locksを参照してください。

ロックマネージャのスケーリング問題の例

この例では、purchasesという名前のテーブルが 5 年分のデータを日ごとにパーティション化して保存しています。各パーティションには 2 つのインデックスがあります。次の一連のイベントが発生します。

1. 何日分ものデータをクエリすると、データベースは多くのパーティションを読み取る必要があります。
2. データベースは、各パーティションに対してロックエントリを作成します。パーティションインデックスがオプティマイザアクセスパスの一部である場合、データベースはそれらのロックエントリも作成します。
3. 同じバックエンドプロセスでリクエストされたロックエントリの数がFP_LOCK_SLOTS_PER_BACKENDの値である16より大きい場合、ロックマネージャは非高速パスロック方式を使用します。

最新のアプリケーションには何百ものセッションがあることがあります。同時セッションが適切なパーティションプルーニングを行わずに親を照会している場合、データベースは数百または数千の非高速パスロックを作成することがあります。通常、この同時実行が vCPUs の数よりも多いと、LWLock:lock_manager待機イベントが表示されます。

> **Note**
>
> LWLock:lock_manager待機イベントは、データベーススキーマのパーティションまたはインデックスの数とは関係ありません。代わりに、データベースが制御する必要がある非高速パスロックの数と関係しています。

待機時間が増加する原因の可能性

LWLock:lock_manager待機イベントが通常より頻繁に発生する場合は、おそらくパフォーマンスの問題を示しており、突然のスパイクが発生する原因として最も可能性が高いものは次のとおりです。

- 同時アクティブセッションは、高速パスロックを使用しないクエリを実行しています。また、これらのセッション数が最大 vCPU を超えています。
- 多数の同時アクティブセッションが、大きくパーティション化されたテーブルにアクセスしています。各パーティションには複数のインデックスがあります。
- データベースで接続ストームが発生しています。デフォルトでは、一部のアプリケーションと接続プールソフトウェアは、データベースが遅い場合により多くの接続を作成します。これは問題を悪化させます。接続ストームが発生しないように接続プールソフトウェアをチューニングします。
- 多数のセッションが、パーティションをプルーニングせずに親テーブルをクエリします。
- データ定義言語 (DDL)、データ操作言語 (DML)、またはメンテナンスコマンドは、頻繁にアクセスまたは変更されるビジーリレーションあるいはタプルのいずれかを排他的にロックします。

アクション

CPU待機イベントが発生する場合、必ずしもパフォーマンスの問題を示しているとは限りません。パフォーマンスが低下し、この待機イベントが DB ロードを支配している場合にのみ、このイベントに応答します。

トピック

- パーティションプルーニングを使用する

- 不要なインデックスを削除する
- 高速パスロック用にクエリをチューニングする
- 他の待機イベントに合わせてチューニングする
- ハードウェアのボトルネックを減らす
- 接続プーラーを使用する
- RDS for PostgreSQL バージョンのアップグレード

パーティションプルーニングを使用する

パーティションプルーニングとは、宣言によってパーティション分割されたテーブルに対するクエリ最適化戦略のことで、不要なパーティションをテーブルスキャンから除外し、パフォーマンスを向上させます。パーティションプルーニングは、デフォルトで有効になっています。オフになっている場合は、次のようにオンにします。

```
SET enable_partition_pruning = on;
```

クエリのWHERE節にパーティショニングに使用される列が含まれる場合は、パーティションのプルーニングを利用できます。詳細については、PostgreSQL のドキュメントの「パーティションプルーニング」を参照してください。を参照してください。

不要なインデックスを削除する

データベースには、未使用またはほとんど使用されないインデックスが含まれている可能性があります。その場合は、それらの削除を検討します。次のいずれかを実行します。

- 不要なインデックスを見つける方法については、PostgreSQL wikiの「未使用のインデックス」を参照してください。
- PG コレクタを実行します。この SQL スクリプトは、データベース情報を収集し、統合 HTML レポートに表示します。「未使用のインデックス」セクションをチェックします。詳細については、AWSLabs GitHub リポジトリの「PG コレクター」を参照してください。

高速パスロック用にクエリをチューニングする

クエリで高速パスロックが使用されているかどうかを調べるには、pg_locksテーブルのfastpath列にクエリを実行します。クエリで高速パスロックを使用していない場合は、クエリあたりのリレーション数を 16 未満に減らしてください。

他の待機イベントに合わせてチューニングする

LWLock:lock_managerが上位待機のリストの第1位または 2 番目である場合、次の待機イベントもリストに表示されるかどうかを確認します。

- Lock:Relation
- Lock:transactionid
- Lock:tuple

上記のイベントがリスト内で上位に表示される場合は、まずこれらの待機イベントのチューニングを検討してください。これらのイベントは、LWLock:lock_managerのドライバーになり得ます。

ハードウェアのボトルネックを減らす

CPU の枯渇や Amazon EBS 帯域幅の最大使用率など、ハードウェアのボトルネックが発生する可能性があります。このような場合は、ハードウェアのボトルネックを減らすことを検討してください。以下のアクションの場合を検討します。

- インスタンスクラスをスケールアップします。
- 大量の CPU とメモリを消費するクエリを最適化します。
- アプリケーションロジックを変更します。
- データをアーカイブします。

CPU、メモリ、および EBS ネットワーク帯域幅の詳細については、Amazon RDS インスタンスタイプを参照してください。

接続プーラーを使用する

アクティブな接続の総数が最大 vCPU を超えると、インスタンスタイプがサポートできるより多くの OS プロセスが CPU を必要とします。このような場合は、接続プールの使用またはチューニングを検討してください。インスタンスタイプの vCPUs の詳細については、「Amazon RDS インスタンスタイプ」を参照してください。

接続プールの詳細については、次のリソースを参照してください。

- Amazon RDS Proxy
- pgbouncer
- PostgreSQL ドキュメントの接続プールとデータソース

RDS for PostgreSQL バージョンのアップグレード

現在使用している RDS for PostgreSQL のバージョンが 12 より前のものであれば、バージョン 12 以降にアップグレードします。PostgreSQL バージョン 12 以降では、パーティションメカニズムが改良されています。バージョン12の詳細については、PostgreSQL 12.0 リリースノートを参照してください。RDS for PostgreSQL のアップグレードの詳細については、「RDS for PostgreSQL DB エンジンのアップグレード」を参照してください。

LWLock:pg_stat_statements

LWLock:pg_stat_statements 待機イベントは、pg_stat_statements 拡張機能が SQL ステートメントを追跡するハッシュテーブルで排他的ロックを取得したときに発生します。これは以下のようなシナリオで発生します。

- 追跡されるステートメントの数が設定された pg_stat_statements.max パラメータ値に達し、さらに多くのエントリを格納する必要がある場合、拡張機能は呼び出し数をソートし、最も実行されていないステートメントの 5% を削除し、ハッシュに残りのエントリを再入力します。
- pg_stat_statements がディスク上の pgss_query_texts.stat ファイルに garbage collection 操作を実行し、ファイルを書き換えた場合。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

pg_stat_statements 拡張機能を理解する – pg_stat_statements 拡張機能は、ハッシュテーブル内の SQL ステートメントの実行統計を追跡します。拡張機能は、pg_stat_statements.max パラメータで定義された制限まで SQL ステートメントを追跡します。このパラメータは、pg_stat_statements ビューの最大行数に対応する、追跡できるステートメントの最大数を決定します。

ステートメント統計の永続性 – 拡張機能は、以下によってインスタンスのすべての再起動でステートメント統計を保持します。

- pg_stat_statements.stat という名前のファイルにデータを書き込む
- pg_stat_statements.save パラメータを使用して永続性の動作を制御する

pg_stat_statements.save の設定と動作。

- on (デフォルト): 統計はシャットダウン時に保存され、サーバー起動時に再ロードされる
- off: 統計はシャットダウン時に保存されず、サーバー起動時に再ロードされない

クエリテキストストレージ – 拡張機能は、追跡されたクエリのテキストを `pgss_query_texts.stat` という名前のファイルに保存します。このファイルは、ガベージコレクションが発生する前に、追跡されるすべての SQL ステートメントの平均サイズの 2 倍に増加する可能性があります。拡張機能では、クリーンアップオペレーションと `pgss_query_texts.stat` ファイルの書き換え中にハッシュテーブルに排他的ロックが必要です。

ステートメントの割り当て解除プロセス – 追跡されたステートメントの数が `pg_stat_statements.max` の制限に達し、新しいステートメントを追跡する必要がある場合、拡張機能は以下の処理を行います。

- ハッシュテーブルで排他的ロック (LWLock:pg_stat_statements) を取得します。
- 既存のデータをローカルメモリにロードします。
- 呼び出しの数に基づいてクイックソートを実行します。
- 最小呼び出しステートメント (下位 5%) を削除します。
- ハッシュテーブルに残りのエントリを再入力します。

ステートメントの割り当て解除のモニタリング – PostgreSQL 14 以降では、pg_stat_statements_info ビューを使用してステートメントの割り当て解除をモニタリングできます。このビューには、新しいステートメントのスペースを確保するためにステートメントが割り当て解除された回数を示す dealloc 列が含まれています。

ステートメントの割り当て解除が頻繁に発生すると、ディスク上の `pgss_query_texts.stat` ファイルのガベージコレクションの頻度が高くなります。

待機時間が増加する原因の可能性

LWLock:pg_stat_statements の待機時間が長くなる一般的な原因は次のとおりです。

- アプリケーションで使用される一意のクエリの数の増加。
- pg_stat_statements.max パラメータ値が、使用されている一意のクエリの数と比較して小さい。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。LWLock:pg_stat_statements イベントは、Amazon RDS Performance Insights を使用するか、ビュー pg_stat_activity のクエリで特定することができます。

次の pg_stat_statements パラメータを調整して追跡動作を制御し、LWLock:pg_stat_statements の待機イベントを減らします。

トピック

- pg_stat_statements.track パラメータを無効にする
- pg_stat_statements.max パラメータを増やす
- pg_stat_statements.track_utility パラメータを無効にする

pg_stat_statements.track パラメータを無効にする

LWLock:pg_stat_statements 待機イベントがデータベースのパフォーマンスに悪影響を及ぼし、pg_stat_statements ビューをさらに分析して根本原因を特定する前に迅速な対応が必要な場合は、pg_stat_statements.track パラメータを none に設定することで無効にできます。これにより、ステートメント統計の収集が無効になります。

pg_stat_statements.max パラメータを増やす

ディスク上の pgss_query_texts.stat ファイルの割り当て解除を減らし、ガベージコレクションを最小限に抑えるため、pg_stat_statements.max パラメータの値を増やします。デフォルト値は 5,000 です。

> **Note**
>
> pg_stat_statements.max パラメータは即時反映されません。これらのパラメータの変更を適用するには、DB インスタンスを再起動する必要があります。

pg_stat_statements.track_utility パラメータを無効にする

pg_stat_statements ビューを分析して、pg_stat_statements によって追跡されるリソースを最も多く消費しているユーティリティコマンドを特定できます。

pg_stat_statements.track_utility パラメータは、SELECT、INSERT、UPDATE、DELETE、MERGE を除くすべてのコマンドを含むユーティリティコマンドをモジュールが追跡するかどうかを制御します。デフォルトでは、このパラメータは on に設定されます。

例えば、アプリケーションが本質的に一意である多くのセーブポイントクエリを使用する場合、ステートメントの割り当て解除が増加する可能性があります。これに対処するには、pg_stat_statements.track_utility パラメータを無効にして pg_stat_statements がセーブポイントクエリを追跡しないようにします。

> **Note**
>
> pg_stat_statements.track_utility パラメータは変更後、即時反映されます。データベースインスタンスを再起動することなく、その値を変更できます。

Example pg_stat_statements での一意のセーブポイントクエリの例

```
                    query                         |       queryid
--------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                  | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                    | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                   |  54791337410474486
 SAVEPOINT JDBC_SAVEPOINT_1294466                 |  8170064357463507593
 ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016       | -33608214779996400
 SAVEPOINT JDBC_SAVEPOINT_14185                   | -2175035613806809562
 SAVEPOINT JDBC_SAVEPOINT_45837                   | -6201592986750645383
 SAVEPOINT JDBC_SAVEPOINT_1324                    |  6388797791882029332
```

PostgreSQL 17 では、ユーティリティコマンドの追跡にいくつかの機能強化が導入されています。

- セーブポイント名が定数として表示されるようになりました。
- 2 フェーズコミットコマンドのグローバルトランザクション ID (GID) が定数として表示されるようになりました。
- DEALLOCATE ステートメントの名前は定数として表示されます。
- CALL パラメータが定数として表示されるようになりました。

LWLock:SubtransSLRU (LWLock:SubtransControlLock)

`LWLock:SubtransSLRU` および `LWLock:SubtransBuffer` 待機イベントは、セッションがサブトランザクション情報の単純最小使用時間 (SLRU) キャッシュへのアクセスを待っていることを示します。これは、トランザクションの可視性と親子関係を決定するときに発生します。

- `LWLock:SubtransSLRU`: プロセスは、サブトランザクションのシンプルな最も長い時間使われていない (SLRU) キャッシュへのアクセスを待っています。バージョン 13 より前の RDS for PostgreSQL では、この待機イベントは `SubtransControlLock` と呼ばれます。
- `LWLock:SubtransBuffer`: プロセスは、サブトランザクションのシンプルな最も長い時間使われていない (SLRU) バッファの I/O を待っています。バージョン 13 より前の RDS for PostgreSQL では、この待機イベントは `subtrans` と呼ばれます。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

サブトランザクションを理解する – サブトランザクションは、PostgreSQL のトランザクション内のトランザクションです。ネストされたトランザクションとも呼ばれます。

サブトランザクションは通常、次の場合に作成されます。

- SAVEPOINT コマンド
- 例外ブロック (BEGIN/EXCEPTION/END)

サブトランザクションを使用すると、トランザクション全体に影響を与えることなく、トランザクションの一部をロールバックできます。これにより、トランザクション管理をきめ細かく制御できます。

実装の詳細 – PostgreSQL は、メイントランザクション内のネストされた構造としてサブトランザクションを実装します。各サブトランザクションは独自のトランザクション ID を取得します。

実装の主な側面。

- トランザクション ID は pg_xact で追跡されます。
- 親子関係は PGDATA の pg_subtrans サブディレクトリに保存されます。
- 各データベースセッションは、最大 64 のアクティブなサブトランザクションを維持できます
- この制限を超えると、サブトランザクションオーバーフローが発生します。サブトランザクション情報を取得するために、最も長い時間使われていない (SLRU) キャッシュにアクセスする必要があります

待機時間が増加する原因の可能性

サブトランザクション SLRU 競合の一般的な原因は次のとおりです。

- SAVEPOINT および EXCEPTION 処理の過剰な使用 – EXCEPTION ハンドラーを使用した PL/pgSQL プロシージャは、例外が発生するかどうかにかかわらず、暗黙的なセーブポイントを自動的に作成します。各 SAVEPOINT は新しいサブトランザクションを開始します。1 つのトランザクションが 64 を超えるサブトランザクションを蓄積すると、サブトランザクション SLRU オーバーフローがトリガーされます。
- ドライバーと ORM の設定 – SAVEPOINT の使用は、アプリケーションコードで明示的で行うことも、ドライバー設定を通じて暗黙的に行うこともできます。一般的に使用される ORM ツールやアプリケーションフレームワークの多くは、ネストされたトランザクションをネイティブにサポートしています。以下は一般的な例です。
  - JDBC ドライバーパラメータ autosave を always または conservative に設定すると、各クエリの前にセーブポイントが生成されます。
  - Spring Framework トランザクション定義を propagation_nested に設定した場合。

  - `requires_new: true` が設定されている場合の Rails。
  - `session.begin_nested` を使用する場合の SQLAlchemy。
  - ネストされた `atomic()` ブロックを使用する場合の Django。
  - `Savepoint` を使用する場合の GORM。
  - ロールバックレベル設定がステートメントレベルのロールバックに設定されている場合の psqlODBC (例: `PROTOCOL=7.4-2`)。
- 長時間実行されるトランザクションとサブトランザクションを伴う高同時ワークロード – 同時ワークロードが高く、長時間実行されるトランザクションとサブトランザクション中にサブトランザクション SLRU オーバーフローが発生すると、PostgreSQL の競合が増加します。これは、`LWLock:SubtransBuffer` および `LWLock:SubtransSLRU` ロックの昇格された待機イベントとして現れます。

アクション

待機イベントの原因に応じたさまざまなアクションをお勧めします。一部のアクションは即時の打開策を提供しますが、他のアクションは調査と長期的な修正を必要とします。

トピック

- サブトランザクション使用状況のモニタリング
- メモリパラメータを設定する
- 長時間のアクション

サブトランザクション使用状況のモニタリング

PostgreSQL バージョン 16.1 以降では、次のクエリを使用して、バックエンドごとのサブトランザクション数とオーバーフローステータスをモニタリングします。このクエリは、バックエンド統計をアクティビティ情報と結合して、サブトランザクションを使用しているプロセスを示します。

```
SELECT a.pid, usename, query, state, wait_event_type,
       wait_event, subxact_count, subxact_overflowed
FROM (SELECT id, pg_stat_get_backend_pid(id) pid, subxact_count, subxact_overflowed
      FROM pg_stat_get_backend_idset() id
           JOIN LATERAL pg_stat_get_backend_subxact(id) AS s ON true
     ) a
JOIN pg_stat_activity b ON a.pid = b.pid;
```

PostgreSQL バージョン 13.3 以降では、サブトランザクションキャッシュプレッシャーについて pg_stat_slru ビューをモニタリングします。次の SQL クエリは、Subtrans コンポーネントの SLRU キャッシュ統計を取得します。

```
SELECT * FROM pg_stat_slru WHERE name = 'Subtrans';
```

blks_read 値が一貫して増加すると、キャッシュされていないサブトランザクションのディスクアクセスが頻繁になり、SLRU キャッシュプレッシャーの可能性が示されます。

メモリパラメータを設定する

PostgreSQL 17.1 以降では、subtransaction_buffers パラメータを使用してサブトランザクション SLRU キャッシュサイズを設定できます。次の設定例は、サブトランザクションバッファパラメータを設定する方法を示しています。

```
subtransaction_buffers = 128
```

このパラメータは、サブトランザクションコンテンツのキャッシュに使用される共有メモリの量を指定します (pg_subtrans)。単位なしで指定した場合、値は BLCKSZ バイトのブロックを表し、通常はそれぞれ 8KB です。例えば、値を 128 に設定すると、サブトランザクションキャッシュに 1MB (128 * 8kB) のメモリが割り当てられます。

> **Note**
>
> このパラメータをクラスターレベルで設定することで、すべてのインスタンスの整合性を維持できます。特定のワークロード要件とインスタンスクラスに適した値をテストして調整します。パラメータの変更を有効にするには、ライターインスタンスを再起動する必要があります。

長時間のアクション

- アプリケーションコードと設定の確認 – アプリケーションコードとデータベースドライバーの設定で、明示的および暗黙的な SAVEPOINT 使用状況とサブトランザクションの一般的な使用状況の

両方を確認します。64 を超えるサブトランザクションを生成する可能性のあるトランザクションを特定します。

- セーブポイント使用量の削減 – トランザクションでのセーブポイントの使用を最小限に抑えます。
  - EXCEPTION ブロックを使用して PL/pgSQL の手順と関数を確認します。EXCEPTION ブロックは暗黙的なセーブポイントを自動的に作成し、サブトランザクションのオーバーフローにつながる可能性があります。各 EXCEPTION 句は、実行中に例外が実際に発生したかどうかに関係なく、サブトランザクションを作成します。

    Example

    例 1: 問題のある EXCEPTION ブロックの使用

    次のコード例は、複数のサブトランザクションを作成する問題のある EXCEPTION ブロックの使用を示しています。

```
CREATE OR REPLACE FUNCTION process_user_data()
RETURNS void AS $$
DECLARE
    user_record RECORD;
BEGIN
    FOR user_record IN SELECT * FROM users LOOP
        BEGIN
            -- This creates a subtransaction for each iteration
            INSERT INTO user_audit (user_id, action, timestamp)
            VALUES (user_record.id, 'processed', NOW());

            UPDATE users
            SET last_processed = NOW()
            WHERE id = user_record.id;

        EXCEPTION
            WHEN unique_violation THEN
                -- Handle duplicate audit entries
                UPDATE user_audit
                SET timestamp = NOW()
                WHERE user_id = user_record.id AND action = 'processed';
        END;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

次の改善されたコード例では、例外処理の代わりに UPSERT を使用することで、サブトランザクションの使用を削減しています。

```
CREATE OR REPLACE FUNCTION process_user_data()
RETURNS void AS $$
DECLARE
    user_record RECORD;
BEGIN
    FOR user_record IN SELECT * FROM users LOOP
        -- Use UPSERT to avoid exception handling
        INSERT INTO user_audit (user_id, action, timestamp)
        VALUES (user_record.id, 'processed', NOW())
        ON CONFLICT (user_id, action)
        DO UPDATE SET timestamp = NOW();

        UPDATE users
        SET last_processed = NOW()
        WHERE id = user_record.id;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Example

例 2: STRICT 例外ハンドラー

次のコード例は、NO_DATA_FOUND を使用した EXCEPTION 処理の問題を示しています。

```
CREATE OR REPLACE FUNCTION get_user_email(p_user_id INTEGER)
RETURNS TEXT AS $$
DECLARE
    user_email TEXT;
BEGIN
    BEGIN
        -- STRICT causes an exception if no rows or multiple rows found
        SELECT email INTO STRICT user_email
        FROM users
        WHERE id = p_user_id;
```

```
        RETURN user_email;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN 'Email not found';
    END;
END;
$$ LANGUAGE plpgsql;
```

次の改善されたコード例では、例外処理の代わりに IF NOT FOUND を使用してサブトランザクションを回避します。

```
CREATE OR REPLACE FUNCTION get_user_email(p_user_id INTEGER)
RETURNS TEXT AS $$
DECLARE
    user_email TEXT;
BEGIN
     SELECT email INTO user_email
     FROM users
     WHERE id = p_user_id;

     IF NOT FOUND THEN
         RETURN 'Email not found';
     ELSE
         RETURN user_email;
     END IF;
END;
$$ LANGUAGE plpgsql;
```

- JDBC ドライバー – autosave パラメータを always または conservative に設定すると、各クエリの前にセーブポイントが生成されます。never 設定がアプリケーションに受け入れられるかどうかを評価します。
- PostgreSQL ODBC ドライバー (psqlODBC) — ロールバックレベル設定 (ステートメントレベルのロールバック用) は、ステートメントのロールバック機能を有効にするための暗黙的なセーブポイントを作成します。トランザクションレベルのロールバックがアプリケーションで許容されるかどうかを評価します。
- ORM トランザクション設定を確認する
- セーブポイントを必要としない代替エラー処理戦略を検討する

- トランザクション設計の最適化 – 過剰なネストを回避し、サブトランザクションオーバーフロー条件の可能性を減らすために、トランザクションを再構築します。
- 長時間実行されるトランザクションの削減 – 長時間実行されるトランザクションでは、サブトランザクション情報を長く保持することで、サブトランザクションの問題が悪化する可能性があります。Performance Insights メトリクスをモニタリングし、アイドル状態のトランザクションを自動的に終了するように idle_in_transaction_session_timeout パラメータを設定します。
- Performance Insights メトリクスのモニタリング – idle_in_transaction_count (トランザクション状態でアイドル状態のセッション数) や idle_in_transaction_max_time (アイドル状態にある最長時間のトランザクションの継続時間) などのメトリクスを追跡して、長時間実行されるトランザクションを検出します。
- idle_in_transaction_session_timeout の設定 – 指定した期間後にアイドル状態のトランザクションを自動的に終了するように、パラメータグループにこのパラメータを設定します。
- プロアクティブモニタリング – LWLock:SubtransBuffer および LWLock:SubtransSLRU 待機イベントの発生頻度をモニタリングし、サブトランザクション関連の競合が重大になる前に検出します。

Timeout:PgSleep

Timeout:PgSleepイベントは、サーバプロセスがpg_sleep関数を呼び出し、スリープタイムアウトの期限切れを待っているときに発生します。

トピック

- サポート対象エンジンバージョン
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

待機時間が増加する原因の可能性

この待機イベントは、アプリケーション、ストアド関数、またはユーザーが次のいずれかの関数を呼び出す SQL ステートメントを発行したときに発生します。

- pg_sleep

- pg_sleep_for
- pg_sleep_until

前述の関数は、指定された秒数が経過するまで実行を遅らせます。例えば、SELECT pg_sleep(1) は 1 秒間一時停止します。詳細については、PostgreSQL のドキュメントの「実行の遅延」を参照してください。

アクション

pg_sleep関数を実行していたステートメントを特定します。関数の使用が適切かどうかを判断します。

Timeout:VacuumDelay

Timeout:VacuumDelay イベントは、バキューム I/O のコスト制限を超え、バキュームプロセスがスリープ状態になったことを示します。バキュームオペレーションは、それぞれのコスト遅延パラメータで指定された期間停止し、その後動作を再開します。手動バキュームコマンドの場合、遅延は vacuum_cost_delay パラメータで指定されます。自動バキュームデーモンの場合、遅延は autovacuum_vacuum_cost_delay parameter. で指定されます。

トピック

- サポート対象エンジンバージョン
- Context
- 待機時間が増加する原因の可能性
- アクション

サポート対象エンジンバージョン

この待機イベント情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

PostgreSQL には、自動バキュームデーモンと手動バキュームコマンドがあります。自動バキュームプロセスは、RDS for PostgreSQL DB インスタンスでデフォルトで「オン」になっています。手動バキュームコマンドは、dead タプルのテーブルの削除や、新しい統計の生成など、必要に応じて使用されます。

バキューム処理中は、PostgreSQL では内部カウンターを使用して、システムがさまざまな I/O オペレーションを実行する際の推定コストを追跡します。カウンターがコスト制限パラメータで指定された値に達すると、そのオペレーションを実行するプロセスは、コスト遅延パラメータで指定された短時間の間スリープ状態になります。その後、カウンターをリセットしてオペレーションを続行します。

バキュームプロセスには、リソース消費量を調整するために使用できるパラメータがあります。自動バキュームコマンドと手動バキュームコマンドには、コスト制限値を設定するための独自のパラメータがあります。コスト遅延、つまり制限に達したときにバキュームをスリープ状態にする時間を指定する独自のパラメータもあります。このように、コスト遅延パラメータは、リソース消費のスロットリングメカニズムとして機能します。以下の一覧で、これらのパラメータの説明を参照できます。

自動バキュームデーモンのスロットリングに影響するパラメータ

- autovacuum_vacuum_cost_limit – 自動バキュームオペレーションで使用するコスト制限値を指定します。このパラメータの設定を増やすと、バキュームプロセスが使用するリソースが増え、Timeout:VacuumDelay 待機イベントが減ります。
- autovacuum_vacuum_cost_delay – 自動バキュームオペレーションで使用するコスト遅延値を指定します。デフォルト値は 2 ミリ秒です。遅延パラメータを 0 に設定すると、スロットリングメカニズムがオフになり、Timeout:VacuumDelay 待機イベントは表示されなくなります。

詳細については、「PostgreSQL のドキュメント」の「Automatic Vacuuming」(自動バキューム処理) を参照してください。

手動バキュームプロセスのスロットリングに影響するパラメータ

- vacuum_cost_limit – バキューム処理をスリープ状態にするしきい値。デフォルトの制限値は 200 です。この数値は、さまざまなリソースが必要とする追加の I/O の累積コスト見積もりを表します。この値を増やすと、Timeout:VacuumDelay 待機イベントの数が減ります。
- vacuum_cost_delay – バキュームのコスト制限に達したときに、バキュームプロセスがスリープになる時間。デフォルト設定は 0 で、この機能はオフになります。この機能を有効にするミリ秒数を整数値で指定できますが、デフォルト設定のままにしておくことをお勧めします。

vacuum_cost_delay パラメータの詳細については、PostgreSQL ドキュメントの「リソース消費」を参照してください。

RDS for PostgreSQL で自動バキュームを設定し、使用する方法の詳細については、「Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用」を参照してください。

待機時間が増加する原因の可能性

Timeout:VacuumDelay は、バキュームのスリープ時間を制御するコスト制限パラメータ設定 (vacuum_cost_limit、autovacuum_vacuum_cost_limit) と、コスト遅延パラメータ (vacuum_cost_delay、autovacuum_vacuum_cost_delay) のバランスの影響を受けます。コスト制限のパラメータ値を上げると、バキュームがスリープ状態になる前に利用できるリソースが増えます。その結果、Timeout:VacuumDelay 待機イベントが少なくなります。いずれかの遅延パラメータを増やすと、これまでよりも頻繁に、長期間 Timeout:VacuumDelay 待機イベントが発生します。

autovacuum_max_workers パラメータ設定により、Timeout:VacuumDelay の数を増やすこともできます。自動バキュームワーカープロセスを追加するごとに内部カウンターメカニズムに影響を与えるため、単一の自動バキュームワーカープロセスを使用する場合よりも早く上限に達する場合があります。コスト制限に早く達するとコスト遅延がより頻繁に発生し、結果的に Timeout:VacuumDelay 待機イベントが増えます。詳細については、PostgreSQL ドキュメントの「autovacuum_max_workers」を参照してください。

バキュームが大きなオブジェクトの処理を完了するまでに時間がかかることがあるため、500 GB 以上の大きなオブジェクトでもこの待機イベントが発生します。

アクション

想定どおりにバキュームオペレーションが完了すれば、修復は不要です。つまり、この待機イベントは、必ずしも問題を示すわけではありません。これは、処理を完了する必要のある他のプロセスにリソースを割り当てることができるように、バキュームが遅延パラメータで指定された時間だけスリープ状態になっていることを示しています。

バキュームオペレーションをより早く完了させる場合は、遅延パラメータを下げることができます。これにより、バキュームのスリープ時間が短縮されます。

Amazon DevOps Guru のプロアクティブインサイトによる RDS for PostgreSQL のチューニング

DevOps Guru のプロアクティブインサイトは、問題の原因となる可能性がある RDS for PostgreSQL DB インスタンスの条件を検出して、問題が発生する前に通知します。プロアクティブインサイトは、トランザクション接続で長時間アイドル状態になっている場合にアラートを送信できます。トランザクション接続で長時間アイドル状態になっている場合のトラブルシューティングの詳細については、「データベースがトランザクション接続で長時間アイドル状態になっている」を参照してください。

DevOps Guru では、次のことができます。

- データベース構成を一般的な推奨設定と照合することで、データベースに関する多くの一般的な問題を防ぎます。
- 未チェックのままにしておくと、後で大きな問題につながる可能性があるフリート内の重大な問題について警告します。
- 新しく発見された問題について警告します。

すべてのプロアクティブインサイトには、問題の原因の分析と是正措置の推奨事項が含まれています。

Amazon DevOps Guru for Amazon RDS の詳細については、「Amazon DevOps Guru for Amazon RDS でパフォーマンスの異常を分析する」を参照してください。

データベースがトランザクション接続で長時間アイドル状態になっている

データベースへの接続が 1800 秒以上 `idle in transaction` 状態です。

トピック

- サポート対象エンジンバージョン
- Context
- この問題の考えられる原因
- アクション
- 関連するメトリクス

サポート対象エンジンバージョン

このインサイト情報は、RDS for PostgreSQL のすべてのバージョンでサポートされています。

Context

`idle in transaction` 状態のトランザクションがロックを保持していて、他のクエリをブロックしている可能性があります。また、VACUUM (自動バキュームを含む) がデッド行をクリーンアップするのを妨げて、インデックスやテーブルが肥大化したり、トランザクション ID がラップアラウンドしたりします。

この問題の考えられる原因

インタラクティブセッションで BEGIN または START TRANSACTION を使用して開始されたトランザクションが、COMMIT、ROLLBACK、または END コマンドを使用しても終了していません。これにより、トランザクションは `idle in transaction` 状態に移行します。

アクション

`pg_stat_activity` クエリを実行すると、アイドル状態のトランザクションを見つけることができます。

SQL クライアントで、次のクエリを実行して、`idle in transaction` 状態にあるすべての接続を一覧表示し、継続時間順に並べ替えます。

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

インサイトの原因に応じて、異なるアクションをお勧めします。

トピック

- 接続を終了する
- 接続を作成する
- idle_in_session_timeout パラメータを設定する
- AUTOCOMMIT のステータスを確認する

- アプリケーションコード内のトランザクションロジックを確認する

接続を終了する

インタラクティブセッションで BEGIN または START TRANSACTION を使用してトランザクションを開始すると、トランザクションは `idle in transaction` 状態に移行します。COMMIT、ROLLBACK、END コマンドを実行してトランザクションを終了するか、接続を完全に切断してトランザクションをロールバックするまで、この状態のままになります。

接続を作成する

次のクエリを使用して、アイドル状態のトランザクションがある接続を終了します。

```
SELECT pg_terminate_backend(pid);
```

pid は接続のプロセス ID です。

idle_in_session_timeout パラメータを設定する

パラメータグループの `idle_in_transaction_session_timeout` パラメータを設定します。このパラメータを設定する利点は、手動操作を行わなくても、長時間アイドル状態になっているトランザクションを終了できることです。このパラメータの詳細については、「PostgreSQL documentation」(PostgreSQL ドキュメント) を参照してください。

接続が終了し、指定した時間を超えてトランザクションが idle_in_transaction 状態にあると、PostgreSQL ログファイルに次のメッセージが報告されます。

```
FATAL: terminating connection due to idle in transaction timeout
```

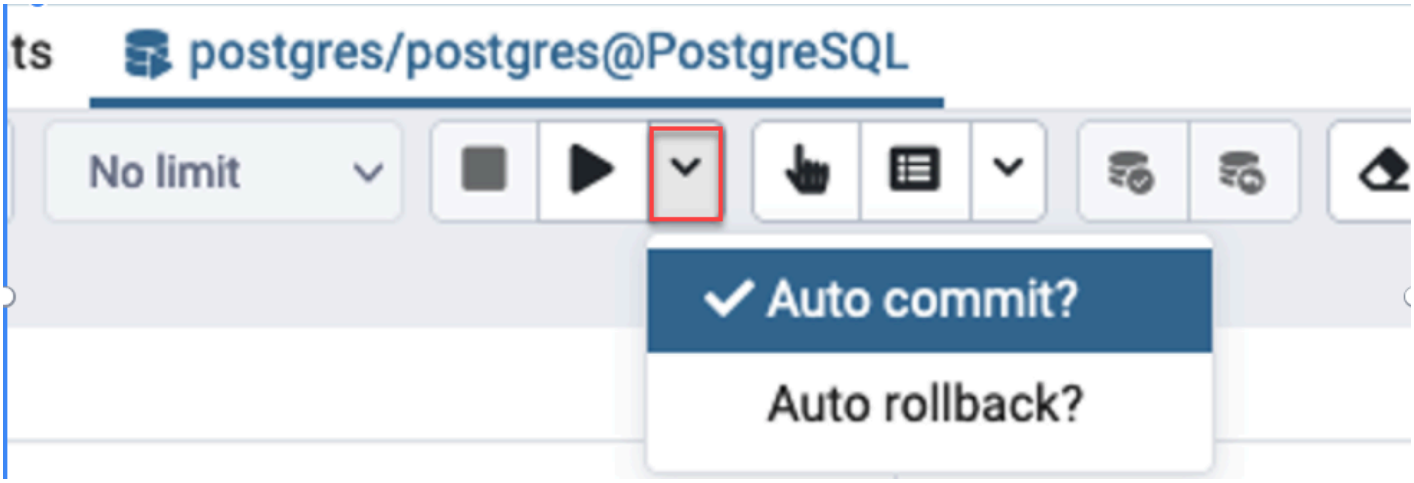
AUTOCOMMIT のステータスを確認する

AUTOCOMMIT は、デフォルトで有効になっています。ただし、クライアントで誤ってオフにした場合は、必ずオンに戻してください。

- psql クライアントで次のコマンドを実行します。

```
postgres=> \set AUTOCOMMIT on
```

- pgadmin で、下矢印から AUTOCOMMIT オプションを選択してオンにします。



アプリケーションコード内のトランザクションロジックを確認する

アプリケーションロジックを調べ手、問題がない確認します。以下のアクションの場合を検討します。

- アプリケーションで JDBC auto commit が true に設定されているかどうかを確認します。また、コード内で明示的な `COMMIT` コマンドを使用することも検討してください。
- エラー処理ロジックをチェックして、エラー後にトランザクションがクローズされるかどうかを確認します。
- トランザクションが開いているときに、アプリケーションがクエリによって返された行の処理に時間がかかるかどうかを確認します。その場合は、行を処理する前にトランザクションを閉じるようにアプリケーションをコーディングすることを検討してください。
- トランザクションに長時間実行される操作が多数含まれていないか確認します。その場合は、1 つのトランザクションを複数のトランザクションに分割します。

関連するメトリクス

以下の PI メトリクスがこのインサイトに関連しています。

- idle_in_transaction_count - `idle in transaction` 状態にあるセッション数。
- idle_in_transaction_max_time - `idle in transaction` 状態で最も長く実行されているトランザクションの継続時間。

Amazon RDS for PostgreSQL で PostgreSQL 拡張機能を使用する

PostgreSQL は、さまざまな拡張機能やモジュールをインストールすることで、機能を拡張することができます。例えば、空間データを操作するには、PostGIS 拡張機能をインストールして使用します。詳細については、「PostGIS 拡張機能を使用した空間データの管理」を参照してください。別の例として、非常に大きなテーブルへのデータ入力を改善する場合は、pg_partman 拡張機能を使用したデータのパーティション化を検討できます。詳細については pg_partman エクステンションによる PostgreSQL パーティションの管理を参照してください。

> **Note**
>
> RDS for PostgreSQL は、DB インスタンスに追加できる pg_tle 拡張機能を通じて、Trusted Language Extensions for PostgreSQL をサポートしています。この拡張を使用することで、開発者は安全な環境で独自の PostgreSQL 拡張を作成できるため、セットアップと設定の要件が簡素化されます。pg_tle 拡張機能をサポートする RDS for PostgreSQL のバージョンと詳細については、「Trusted Language Extensions for PostgreSQL を使用した操作」を参照してください。

場合によっては、拡張機能をインストールする代わりに、Aurora PostgreSQL DB クラスターのカスタム DB クラスターパラメータグループの shared_preload_libraries リストに特定のモジュールを追加することもできます。通常、デフォルトの DB クラスターパラメータグループでは、pg_stat_statements のみが読み込まれますが、リストに追加できるモジュールは他にもいくつかあります。例えば、PostgreSQL pg_cron エクステンションによるメンテナンスのスケジューリング で説明されているように、pg_cron モジュールを追加することでスケジュール機能を追加できます。別の例として、auto_explain モジュールをロードすることでクエリ実行計画を記録できます。詳細については、AWS ナレッジセンターの「クエリ実行計画の口グ記録」をご覧ください。

RDS for PostgreSQL のバージョンによっては、拡張機能をインストールする際に、以下のような rds_superuser の権限が必要になる場合があります。

- RDS for PostgreSQL バージョン 12 以前のバージョンでは、拡張機能をインストールする際に rds_superuser の権限が必要となります。
- RDS for PostgreSQL バージョン 13 以降のバージョンでは、特定のデータベースインスタンスに対する作成権限を持つユーザー (ロール) は、信頼できる拡張機能をインストールして使用することができます。信頼できる拡張機能のリストについては、「PostgreSQL 信頼できるエクステンション」を参照してください。

また、RDS for PostgreSQL DBインスタンスにインストール可能な拡張機能は、rds.allowed_extensions パラメータにリストアップして、正確に指定することができます。詳細については、「PostgreSQL エクステンションのインストールを制限する」を参照してください。

rds_superuser ロールの詳細については、「PostgreSQL のロールとアクセス権限について」を参照してください。

トピック

- orafce 拡張機能の関数の使用
- PostgreSQL での Amazon RDS 委任拡張機能サポートの使用
- pg_partman エクステンションによる PostgreSQL パーティションの管理
- pgAudit を使用してデータベースのアクティビティを記録する
- PostgreSQL pg_cron エクステンションによるメンテナンスのスケジューリング
- pglogical を使用してインスタンス間でデータを同期する
- pgactive を使用したアクティブ/アクティブレプリケーションのサポート
- pg_repack 拡張機能を使用して、テーブルやインデックスの膨張を抑制する
- PLV8 拡張機能のアップグレードおよび使用
- PL/Rust を使って Rust 言語で PostgreSQL 関数を記述する
- PostGIS 拡張機能を使用した空間データの管理

orafce 拡張機能の関数の使用

orafce 拡張機能は、Oracle データベースから関数とパッケージのサブセットをエミュレートする関数と演算子を提供します。Oracle 拡張機能を使用すると、Oracle アプリケーションを PostgreSQL に簡単に移植できます。この拡張機能は、RDS for PostgreSQL バージョン 9.6.6 以降でサポートされています。orafce についての詳細は、GitHub で「orafce」を参照してください。

> **Note**
>
> RDS for PostgreSQL では、orafce 拡張機能の一部である utl_file パッケージがサポートされていません。これは、utl_file スキーマ関数が、基になるモストへのスーパーユーザーアクセスに必要なオペレーティングシステムテキストファイルで読み書き操作を実行するためです。マネージド型サービスの RDS for PostgreSQL では、ホストアクセスが許可されません。

orafce エクステンションを使用するには

1. DB インスタンスの作成で使用したプライマリユーザー名を使用して DB インスタンスに接続します。

   同じ DB インスタンスにある別のデータベースで orafce をオンにする場合は、/c dbname psql コマンドを使用します。このコマンドを使用すると、接続を開始した後にプライマリデータベースから変更できます。
2. CREATE EXTENSION ステートメントを使用して、orafce 拡張機能をオンにします。

   ```
   CREATE EXTENSION orafce;
   ```
3. ALTER SCHEMA ステートメントを使用して、oracle スキーマの所有権を rds_superuser ロールに転送します。

   ```
   ALTER SCHEMA oracle OWNER TO rds_superuser;
   ```

   oracle スキーマの所有者のリストを表示する場合は、\dn psql コマンドを使用します。

PostgreSQL での Amazon RDS 委任拡張機能サポートの使用

PostgreSQL に対して Amazon RDS 委任拡張機能のサポートを使用すると、拡張機能管理を `rds_superuser` である必要のないユーザーに委任できます。この委任拡張機能のサポートにより、`rds_extension` という新しいロールが作成され、他の拡張機能を管理するには、これをユーザーに割り当てる必要があります。このロールは、拡張機能を作成、更新、削除できます。

RDS DB インスタンスにインストール可能な拡張機能は、`rds.allowed_extensions` パラメータにそれらをリストアップして指定することができます。詳細については、「Amazon RDS for PostgreSQL で PostgreSQL 拡張機能を使用する」を参照してください。

`rds.allowed_delegated_extensions` パラメータを使用して、`rds_extension` ロールでユーザーが管理できる拡張機能のリストを制限できます。

委任拡張機能のサポートは、次のバージョンで利用できます。

- すべての上位バージョン
- 16.4 以降の 16 バージョン
- 15.8 以降の 15 バージョン
- 14.13 以降の 14 バージョン
- 13.16 以降の 13 バージョン
- 12.20 以降の 12 バージョン

トピック

- ユーザーに対する委任拡張機能のサポートの有効化
- PostgreSQL での RDS 委任拡張機能サポートで使用される設定
- 委任拡張機能のサポートの無効化
- Amazon RDS 委任拡張機能サポートの使用のメリット
- PostgreSQL での Amazon RDS 委任拡張機能サポートの制限
- 特定の拡張機能に必要なアクセス許可
- セキュリティに関する考慮事項
- DROP EXTENSION CASCADE を無効化
- 委任拡張機能サポートを使用して追加できる拡張機能の例

ユーザーに対する委任拡張機能のサポートの有効化

ユーザーに対して委任拡張機能のサポートを有効にするには、以下を実行する必要があります。

1. ユーザーに **rds_extension** ロールを付与する – rds_superuser としてデータベースに接続し、次のコマンドを実行します。

```
Postgres => grant rds_extension to user_name;
```

2. 委任されたユーザーが管理できる拡張機能のリストを設定する – rds.allowed_delegated_extensions では、DB クラスターパラメータで rds.allowed_extensions を使用して、使用可能な拡張機能のサブセットを指定できます。これは、次のいずれかのレベルで実行できます。

   - クラスターまたはインスタンスパラメータグループで、AWS マネジメントコンソール または API を使用します。詳細については、「Amazon RDS のパラメータグループ」を参照してください。
   - データベースレベルで次のコマンドを使用します。

```
alter database database_name set rds.allowed_delegated_extensions =
 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

   - ユーザーレベルで次のコマンドを使用します。

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

> **Note**
>
> rds.allowed_delegated_extensions 動的パラメータを変更した後にデータベースを再起動する必要はありません。

3. 拡張機能の作成プロセス中に作成されたオブジェクトへのアクセスを委任されたユーザーに許可する – 特定の拡張機能では、rds_extension ロールを持つユーザーがオブジェクトにアクセスする前に、追加のアクセス許可を付与する必要があるオブジェクトが作成されます。rds_superuser は、それらのオブジェクトへのアクセス権を委任されたユーザーに付与

する必要があります。オプションの 1 つは、イベントトリガーを使用して、委任されたユーザーにアクセス許可を自動的に付与することです。

イベントトリガーの例

rds_extension を持つ委任ユーザーに、拡張機能の作成によって作成されたオブジェクトに対するアクセス許可の設定を必要とする拡張機能の使用を許可する場合は、次のイベントトリガーの例をカスタマイズし、委任されたユーザーに完全な機能へのアクセスを許可する拡張機能のみを追加できます。このイベントトリガーは template1 (デフォルトのテンプレート) で作成できるため、template1 から作成されたすべてのデータベースにそのイベントトリガーがあります。委任されたユーザーが拡張機能をインストールすると、このトリガーは拡張機能によって作成されたオブジェクトの所有権を自動的に付与します。

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;

BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_catalog.pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';

        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );
```

```
        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
 schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
 schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_gaz TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_lex TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_rules TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE pg_catalog.format('ALTER TEXT SEARCH DICTIONARY %I.intdict
 OWNER TO %I;', schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE
 ON TABLE %I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname,
 databaseowner);
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL
 TABLES IN SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE, SELECT ON ALL SEQUENCES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA
 topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE ON SCHEMA topology TO %I WITH
 GRANT OPTION;', databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE
 create_ext();
```

PostgreSQL での RDS 委任拡張機能サポートで使用される設定

設定名	説明	デフォルト値	注意事項	アクセス許可を変更または付与できるユーザー
rds.allowed_delegated_extensions	このパラメータは、rds_extension ロールがデータベースで管理できる拡張機能を制限します。rds.allowed_extensions のサブセットである必要があります。	空の文字列	• デフォルトでは、このパラメータは空の文字列です。つまり、rds_extension を持つユーザーに拡張機能が委任されていません。 • ユーザーにアクセス許可がある場合は、サポートされている拡張機能を追加できます。これを実行するためには、rds.allowed_delegated_extensions パラメータをカンマで区切った拡張子名の文字列に設定します。このパラメータに拡張機能のリストを追加すると、rds_extension ロールの	rds_superuser

設定名	説明	デフォルト値	注意事項	アクセス許可を変更または付与できるユーザー
			あるユーザーがインストールできる拡張機能を明示的に特定できます。 • * に設定すると、rds_allowed_extensions にリストされているすべての拡張機能が rds_extension ロールを持つユーザーに委任されます。 このパラメータの設定の詳細については、「ユーザーに対する委任拡張機能のサポートの有効化」を参照してください。	

設定名	説明	デフォルト値	注意事項	アクセス許可を変更または付与できるユーザー
rds.allowed_extensions	このパラメータにより、カスタマーは RDS DB インスタンスにインストールできる拡張機能を制限できます。詳細については、「PostgreSQL 拡張機能のインストールの制限」を参照してください。	"*"	デフォルトでは、このパラメータは「*」に設定されています。つまり、RDS for PostgreSQL および Aurora PostgreSQL でサポートされているすべての拡張機能は、必要な権限を持つユーザーが作成できます。 空の場合、RDS DB インスタンスに拡張機能をインストールできないことを意味します。	管理者

設定名	説明	デフォルト値	注意事項	アクセス許可を変更または付与できるユーザー
rds-delegated_nsion_ow_drascac	このパラメータは、rds_extension を持つユーザーがカスケードオプションを使用して拡張機能を削除する機能を制御します。	化	デフォルトで、rds-delegated_extensionallow_drop_cascade は off に設定されています。つまり、rds_extension を持つユーザーは、カスケードオプションを使用して拡張機能を削除することはできません。 その機能を許可するには、rds.delegated_extensionallow_drop_cascade パラメータを on に設定する必要があります。	rds_superuser

委任拡張機能のサポートの無効化

部分的にオフにする

委任されたユーザーは、新しい拡張機能を作成することはできませんが、既存の拡張機能を更新することはできます。

- DB クラスターパラメータグループでデフォルト値に rds.allowed_delegated_extensions をリセットします。
- データベースレベルで次のコマンドを使用します。

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- ユーザーレベルで次のコマンドを使用します。

```
alter user user_name reset rds.allowed_delegated_extensions;
```

すべてオフにする

ユーザーから rds_extension ロールを取り消すと、ユーザーは標準のアクセス許可に戻ります。ユーザーは拡張機能を作成、更新、削除できなくなります。

```
postgres => revoke rds_extension from user_name;
```

Amazon RDS 委任拡張機能サポートの使用のメリット

PostgreSQL に対して Amazon RDS 委任拡張機能のサポートを使用すると、拡張機能管理を rds_superuser ロールを持たないユーザーに対してセキュアに委任できます。この機能には次の利点があります。

- 選択したユーザーに拡張機能管理を簡単に委任できます。
- これには rds_superuser ロールは不要です。
- 同じ DB クラスター内の異なるデータベースに対して、異なる拡張機能セットをサポートする機能を提供します。

PostgreSQL での Amazon RDS 委任拡張機能サポートの制限

- 拡張機能の作成プロセス中に作成されたオブジェクトは、拡張機能が正しく機能するために追加の権限が必要になる場合があります。
- 一部の拡張機能は、デフォルトで log_fdw、pg_cron、pg_tle、pgactive、pglogical、postgis_raster、postgis_tiger_geoc などの委任された拡張機能ユーザーが管理することはできません。

特定の拡張機能に必要なアクセス許可

次の拡張機能を作成、使用、または更新するには、委任されたユーザーに次の関数、テーブル、スキーマに対する必要な権限が必要です。

所有権またはアクセス許可が必要な拡張機能	関数	テーブル	スキーマ	テキスト検索ディクショナリ	Comment
addr tand er_d s	なし	us_gaz、us _lex、us_l ex、l.us_rules	なし	なし	なし
amch	bt_index_ check、bt_ index_par ent_check	なし	なし	なし	なし
dict_i	なし	なし	なし	intdict	なし

所有権またはアクセス許可が必要な拡張機能	関数	テーブル	スキーマ	テキスト検索ディクショナリ	Comment
pg_pn	なし	custom_time_partitions、part_config、part_config_sub	なし	なし	なし
pg_statem	なし	なし	なし	なし	なし
Post(	st_tileenvelope	spatial_ref_sys	なし	なし	なし
postg aster	なし	なし	なし	なし	なし

所有権またはアクセス許可が必要な拡張機能	関数	テーブル	スキーマ	テキスト検索ディクショナリ	Comment
postg opolo	なし	トポロジ、レイヤー	トポロジ	なし	委任されたユーザーはデータベース所有者であること
log_f	create_foreign_table_for_log_file	なし	なし	なし	なし
rds_t	role_password_encryption_type	なし	なし	なし	なし

所有権またはアクセス許可が必要な拡張機能	関数	テーブル	スキーマ	テキスト検索ディクショナリ	Comment
postg iger_ oder	なし	geocode_s ettings_default, geocode_s ettings	tiger	なし	なし
pg_fr acem	pg_freespace	なし	なし	なし	なし
pg_vi lity	pg_visibility	なし	なし	なし	なし

セキュリティに関する考慮事項

`rds_extension` ロールを持つユーザーは、接続権限を持つすべてのデータベースの拡張機能を管理できることに注意してください。委任されたユーザーが 1 つのデータベースの拡張機能を管理す

る場合は、各データベースのパブリックからすべての権限を取り消し、その特定のデータベースの接続権限を委任されたユーザーに明示的に付与することをお勧めします。

ユーザーが複数のデータベースから情報にアクセスできる拡張機能がいくつかあります。これらの拡張機能を rds.allowed_delegated_extensions に追加する前に、rds_extension を付与するユーザーにクロスデータベース機能があることを確認してください。例えば、postgres_fdw と dblink は、同じインスタンスまたはリモートインスタンス上のデータベース間でクエリを実行する機能を提供します。log_fdw は postgres エンジンのログファイルを読み取ります。これらは、インスタンス内のすべてのデータベース用であり、複数のデータベースからのスロークエリやエラーメッセージが含まれている可能性があります。pg_cron は、DB インスタンスでスケジュールされたバックグラウンドジョブの実行を有効にし、別のデータベースで実行するようにジョブを設定できます。

DROP EXTENSION CASCADE を無効化

rds_extension ロールを持つユーザーがカスケードオプションを使用して拡張機能を削除する機能は、rds.delegated_extension_allow_drop_cascade パラメータによって制御されます。デフォルトで、rds-delegated_extension_allow_drop_cascade は off に設定されています。つまり、rds_extension ロールを持つユーザーは、以下のクエリに示すように、カスケードオプションを使用して拡張機能を削除することはできません。

```
DROP EXTENSION CASCADE;
```

これにより、拡張機能に依存するオブジェクト、およびそれらのオブジェクトに依存するすべてのオブジェクトが自動的に削除されます。カスケードオプションを使用しようとすると、エラーが発生します。

その機能を許可するには、rds.delegated_extension_allow_drop_cascade パラメータを on に設定する必要があります。

rds.delegated_extension_allow_drop_cascade 動的パラメータを変更しても、データベースを再起動する必要はありません。これは、次のいずれかのレベルで実行できます。

- クラスターまたはインスタンスパラメータグループで、AWS マネジメントコンソール または API を使用します。
- データベースレベルで次のコマンドを使用する。

```
alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
```

- ユーザーレベルで次のコマンドを使用する。

```
alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
```

委任拡張機能サポートを使用して追加できる拡張機能の例

- rds_tools

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- amcheck

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- pg_freespacemap

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- pg_visibility

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- postgres_fdw

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

pg_partman エクステンションによる PostgreSQL パーティションの管理

PostgreSQL テーブルパーティションは、データ入力とレポートにおける高性能な処理のためのフレームワークを提供します。大量のデータを非常に速く入力する必要があるデータベースには、パーティションを使用します。またパーティションは、大きなテーブルでより高速のクエリを提供します。パーティションは、必要とする I/O リソースが少なくて済むため、データベースインスタンスに影響を与えずにデータを維持するのに役立ちます。

パーティションを使用すると、データをカスタムサイズのチャンクに分割して処理することができます。例えば、時間単位、日単位、週単位、月単位、四半期単位、年単位、カスタム、またはこれらの組み合わせなどの範囲のパーティションの時系列データを選択できます。時系列データの例では、テーブルを時間単位でパーティション化した場合、各パーティションには 1 時間のデータが含まれます。時系列テーブルを日単位でパーティション化した場合、パーティションには 1 日分のデータが保持されます。パーティションキーは、パーティションのサイズを制御します。

パーティション化されたテーブルで `INSERT` SQL コマンドまたは `UPDATE` SQL コマンドを使用すると、データベースエンジンはデータを適切なパーティションにルーティングします。データを格納する PostgreSQL テーブルパーティションは、メインテーブルの子テーブルです。

データベースクエリの読み取り中、PostgreSQL オプティマイザはクエリの `WHERE` 句を調べ、可能な場合、関連するパーティションだけにデータベーススキャンを行うよう指示します。

バージョン 10 以降、PostgreSQL は宣言的なパーティショニングを使用してテーブルパーティションを実装します。これは、ネイティブ PostgreSQL パーティションとも言います。PostgreSQL バージョン 10 より前は、トリガーを使用してパーティションを実装していました。

PostgreSQL テーブルパーティションは、次の機能を提供します。

- 新しいパーティションの作成がいつでも可能。
- 可変のパーティション範囲。
- データ定義言語 (DDL) ステートメントを使用して、取り外し可能かつ再接続可能なパーティション。

  例えば、デタッチ可能なパーティションは、メインパーティションから履歴データを削除し、履歴データを分析用に保持する場合に便利です。
- 新しいパーティションは、親のデータベーステーブルの以下のプロパティを継承します。
  - インデックス
  - プライマリキー (パーティションキー列を含める必要があります)

  - 外部キー
  - 検査制約
  - 参照
- フルテーブルまたは各特定のパーティションのインデックスの作成。

個々のパーティションのスキーマを変更することはできません。ただし、パーティションに伝播される親テーブル (新しい列の追加など) は変更できます。

トピック

- PostgreSQL pg_partman エクステンションの概要
- pg_partman エクステンションの有効化
- create_parent 関数を使用したパーティションの設定
- run_maintenance_proc 関数を使用したパーティションのメンテナンス設定

PostgreSQL pg_partman エクステンションの概要

PostgreSQL pg_partman エクステンションを使用すると、テーブルパーティションの作成とメンテナンスを自動化できます。一般的な情報については、pg_partman ドキュメントの「PG Partition Manager」を参照してください。

> **Note**
>
> pg_partman エクステンションは、RDS for PostgreSQL のバージョン 12.5 以降でサポートされています。

各パーティションを手動で作成する代わりに、次の設定で pg_partman を設定します。

- パーティション化するテーブル
- パーティションタイプ
- パーティションキー
- パーティションの粒度
- パーティションの事前作成および管理オプション

PostgreSQL のパーティション化されたテーブルの作成後、create_parent 関数を呼び出して、そのテーブルを pg_partman に登録します。これにより、関数に渡すパラメータに基づいて、必要なパーティションを作成します。

pg_partman エクステンションには、設定したスケジュールに基づいて呼び出しを行うことでパーティションを自動的に管理できる run_maintenance_proc 関数も用意されています。必要に応じて適切なパーティションが作成されるようにするには、この関数を定期的に (時間単位など) 実行するようにスケジュールします。また、パーティションが自動的に削除されるようにすることもできます。

pg_partman エクステンションの有効化

パーティションを管理する同じ PostgreSQL DB インスタンス内に複数のデータベースがある場合は、データベースごとに pg_partman エクステンションを有効にします。特定のデータベースで pg_partman エクステンションを有効にするには、パーティションメンテナンススキーマを作成した上で、次のように pg_partman エクステンションを作成します。

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> pg_partman エクステンションを作成するには、rds_superuser 権限が必要です。

次のようなエラーが表示された場合は、アカウントに rds_superuser 権限を付与するか、スーパーユーザーアカウントを使用します。

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

rds_superuser 権限を付与するには、スーパーユーザーアカウントを使用して接続し、以下のコマンドを実行します。

```
GRANT rds_superuser TO user-or-role;
```

pg_partman エクステンションの使用方法を示す例では、次のサンプルのデータベーステーブルとパーティションを使用します。このデータベースでは、タイムスタンプに基づいてパーティション化

されたテーブルを使用します。スキーマ data_mart には、events という列を持つ created_at という名前のテーブルが含まれています。この events テーブルには、次の設定が含まれています。

- プライマリキー event_id および created_at。パーティションのガイドに使用される列を含める必要があります。
- ck_valid_operation テーブル列に値を適用するための検査制約 operation。
- 2 つの外部キー。1 つ (fk_orga_membership)) は外部テーブル organization で、もう 1 つ (fk_parent_event_id) は自己参照外部キーです。
- 2 つのインデックス。1 つ (idx_org_id) は外部キー用で、もう 1 つ (idx_event_type) はイベントタイプ用です。

次の DDL ステートメントは、各パーティションに自動的に含まれるこれらのオブジェクトを作成します。

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
```

```
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

create_parent 関数を使用したパーティションの設定

pg_partman エクステンションを有効にした後、create_parent 関数を使用して、パーティションメンテナンススキーマ内でパーティションの設定を行います。以下の例では、events で作成される pg_partman エクステンションの有効化 テーブルの例を使用します。create_parent 関数を次のように呼び出します。

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

パラメータは次のとおりです。

- p_parent_table - 親パーティションテーブル。このテーブルは既に存在しており、スキーマを含めて完全修飾である必要があります。
- p_control - パーティションのベースとなる列。データタイプは、整数または時間ベースである必要があります。
- p_type - タイプは 'range' または 'list' です。
- p_interval - 各パーティションの時間間隔または整数の範囲。値の例は、1 day、1 hour などです。
- p_premake - 新しい挿入をサポートするために事前に作成するパーティションの数。

create_parent 関数の詳細については、pg_partman ドキュメントの「Creation Functions (関数の作成)」を参照してください。

run_maintenance_proc 関数を使用したパーティションのメンテナンス設定

パーティションのメンテナンスオペレーションを実行して、自動的に新しいパーティションの作成、パーティションのデタッチ、または古いパーティションの削除ができます。パーティションのメンテナンスは、内部のスケジューラをスタートする pg_partman および pg_cron エクステンション の run_maintenance_proc 関数により異なります。pg_cron スケジューラは、データベースで定義された SQL ステートメント、関数、および手順を自動的に実行します。

次の例では、events で作成した pg_partman エクステンションの有効化 テーブルの例を使用して、パーティションのメンテナンスオペレーションを自動的に実行するように設定します。前提条件にあるように、DB インスタンスのパラメータグループで、shared_preload_libraries パラメータに pg_cron を追加します。

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

その後、前の例についてのステップバイステップの説明を確認できます。

1. DB インスタンスに関連付けられているパラメータグループを変更して、pg_cron を shared_preload_libraries パラメータ値に追加します。この変更を有効にするには、DB インスタンスの再起動が必要です。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。
2. CREATE EXTENSION pg_cron; のアクセス許可を持つアカウントを使用して、コマンド rds_superuser を実行します。これにより、pg_cron エクステンションが有効になります。詳細については、「PostgreSQL pg_cron エクステンションによるメンテナンスのスケジューリング」を参照してください。
3. UPDATE partman.part_config コマンドを実行して、data_mart.events テーブルの pg_partman 設定を調整します。
4. SET . . .コマンドを実行して、以下の句を使用しながら、data_mart.events テーブルを設定します。
   a. infinite_time_partitions = true, - 制限なしで新しいパーティションを自動的に作成できるようにテーブルを設定します。
   b. retention = '3 months', - テーブルの最大保持期間を 3 か月に設定します。
   c. retention_keep_table=true - 保存期間の期限が過ぎてもテーブルが自動的に削除されないように、テーブルを構成します。代わりに、保持期間より古いパーティションは、親テーブルからのみデタッチされます。
5. SELECT cron.schedule . . .コマンドを実行して、pg_cron 関数を呼び出します。この呼び出しは、pg_partman メンテナンスプロシージャの partman.run_maintenance_proc が、スケ

ジューラにより実行される頻度を定義します。この例では、プロシージャは 1 時間ごとに実行されます。

`run_maintenance_proc` 関数の詳細については、`pg_partman` ドキュメントの「Maintenance Functions (メンテナンス機能)」を参照してください。

pgAudit を使用してデータベースのアクティビティを記録する

金融機関、政府機関、および多くの業界では、規制要件を満たすために監査ログを保存する必要があります。 RDS for PostgreSQL DB インスタンスで PostgreSQL 監査拡張機能 (pgAudit) を使用することで、監査人が通常必要とする詳細なレコードや規制要件を満たすための詳細なレコードをキャプチャできます。例えば、pgAudit 拡張機能を設定して、特定のデータベースやテーブルに加えられた変更を追跡したり、変更を加えたユーザーやその他の多くの詳細を記録したりできます。

pgAudit 拡張機能は、ログメッセージをより詳細に拡張することにより、ネイティブの PostgreSQL ログ記録インフラストラクチャの機能に基づいて構築されています。つまり、監査ログは、他のログメッセージを表示するのと同じ方法を使用します。PostgreSQL ログ記録の詳細については、「RDS for PostgreSQL データベースログファイル」を参照してください。

pgAudit 拡張機能は、クリアテキストパスワードなどの機密データをログから編集します。 RDS for PostgreSQL DB インスタンスが、 RDS for PostgreSQL DB インスタンスのクエリログ記録をオンにする で説明されているようにデータ操作言語 (DML) ステートメントをログに記録するように設定されている場合は、PostgreSQL Audit 拡張機能を使用することでクリアテキストパスワードの問題を回避できます。

データベースインスタンスの監査は、きわめて詳細に構成できます。すべてのデータベースとすべてのユーザーを監査できます。また、特定のデータベース、ユーザー、その他のオブジェクトのみを監査することもできます。特定のユーザーやデータベースを監査対象から明示的に除外することもできます。詳細については、「監査ログからのユーザーまたはデータベースの除外」を参照してください。

キャプチャできる詳細の量を考慮すると、pgAudit を使用する場合はストレージ消費量を監視することをお勧めします。

pgAudit 拡張モジュールは、使用可能なすべての RDS for PostgreSQL バージョン。利用可能な RDS for PostgreSQL バージョンでサポートされている pgAudit バージョンのリストについては、Amazon RDS for PostgreSQL リリースノートの「Amazon RDS for PostgreSQL の拡張バージョン」を参照してください。

トピック

- pgAudit 拡張機能のセットアップ
- データベースオブジェクトの監査
- 監査ログからのユーザーまたはデータベースの除外
- pgAudit 拡張機能のリファレンス

pgAudit 拡張機能のセットアップ

RDS for PostgreSQL DB インスタンス に pgAudit 拡張機能を設定するには、まず RDS for PostgreSQL DB インスタンスのカスタム DB パラメータグループの共有ライブラリに pgAudit を追加します。カスタム DB パラメータグループの作成については、「Amazon RDS のパラメータグループ」を参照してください。次に、pgAudit 拡張機能をインストールします。最後に、監査するデータベースとオブジェクトを指定します。このセクションの手順で、方法を示します。AWS マネジメントコンソール または AWS CLI を使用できます。

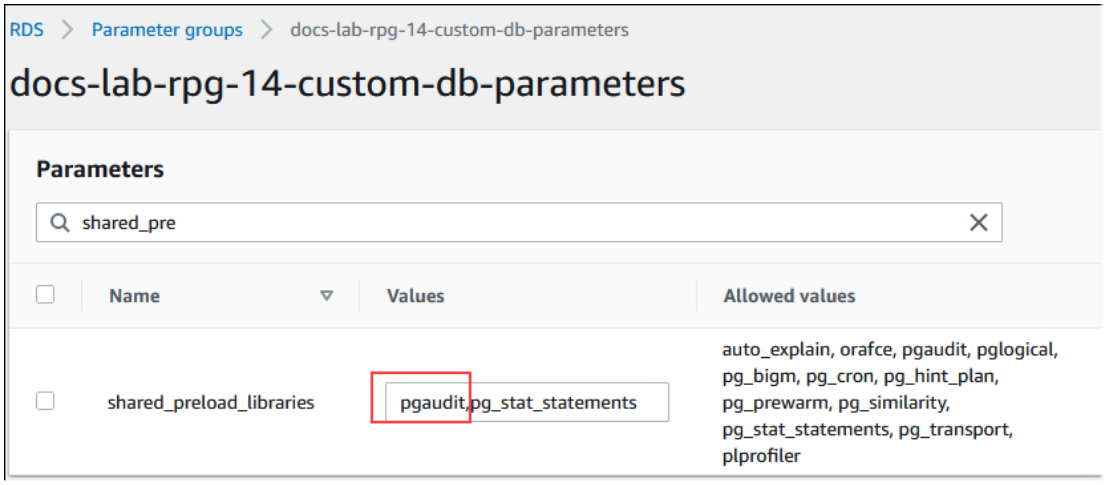
これらすべてのタスクを実行するには、`rds_superuser` ロールとして権限が必要です。

以下のステップでは、RDS for PostgreSQL DB インスタンスがカスタム DB パラメータグループに関連付けられていることを前提としています。

コンソール

pgAudit 拡張機能をセットアップするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、 RDS for PostgreSQL DB インスタンスを選択します。
3. の [Configuration] (設定) タブを開きます。RDS for PostgreSQL DB インスタンス。インスタンスの詳細の中から、パラメータグループのリンクを見つけてください。
4. リンクを選択して、に関連するカスタムパラメータを開きます。RDS for PostgreSQL DB インスタンス。
5. パラメータ検索フィールドに、`shared_pre` を入力して `shared_preload_libraries` パラメータを検索します。
6. プロパティ値にアクセスするには、[Edit parameters] (パラメータの編集) を選択します。
7. [Values] (値) フィールドのリストに `pgaudit` を追加します。値のリスト内の項目を区切るにはカンマを使用します。



8. RDS for PostgreSQL DB instance を再起動して、`shared_preload_libraries` パラメータの変更を有効にします。
9. インスタンスが使用可能になったら、pgAudit が初期化されていることを確認します。`psql` を使用して RDS for PostgreSQL DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. pgAudit を初期化すると、拡張機能を作成できるようになりました。`pgaudit` 拡張機能はデータ定義言語 (DDL) ステートメントを監査するためのイベントトリガーをインストールするため、ライブラリを初期化した後に拡張機能を作成する必要があります。

```
CREATE EXTENSION pgaudit;
```

11. `psql` セッションを終了します。

```
labdb=> \q
```

12. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
13. リストで `pgaudit.log` パラメータを検索し、ユースケースに応じた値に設定します。例えば、次の画像に示すように `pgaudit.log` パラメータを `write` に設定すると、ログへの挿入、更新、削除、およびその他のタイプの変更がキャプチャされます。

`pgaudit.log` パラメータには、次のいずれかの値を選択することもできます。

- none – これはデフォルトです。データベースの変更は記録されません。
- すべて — すべてをログに記録します (読み取り、書き込み、関数、ロール、DDL、その他)。
- ddl – `ROLE` クラスに含まれていない、すべてのデータ定義言語 (DDL) ステートメントのログ記録。
- function – 関数呼び出し、および `DO` ブロックのログ記録。
- misc – `DISCARD`、`FETCH`、`CHECKPOINT`、`VACUUM`、`SET` など、さまざまなコマンドのログ記録。
- read – `SELECT` および `COPY` のログ記録 (ソースがリレーション (テーブルなどの) またはクエリの場合)。
- role – `GRANT`、`REVOKE`、`CREATE ROLE`、`ALTER ROLE`、`DROP ROLE` など、ロールと権限に関連するステートメントのログ記録。
- write – `INSERT`、`UPDATE`、`DELETE`、`TRUNCATE`、および `COPY` のログ記録 (送信先がリレーション (テーブル) の場合)。

14. [Save changes] (変更の保存) をクリックします。
15. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
16. データベースリストから、 RDS for PostgreSQL DB インスタンスを選択します。

AWS CLI

pgAudit をセットアップするには

AWS CLI を使用して pgAudit を設定するには、次の手順に示すように、modify-db-parameter-group オペレーションを呼び出してカスタムパラメータグループの監査ログパラメータを変更します。

1. 次の AWS CLI コマンドを使用してshared_preload_librariesパラメータに pgaudit を追加します。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. 次の AWS CLI コマンドを使用して RDS for PostgreSQL DB インスタンスを再起動し、pgaudit ライブラリを初期化します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. インスタンスが使用可能になると、pgaudit が初期化されていることを確認できます。psql を使用して RDS for PostgreSQL DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

pgAudit を初期化すると、拡張機能を作成できるようになりました。

```
CREATE EXTENSION pgaudit;
```

4. AWS CLI を使用できるように psql セッションを終了します。

```
labdb=> \q
```

5. 次の AWS CLI コマンドを使用して、セッション監査ログによって記録するステートメントのクラスを指定します。この例では、pgaudit.log パラメータを write に設定し、ログへの挿入、更新、削除をキャプチャします。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
```

```
  --parameters
"ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
  --region aws-region
```

pgaudit.log パラメータには、次のいずれかの値を選択することもできます。

- none – これはデフォルトです。データベースの変更は記録されません。
- すべて — すべてをログに記録します (読み取り、書き込み、関数、ロール、DDL、その他)。
- ddl – ROLE クラスに含まれていない、すべてのデータ定義言語 (DDL) ステートメントのログ記録。
- function – 関数呼び出し、および DO ブロックのログ記録。
- misc – DISCARD、FETCH、CHECKPOINT、VACUUM、SET など、さまざまなコマンドのログ記録。
- read – SELECT および COPY のログ記録 (ソースがリレーション (テーブルなどの) またはクエリの場合)。
- role – GRANT、REVOKE、CREATE ROLE、ALTER ROLE、DROP ROLE など、ロールと権限に関連するステートメントのログ記録。
- write – INSERT、UPDATE、DELETE、TRUNCATE、および COPY のログ記録 (送信先がリレーション (テーブル) の場合)。

次の AWS CLI コマンドを使用して、 RDS for PostgreSQL DB インスタンスを再起動します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

データベースオブジェクトの監査

RDS for PostgreSQL DB インスタンスに pgAudit をセットアップし、要件に合わせて設定すると、より詳細な情報が PostgreSQL ログに取得されます。例えば、デフォルトの PostgreSQL ログ設定はデータベーステーブルに変更が加えられた日付と時刻を識別しますが、pgAudit 拡張機能では、拡張機能のパラメータの設定方法に応じて、スキーマ、変更を行ったユーザー、その他の詳細をログエントリに含めることができます。監査を設定して、次の方法で変更を追跡できます。

- セッションごとに、ユーザー別。セッションレベルでは、完全修飾コマンドテキストをキャプチャできます。

- オブジェクトごとに、ユーザー別、データベース別。

オブジェクト監査機能は、システムで rds_pgaudit ロールを作成し、そのロールをカスタムパラメータグループの pgaudit.role パラメータに追加したときに有効になります。デフォルトでは、pgaudit.role パラメータは設定されておらず、許容される値は rds_pgaudit だけです。以下の手順は、pgaudit が初期化され、pgAudit 拡張機能のセットアップ の手順に従って pgaudit 拡張機能を作成したことを前提としています。

```
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: statement: SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: AUDIT: SESSION,2,1,READ,SELECT,TABLE,public.support,"SELECT
feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;",<none>
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: QUERY STATISTICS
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:DETAIL: ! system usage stats:
! 0.009494 s user, 0.007442 s system, 0.141985 s elapsed
! [0.022327 s user, 0.007442 s system total]
```

この例に示すように、「LOG: AUDIT: SESSION」行には、テーブルとそのスキーマなどの詳細情報が表示されます。

オブジェクト監査をセットアップするには

1. psql を使用して RDS for PostgreSQL DB インスタンスに接続します。

   ```
   psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
   username=postgrespostgres --password --dbname=labdb
   ```

2. 次のコマンドを使用して、rds_pgaudit というデータベースロールを作成します。

   ```
   labdb=> CREATE ROLE rds_pgaudit;
   CREATE ROLE
   labdb=>
   ```

3. psql セッションを終了します。

   ```
   labdb=> \q
   ```

   次のステップでは、AWS CLI を使用してカスタムパラメータグループの監査ログパラメータを変更します。

4. 次の AWS CLI コマンドを使用して、pgaudit.role パラメータを rds_pgaudit に設定します。デフォルトでは、このパラメータは空で、rds_pgaudit は、唯一の許容値です。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
   --region aws-region
```

5. 次の AWS CLI コマンドを使用して RDS for PostgreSQL DB インスタンスを再起動し、パラメータの変更を有効にします。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

6. 次のコマンドを実行して、pgaudit.role が rds_pgaudit に設定されたことを確認します。

```
SHOW pgaudit.role;
pgaudit.role
-----------------
rds_pgaudit
```

pgAudit ログ記録をテストするには、監査するサンプルコマンドをいくつか実行します。例えば、次のコマンドを実行します。

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

データベースログには、次のようなエントリが含まれます。

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

ログの表示方法については、「Amazon RDS ログファイルのモニタリング」を参照してください。

pgAudit 拡張機能の詳細については、GitHub で「pgAudit」を参照してください。

監査ログからのユーザーまたはデータベースの除外

RDS for PostgreSQL データベースログファイル で説明したように、PostgreSQL ログはストレージ容量を使用します。pgAudit 拡張機能を使用すると、追跡する変更に応じて、ログに収集されるデータの量が、程度の差はありますが、増加します。内のすべてのユーザーまたはデータベースを監査する必要はないかもしれません。RDS for PostgreSQL DB インスタンス。

ストレージへの影響を最小限に抑え、監査記録を不必要にキャプチャしないようにするには、ユーザーとデータベースを監査対象から除外できます。特定のセッション内の口ギングを変更することもできます。次の例は、その方法を示しています。

> **Note**
>
> セッションレベルのパラメータ設定は、 RDS for PostgreSQL DB インスタンスのカスタム DB パラメータグループの設定よりも優先されます。データベースユーザーに監査口グ設定の設定をバイパスさせたくない場合は、必ず権限を変更してください。

RDS for PostgreSQL DB インスタンスが、すべてのユーザーとデータベースについて同じレベルのアクティビティを監査するように設定されているとします。次に、myuser ユーザーを監査しないことにします。次の SQL コマンドを使用して、myuser の監査機能を無効にできます。

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

次に、次のクエリを使用して pgaudit.log の user_specific_settings 列をチェックし、パラメータが NONE に設定されていることを確認できます。

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

次のような出力が表示されます。

```
 user_name | user_specific_settings
```

```
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

次のコマンドを使用すると、データベースとのセッションの最中に、指定したユーザーのログをオフにできます。

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

次のクエリを使用して、特定のユーザーとデータベースの組み合わせの pgaudit.log の設定列を確認します。

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

以下のような出力が表示されます。

```
  user_name | database_name |     settings
-----------+---------------+-----------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

myuser の監査を無効化した後に、mydatabase の変更を追跡しないことにしました。次のコマンドを使用して、その特定のデータベースの監査を無効化します。

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

次に、以下のクエリで database_specific_settings 列を確認し、pgaudit.log が NONE に設定されていることを確認します。

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
a.datname = 'mydatabase';
```

次のような出力が表示されます。

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

myuser の設定をデフォルト設定に戻すには、次のコマンドを使用します。

```
ALTER USER myuser RESET pgaudit.log;
```

設定をデータベースのデフォルト設定に戻すには、次のコマンドを使用します。

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

ユーザーとデータベースをデフォルト設定にリセットするには、次のコマンドを使用します。

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

また、pgaudit.log パラメータに pgaudit.log を他の許容値のいずれかに設定することで、特定のイベントをログに記録することもできます 詳細については、「pgaudit.log パラメータの許容設定のリスト」を参照してください。

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

pgAudit 拡張機能のリファレンス

このセクションにリストされている 1 つまたは複数のパラメータを変更することで、監査ログに必要な詳細レベルを指定できます。

pgAudit 動作の制御

監査ログは、次のテーブルに示す 1 つ以上のパラメータを変更することで制御できます。

パラメータ	説明
pgaudit.log	セッション監査ログ記録によってログに記録されるステートメントのクラスを指定します。許容値には、ddl、関数、その他、読み取り、ロール、書き込み、なし、すべてが含まれます。(詳しくは、「pgaudit.log パラメータの許容設定のリスト」を参照してください)。
pgaudit.log_catalog	オンにすると (1 に設定)、ステートメント内のすべてのリレーションが pg_catalog 内にある場合に、ステートメントを監査証跡に追加します。
pgaudit.log_level	ログエントリに使用されるログレベルを指定します。指定できる値は debug5、debug4、debug3、debug2、debug1、info、notice、warning、log です。
pgaudit.log_parameter	オン (1 に設定) すると、ステートメントとともに渡されたパラメータが監査ログに記録されます。
pgaudit.log_relation	オンにすると (1 に設定)、セッションの監査ログで、SELECT ステートメントまたは DML ステートメントで参照されるリレーション (TABLE、VIEW など) ごとに個別のログエントリが作成されます。
pgaudit.log_statement_once	ログ記録に、ステートメントテキストとパラメータを、ステートメントとサブステートメントの組み合わせの最初のログエントリとともに含めるか、すべてのエントリとともに含めるかを指定します。
pgaudit.role	オブジェクト監査ログ記録に使用するマスターロールを指定します。唯一許容されるエントリは rds_pgaudit です。

pgaudit.log パラメータの許容設定のリスト

値	説明
なし	これがデフォルトです。データベースの変更は記録されません。
すべて	すべてをログに記録します (読み取り、書き込み、関数、ロール、DDL、その他)。
ddl	ROLE クラスに含まれていない、すべてのデータ定義言語 (DDL) ステートメントのログ記録。
関数	関数呼び出し、および DO ブロックのログ記録。
misc	DISCARD、FETCH、CHECKPOINT 、VACUUM、SET など、さまざまなコマンドのログ記録。
読む	SELECT および COPY のログ記録 (ソースがリレーション (テーブルなどの) またはクエリの場合)。
ロール	GRANT、REVOKE、CREATE ROLE、ALTER ROLE、DROP ROLE など、ロールと権限に関連するステートメントのログ記録。
書き込み	INSERT、UPDATE、DELETE、TRUNCATE、および COPY のログ記録 (送信先がリレーションの場合)。

セッション監査で複数のイベントタイプを口グ記録するには、カンマ区切りリストを使用します。すべてのイベントタイプを口グ記録するには、pgaudit.log を ALL に設定します。DB インスタンスを再起動して、変更を適用します。

オブジェクト監査では、監査の口グ記録を絞り込み、特定のリレーションを操作できます。例えば、1 つまたは複数のテーブルで、READ オペレーションの口グ記録を監査するよう指定できます。

PostgreSQL pg_cron エクステンションによるメンテナンスのスケジューリング

PostgreSQL pg_cron エクステンションを使用すると、PostgreSQL データベース内でメンテナンスコマンドのスケジュールを組むことができます。拡張機能の詳細については、pg_cron ドキュメントの 「What is pg_cron?」 を参照してください。

pg_cron エクステンションは PostgreSQL の RDS エンジンのバージョン 12.5 以降でサポートされています。

pg_cron の使用の詳細については、「RDS for PostgreSQL または Aurora PostgreSQL 互換エディションのデータベースで pg_cron を使用してジョブをスケジュールする」を参照してください

> **Note**
>
> pg_cron 拡張機能バージョンは、pg_available_extensions ビューに 1.6 などの 2 桁のバージョンとして表示されます。1.6.4 や 1.6.5 などの 3 桁のバージョンが表示される場合がありますが、一部のコンテキストでは、拡張機能のアップグレードを実行するときに 2 桁のバージョンを指定する必要があります。

トピック

- pg_cron 拡張機能のセットアップ
- データベースユーザーに pg_cron を使用する権限を付与する
- pg_cron ジョブのスケジューリング
- pg_cron 拡張機能のリファレンス

pg_cron 拡張機能のセットアップ

次のように pg_cron 拡張機能をセットアップします。

1. shared_preload_libraries パラメータ値に pg_cron を追加して、PostgreSQL DB インスタンスに関連付けられているカスタムパラメータグループを変更します。
   - RDS for PostgreSQL DB インスタンスが rds.allowed_extensions パラメータを使用して、インストール可能な拡張機能を明示的に一覧表示するには、リストに pg_cron 拡張機能を追加する必要があります。RDS for PostgreSQL の特定のバージョンのみが、rds.allowed_extensions パラメータに対応しています。デフォルトでは使用可能なす

べての拡張機能が許可されます。詳しくは、「PostgreSQL エクステンションのインストールを制限する」を参照してください。

静的パラメータグループの変更を反映するために PostgreSQL DB インスタンスを再起動します。パラメータグループを使用する方法の詳細については、Amazon RDS の DB パラメータグループのパラメータの変更 を参照してください。

2. PostgreSQL DB インスタンスが再起動したら、rds_superuser の許可を持つアカウントを使用して以下のコマンドを実行します。例えば、RDS for PostgreSQL DB インスタンスの作成時にデフォルト設定を使用した場合は、ユーザー postgres として接続し拡張機能を作成します。

```
CREATE EXTENSION pg_cron;
```

pg_cron スケジューラは、postgres という名前のデフォルトの PostgreSQL データベースに設定されます。pg_cron オブジェクトはこの postgres データベースに作成され、すべてのスケジューリングアクションがこのデータベースで実行されます。

3. デフォルト設定を使用することも、ジョブをスケジュールして、PostgreSQL DB インスタンス内の他のデータベースで実行させることもできます。PostgreSQL DB インスタンス内の他のデータベースでジョブをスケジュールするには、デフォルトのデータベース以外のデータベースでの cron ジョブのスケジューリング の例を参照してください。

データベースユーザーに pg_cron を使用する権限を付与する

pg_cron 拡張機能をインストールするには、rds_superuser 権限が必要です。ただし、pg_cron の使用権限は (rds_superuser グループ/ロールのメンバーによって) 他のデータベースユーザーに付与して、各ユーザーが自分のジョブをスケジュールできるようにすることができます。本番環境での運用が改善される場合にのみ、cron スキーマへのアクセス許可を付与することをお勧めします。

cron スキーマでデータベースユーザー権限を付与するには、以下のコマンドを実行します。

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

これにより、アクセス権限のあるオブジェクトの cron ジョブをスケジュールするための cron スキーマへの *db-user* アクセス許可が付与されます。データベースユーザーに権限がない場合、以下に示すように、エラーメッセージを postgresql.log ファイルに投稿した後にジョブは失敗します。

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
```

```
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

つまり、cron スキーマへの権限を付与されているデータベースユーザーに、スケジュールを設定する予定のオブジェクト (テーブル、スキーマなど) に対する権限も付与されていることを確認します。

この cron ジョブの詳細と、その成功または失敗も cron.job_run_details テーブルにキャプチャされます。詳細については、「ジョブのスケジュール設定とステータス取得用のテーブル」を参照してください。

pg_cron ジョブのスケジューリング

次のセクションでは、pg_cron ジョブを使用してさまざまな管理タスクをスケジュールする方法について説明します。

> **Note**
>
> pg_cron ジョブの作成時は、max_worker_processes 設定が cron.max_running_jobs の数より大きいことを確認します。バックグラウンドのワーカープロセスを使い切ると、pg_cron ジョブは失敗します。pg_cron ジョブのデフォルト数は 5 です。詳しくは、「pg_cron 拡張機能の管理用パラメータ」を参照してください。

トピック

- テーブルのバキューム処理
- pg_cron の履歴テーブルの除去
- エラーのログを postgresql.log ファイルにのみ記録する
- デフォルトのデータベース以外のデータベースでの cron ジョブのスケジューリング

テーブルのバキューム処理

Autovacuum は、ほとんどの場合、バキュームのメンテナンスを実行します。ただし、特定のテーブルのバキューム処理を、選択した特定の時点にスケジュールしたい、というケースも考えられます。

「Amazon RDS for PostgreSQL での PostgreSQL 自動バキュームの使用」も参照してください。

以下は、cron.schedule 関数を使用して、毎日 22:00 (GMT) に特定のテーブルで VACUUM FREEZE を使用するようにジョブをセットアップする例です。

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

上記の例を実行した後、次のように cron.job_run_details テーブル内の履歴を確認できます。

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                          |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+-------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

失敗したジョブを確認するための cron.job_run_details テーブルのクエリは、次のとおりです。

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                         | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+---------------------------------+--------
+-----------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

詳細については、「ジョブのスケジュール設定とステータス取得用のテーブル」を参照してください。

pg_cron の履歴テーブルの除去

cron.job_run_details テーブルには、時間の経過とともに非常に大きくなる可能性がある cron ジョブの履歴が含まれています。そのため、このテーブルをクリアにするジョブをスケジュールする

ことをお勧めします。例えば、トラブルシューティングの目的では、1 週間分のエントリを保持するだけで十分です。

次の例では、cron.schedule 関数を使用して、cron.job_run_details テーブルをクリアにするよう、毎日午前 0 時に実行されるジョブをスケジュールします。このジョブは過去 7 日間しか残せません。rds_superuser アカウントを使用して、以下のようなジョブをスケジュールできます。

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

(詳しくは、「ジョブのスケジュール設定とステータス取得用のテーブル」を参照してください。)

エラーのログを postgresql.log ファイルにのみ記録する

cron.job_run_details テーブルへの書き込みをしないようにするには、PostgreSQL DB インスタンスに関連付けられているパラメータグループを変更し、cron.log_run パラメータをオフに設定します。pg_cron 拡張機能によって対象のテーブルには書き込まなくなり、エラーは postgresql.log ファイルのみに記録されるようになります。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

cron.log_run パラメータの値を確認するには、次のコマンドを使用します。

```
postgres=> SHOW cron.log_run;
```

詳細については、「pg_cron 拡張機能の管理用パラメータ」を参照してください。

デフォルトのデータベース以外のデータベースでの cron ジョブのスケジューリング

pg_cron のメタデータはすべて、postgres という名前の PostgreSQL のデフォルトのデータベースに保持されます。メンテナンスの cron ジョブの実行にはバックグラウンドワーカーが使用されるため、PostgreSQL DB インスタンス内の任意のデータベースでジョブのスケジューリングが可能です。

> **Note**
>
> rds_superuser ロールまたは rds_superuser 権限を持つユーザーのみが、データベース内のすべての cron ジョブを一覧表示できます。他のユーザーは、自分のジョブのみを cron.job テーブルに表示できます。

1. cron データベースで、cron.schedule を使用して通常どおりにジョブをスケジュールします。

```
postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
 freeze test_table');
```

2. 作成したジョブのデータベース列を、rds_superuser ロールを持つユーザーとして更新し、そのジョブを PostgreSQL DB インスタンス内の別のデータベースで実行できるようにします。

```
postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
```

3. cron.job テーブルのクエリを実行して確認します。

```
postgres=> SELECT * FROM cron.job;
jobid | schedule    | command                          | nodename  | nodeport |
 database | username  | active | jobname
------+-------------+----------------------------------+-----------+----------
+----------+-----------+--------+------------------------
106   | 29 03 * * * | vacuum freeze test_table       | localhost | 8192     |
 database1| adminuser | t      | database1 manual vacuum
  1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
 postgres | adminuser | t      | manual vacuum
(2 rows)
```

> **Note**
>
> 状況によっては、別のデータベースで実行する cron ジョブを追加することがあります。このような場合、ジョブは、正しいデータベース列を更新する前に、デフォルトのデータベース (postgres) で実行しようとする可能性があります。ユーザー名に権限がある場合、デフォルトのデータベースでジョブが正常に実行されます。

pg_cron 拡張機能のリファレンス

pg_cron エクステンションでは、次のパラメータ、関数、およびテーブルを使用できます。詳細については、pg_cron ドキュメントの What is pg_cron? を参照してください。

トピック

- pg_cron 拡張機能の管理用パラメータ
- 関数リファレンス: cron.schedule

- 関数リファレンス: cron.unschedule
- ジョブのスケジュール設定とステータス取得用のテーブル

pg_cron 拡張機能の管理用パラメータ

pg_cron エクステンションの動作を制御するパラメータの一覧を次に示します。

Parameter	説明
cron.database_name	pg_cron メタデータが保持されるデータベース。
cron.host	PostgresSQL に接続するためのホスト名。この値は変更できません。
cron.log_run	job_run_details テーブルで実行されるすべてのジョブを口グに記録します。有効な値は on または off です。詳細については、「ジョブのスケジュール設定とステータス取得用のテーブル」を参照してください。
cron.log_statement	実行する前に、すべての cron ステートメントを記録します。有効な値は on または off です。
cron.max_running_jobs	同時に実行できるジョブの最大数。
cron.use_background_workers	クライアントセッションの代わりにバックグラウンドワーカーを使用します。この値は変更できません。

次の SQL コマンドを使用して、これらのパラメータとその値を表示します。

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

関数リファレンス: cron.schedule

この関数は、cron ジョブをスケジュールします。このジョブは、デフォルトの `postgres` データベースで初期にスケジュールされます。この関数は、ジョブ識別子を表す `bigint` の値を返します。PostgreSQL DB インスタンス内の他のデータベースで実行するようにジョブをスケジュールするには、デフォルトのデータベース以外のデータベースでの cron ジョブのスケジューリング の例を参照してください。

この関数には、2 つの構文形式があります。

構文

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

パラメータ

Parameter	説明
`job_name`	cron ジョブの名前。
`schedule`	cron ジョブのスケジュールを示すテキスト。形式はスタンダードの cron 形式です。
`command`	実行するコマンドのテキスト。

例

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)
```

```
postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

関数リファレンス: cron.unschedule

この関数は、cron ジョブを削除します。job_name または job_id を指定できます。ポリシーにより、ユーザーがジョブのスケジュールを削除する所有者であることが確認されます。この関数は、成功または失敗を示すブール値を返します。

関数の構文形式は以下のとおりです。

構文

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

パラメータ

Parameter	説明
job_id	cron ジョブがスケジュールされたときに cron.schedule 関数から返されたジョブ識別子。
job_name	cron.schedule 関数でスケジュールされた cron ジョブの名前。

例

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)
```

```
postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

ジョブのスケジュール設定とステータス取得用のテーブル

以下のテーブルは、cron ジョブのスケジューリングのためと、そのジョブがどのように完了したかを記録するために使用されます。

表	説明
`cron.job`	スケジュールされた各ジョブに関するメタデータが含まれます。このテーブルとのほとんどのやり取りは、`cron.schedule` 関数および `cron.unschedule` 関数を使用して行う必要があります。 **Important** 更新または挿入の権限をこのテーブルに直接与えないようにお勧めします。そうすることで、ユーザーは username 列を更新し、`rds-superuser` として実行できるようになります。
`cron.job_run_details`	ここには、過去にスケジュールされ実行されたジョブに関する履歴の情報が含まれます。これは、実行されたジョブのステータス、返されたメッセージ、およびスタートと終了の時間を調査する場合に便利です。 **Note** このテーブルが無期限に増えないようにするには、定期的に削除してください。例については、「pg_cron の履歴テーブルの除去」を参照してください。

pglogical を使用してインスタンス間でデータを同期する

現在利用可能なすべての RDS for PostgreSQL バージョンは、pglogical 拡張機能をサポートしています。pglogical 拡張は、バージョン 10 で PostgreSQL により導入された機能的に類似した論理レプリケーション機能よりも前のものです。詳細については、「Amazon RDS for PostgreSQL の論理レプリケーションの実行」を参照してください。

pglogical 拡張が、2 つ以上の 間の論理レプリケーションをサポートします。RDS for PostgreSQL DB インスタンス。また、異なる PostgreSQL バージョン間のレプリケーション、および RDS for PostgreSQL DB と Aurora PostgreSQL DB クラスターで実行されているデータベース間のレプリケーションもサポートしています。pglogical 拡張は、公開/サブスクライブモデルを使用して、テーブルやその他のオブジェクト (シーケンスなど) への変更をパブリッシャーからサブスクライバーに複製します。パブリッシャーノードからサブスクライバーノードに変更が確実に同期されるようにするには、レプリケーションスロットを使用し、次のように定義されます。

- パブリッシャーノードは、他のノードにレプリケートされるデータのソースである RDS for PostgreSQL DB インスタンスです。パブリッシャーノードは、パブリケーションセットでレプリケートするテーブルを定義します。
- サブスクライバーノードは、公開者から WAL の更新を受け取る RDS for PostgreSQL DB インスタンスです。サブスクライバーは、パブリッシャーに接続してデコードされた WAL データを取得するためのサブスクリプションを作成します。サブスクライバーがサブスクリプションを作成すると、パブリッシャーノードに複製スロットが作成されます。

pglogical 拡張の設定についての情報は、以下を参照してください。

トピック

- pglogical 拡張の要件と制限
- pglogical 拡張のセットアップ
- RDS for PostgreSQL DB インスタンスに論理レプリケーションを設定する
- メジャーアップグレード後の論理レプリケーションの再確立
- RDS for PostgreSQL 用ロジカルレプリケーションスロットの管理
- pglogical 拡張のパラメータリファレンス

pglogical 拡張の要件と制限

RDS for PostgreSQL の現在利用可能なすべてのリリースが pglogical 拡張機能をサポートしています。

パブリッシャーノードとサブスクライバーノードの両方を論理レプリケーション用に設定する必要があります。

サブスクライバーからパブリッシャーにレプリケートするテーブルは、同じ名前と同じスキーマである必要があります。これらのテーブルにも同じ列が含まれている必要があり、列は同じデータ型を使用する必要があります。パブリッシャーテーブルとサブスクライバーテーブルの両方に同じプライマリキーが必要です。一意の制約事項としては PRIMARY KEY のみを使用することをお勧めします。

サブスクライバーノードのテーブルには、CHECK 制約と NOT NULL 制約について、パブリッシャーノードのテーブルよりも許可度が高い制約を設定できます。

pglogical 拡張は、PostgreSQL (バージョン 10 以降) に組み込まれている論理レプリケーション機能ではサポートされていない双方向レプリケーションなどの機能を提供します。詳細については、「PostgreSQL bi-directional replication using pglogical」(pglogical を使用した PostgreSQL の双方向レプリケーション) を参照してください。

pglogical 拡張のセットアップ

RDS for PostgreSQL DB インスタンスに pglogical 拡張機能を設定するには、RDS for PostgreSQL DB インスタンスのカスタム DB パラメータグループの共有ライブラリに pglogical を追加します。また、論理デコードをオンにするには、rds.logical_replication パラメータの値を 1 に設定する必要があります。最後に、データベースに拡張を作成します。これらのタスクには、AWS マネジメントコンソール または AWS CLI を使用できます。

これらのタスクを実行するには、rds_superuser ロールとしてアクセス許可が必要です。

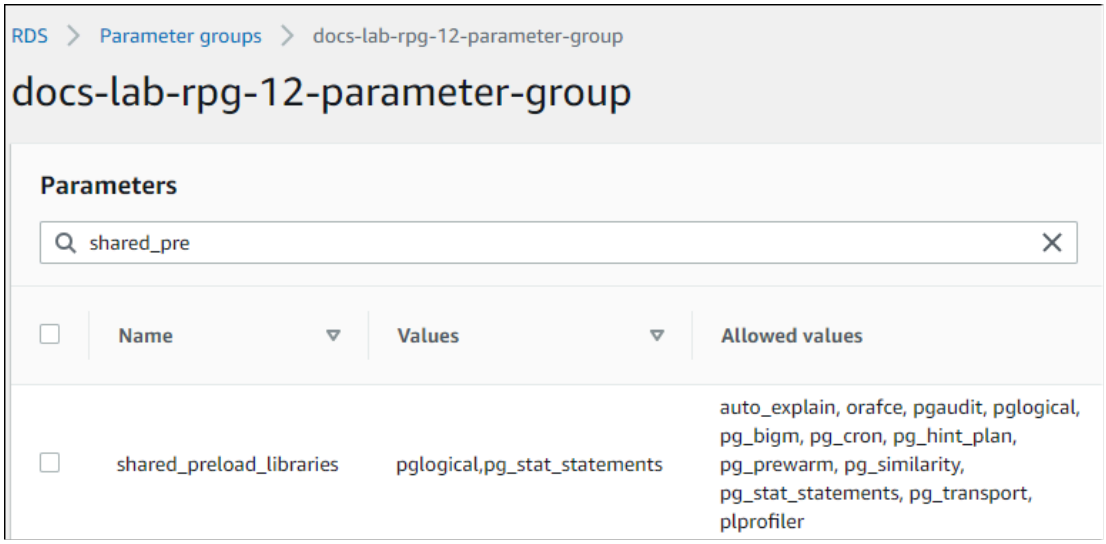
以下のステップでは、 RDS for PostgreSQL DB インスタンスがカスタム DB パラメータグループに関連付けられていることを前提としています。カスタム DB パラメータグループの作成については、「Amazon RDS のパラメータグループ」を参照してください。

コンソール

pglogical 拡張をセットアップするには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、 RDS for PostgreSQL DB インスタンスを選択します。
3. の [Configuration] (設定) タブを開きます。RDS for PostgreSQL DB インスタンス。インスタンスの詳細の中から、パラメータグループのリンクを見つけてください。
4. リンクを選択して、に関連するカスタムパラメータを開きます。RDS for PostgreSQL DB インスタンス。
5. パラメータ検索フィールドに、`shared_pre` を入力して `shared_preload_libraries` パラメータを検索します。
6. プロパティ値にアクセスするには、[Edit parameters] (パラメータの編集) を選択します。
7. [Values] (値) フィールドのリストに `pglogical` を追加します。値のリスト内の項目を区切るにはカンマを使用します。



8. `rds.logical_replication` パラメータを見つけて `1` に設定し、論理レプリケーションをオンにします。
9. RDS for PostgreSQL DB インスタンス を再起動して、変更を有効にします。
10. インスタンスが使用可能になったら、`psql` (または pgAdmin) を使用して RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. pglogical が初期化されていることを確認するには、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
```

```
rdsutils,pglogical
(1 row)
```

12. 次のように、論理デコードを有効にする設定を確認します。

```
SHOW wal_level;
wal_level
-----------
 logical
(1 row)
```

13. 次のように拡張を作成します。

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. [Save changes] (変更の保存) をクリックします。
15. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
16. データベースリストから RDS for PostgreSQL DB インスタンス を選択して選択し、アクションメニューから [Reboot] (再起動) を選択します。

AWS CLI

pglogical 拡張のセットアップするには

AWS CLI を使用して pglogical を設定するには、次の手順に示すように、modify-db-parameter-group オペレーションを呼び出してカスタムパラメータグループの特定のパラメータを変更します。

1. 次の AWS CLI コマンドを使用してshared_preload_librariesパラメータに pglogical を追加します。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. 次の AWS CLI コマンドを使用して rds.logical_replication を 1 に設定し、の論理デコード機能をオンにします。RDS for PostgreSQL DB インスタンス。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. 次の AWS CLI コマンドを使用して RDS for PostgreSQL DB インスタンスを再起動し、pglogical ライブラリを初期化します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

4. インスタンスが使用可能になったら、psql を使用して RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. 次のように拡張を作成します。

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. 次の AWS CLI コマンドを使用して、 RDS for PostgreSQL DB インスタンスを再起動します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

RDS for PostgreSQL DB インスタンスに論理レプリケーションを設定する

以下の手順では、2 つの RDS for PostgreSQL DB インスタンス間で論理レプリケーションを開始する方法を示しています。これらのステップでは、ソース (パブリッシャー) とターゲット (サブスクライバー) の両方に、pglogical 拡張のセットアップ で説明されているように pglogical 拡張が設定されていることを前提としています。

> **Note**
>
> サブスクライバーノードの node_name は、rds で開始することができません。

パブリッシャーノードを作成し、複製するテーブルを定義するには

これらのステップは、別のノードに複製する 1 つ以上のテーブルがあるデータベースが RDS for PostgreSQL インスタンスにあることを前提としています。サブスクライバー上のパブリッシャーからテーブル構造を再作成する必要があるため、まず必要に応じてテーブル構造を取得します。そのためには、psql メタコマンド \d *tablename* を使用してサブスクライバーインスタンスに同じテーブルを作成します。次の手順では、デモンストレーションを目的として、パブリッシャー (ソース) でサンプルテーブルを作成します。

1. psql を使用して、サブスクライバーのソースとして使用したいテーブルがあるインスタンスに接続します。

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   複製する既存のテーブルがない場合は、次のようにサンプルテーブルを作成できます。

   a. 次の SQL ステートメントを使用してサンプルテーブルを作成します。

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. 次の SQL ステートメントを使用して、生成されたデータをテーブルに入力します。

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

   c. 次の SQL ステートメントを使用して、テーブルにデータが存在することを確認します。

      ```
      SELECT count(*) FROM docs_lab_table;
      ```

2. 次のように、この RDS for PostgreSQL DB インスタンスをパブリッシャーノードとして指定します。

   ```
   SELECT pglogical.create_node(
       node_name := 'docs_lab_provider',
   ```

```
    dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 dbname=labdb');
 create_node
-------------
   3410995529
(1 row)
```

3. 複製するテーブルをデフォルトのレプリケーションセットに追加します。レプリケーションセットの詳細については、pglogical ドキュメントの「Replication sets」(レプリケーションセット)を参照してください。

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  --------------------------
  t
  (1 row)
```

パブリッシャーノードの設定が完了しました。これで、パブリッシャーから更新を受け取るようにサブスクライバーノードを設定できます。

サブスクライバーノードを設定し、更新を受信するサブスクリプションを作成するには

これらのステップは、RDS for PostgreSQL DB インスタンスが pglogical 拡張機能を使用してセットアップされていることを前提としています。詳細については、「pglogical 拡張のセットアップ」を参照してください。

1. psql を使用して、パブリッシャーから更新を受け取るインスタンスに接続します。

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. サブスクライバーの RDS for PostgreSQL DB インスタンスで、パブリッシャーに存在するのと同じテーブルを作成します。この例では、テーブルは docs_lab_table です。次に示すようにテーブルを作成できます。

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. このテーブルが空であることを確認します。

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. 次のように、この RDS for PostgreSQL DB インスタンスをサブスクライバーノードとして指定します。

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. サブスクリプションを作成します。

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

このステップを完了すると、パブリッシャーのテーブルのデータが、サブスクライバーのテーブルに作成されます。このことを確認するには、次の SQL クエリを使用します。

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

これ以降、パブリッシャーのテーブルに加えられた変更は、サブスクライバーのテーブルにレプリケートされます。

メジャーアップグレード後の論理レプリケーションの再確立

論理レプリケーションのパブリッシャーノードとして設定されている RDS for PostgreSQL DB インスタンスのメジャーバージョンアップグレードを実行する前に、アクティブではないものを含め、すべてのレプリケーションスロットを削除する必要があります。パブリッシャーノードからデータベーストランザクションを一時的に迂回させ、レプリケーションスロットを削除し、RDS for PostgreSQL DB インスタンスをアップグレードしてから、レプリケーションを再確立して再開することをお勧めします。

レプリケーションスロットはパブリッシャーノードでのみホストされます。論理レプリケーションシナリオの RDS for PostgreSQL サブスクライバーノードには削除するスロットはありませんが、パブリッシャーへのサブスクリプションを持つサブスクライバーノードとして指定されている間は、メジャーバージョンにアップグレードできません。RDS for PostgreSQL サブスクライバーノードをアップグレードする前に、サブスクリプションとノードを削除してください。詳細については、「RDS for PostgreSQL 用ロジカルレプリケーションスロットの管理」を参照してください。

論理レプリケーションが中断されたことの確認

次のように、パブリッシャーノードまたはサブスクライバーノードのいずれかにクエリを実行することで、レプリケーションプロセスが中断されたことを確認できます。

パブリッシャーノードを確認するには

- psql を使用してパブリッシャーノードに接続して、pg_replication_slots 関数をクエリします。active 列の値に注目します。通常は t (true) が返されます。これは、レプリケーションがアクティブであることを示します。クエリが f (false) を返す場合は、サブスクライバーへのレプリケーションが停止したことを示します。

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-----------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

サブスクライバーノードを確認するには

サブスクライバーノードでは、3 つの異なる方法でレプリケーションのステータスを確認できます。

- サブスクライバーノードの PostgreSQL ログを調べて、失敗のメッセージを見つけます。ログでは、次に示すように、終了コード 1 を含むメッセージで失敗が識別されます。

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- pg_replication_origin 関数をクエリします。次のように、psql を使用してサブスクライバーノード上のデータベースに接続し、pg_replication_origin 関数をクエリします。

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  結果セットが空の場合は、レプリケーションが中断されたことを意味します。通常、次のような出力が表示されます。

```
   roident |                       roname
  ---------+-------------------------------------------------
         1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
  (1 row)
```

- 次の例に示すように、pglogical.show_subscription_status 関数をクエリします。

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
---====----------------+--------+-------------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

  この出力は、レプリケーションが中断されたことを示しています。そのステータスは down です。通常、出力にはステータスが replicating として表示されます。

論理レプリケーションプロセスが中断された場合は、次のステップに従ってレプリケーションを再確立できます。

パブリッシャーノードとサブスクライバーノード間の論理レプリケーションを再確立するには

レプリケーションを再確立するには、以下のステップで説明するように、まずサブスクライバーをパブリッシャーノードから切断し、次にサブスクリプションを再確立します。

1. 次のように psql を使用してサブスクライバーノードに接続します。

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. pglogical.alter_subscription_disable 関数を使用してサブスクリプションを非アクティブ化します。

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. 以下のように、pg_replication_origin をクエリして、パブリッシャーノードの識別子を取得します。

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+--------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. 前のステップからの応答を pg_replication_origin_create コマンドに使用して、サブスクリプションが再確立されたときに使用できる識別子を割り当てます。

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. 次の例のように、ステータスを true にして名前を渡し、サブスクリプションを有効にします。

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
```

```
 t
(1 row)
```

ノードのステータスを確認します。ステータスはこの例のように replicating として表示されているはずです。

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+------------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

パブリッシャーノード上のサブスクライバーのレプリケーションスロットのステータスを確認します。スロットの active 列は t (true) を返し、レプリケーションが再確立されたことを示します。

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                   slot_name              |      plugin      | slot_type | active
------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

RDS for PostgreSQL 用ロジカルレプリケーションスロットの管理

論理レプリケーションシナリオでパブリッシャーノードとして機能している RDS for PostgreSQL DB インスタンスでメジャーバージョンアップグレードを実行する前に、インスタンスのレプリケーションスロットを削除する必要があります。メジャーバージョンアップグレードの事前確認プロセスにより、スロットが削除されるまでアップグレードを続行できないことが通知されます。

RDS for PostgreSQL DB インスタンスからスロットを削除するには、まずサブスクリプションを削除してからスロットを削除します。

pglogical 拡張を使用して作成されたレプリケーションスロットを特定するには、各データベースにログインしてノードの名前を取得します。サブスクライバーノードにクエリを実行すると、次の例に示すように、パブリッシャーノードとサブスクライバーノードの両方が出力されます。

```
SELECT * FROM pglogical.node;
node_id   |     node_name
```

```
------------+-------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

次のクエリで、サブスクリプションの詳細を取得できます。

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

これで、次のようにサブスクリプションを削除できます。

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

サブスクリプションを削除すると、ノードを削除できます。

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
-----------
 t
(1 row)
```

次のように、ノードが存在しないことを確認できます。

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

pglogical 拡張のパラメータリファレンス

表には、pglogical 拡張に関連するパラメータがあります。pglogical.conflict_log_level や pglogical.conflict_resolution などのパラメータは、更新の競合を処理するために使用さ

れます。パブリッシャーから変更をサブスクライブしているテーブルにローカルで変更を加えると、競合が発生する可能性があります。これ以外にも、競合は、双方向のレプリケーションや、複数のサブスクライバーが同じパブリッシャーからレプリケートする場合など、さまざまなシナリオで発生する可能性があります。詳細については、「PostgreSQL bi-directional replication using pglogical」(pglogical を使用した PostgreSQL の双方向レプリケーション) を参照してください。

パラメータ	説明
pglogical.batch_inserts	可能であれば、バッチ挿入。デフォルトでは設定されていません。オンにする場合は「1」に、オフにする場合は「0」に変更します。
pglogical.conflict_log_level	解決された競合のログ記録に使用するログレベルを設定します。サポートされている文字列値は、debug5、debug4、debug3、debug2、debug1、info、notice、warning、error、log、fatal、panic です。
pglogical.conflict_resolution	競合が解決可能な場合に競合を解決するために使用するメソッドを設定します。サポートされている文字列値は、error、apply_remote、keep_local、last_update_wins、first_update_wins です。
pglogical.extra_connection_options	すべてのピアノード接続に追加する接続オプション。
pglogical.synchronous_commit	pglogical 固有の同期コミット値
pglogical.use_spi	低レベル API の代わりに SPI (サーバープログラミングインターフェイス) を使用して変更を適用します。オンにする場合は「1」に、オフにする場合は「0」に設定します。SPI の詳細については、PostgreSQL ドキュメントの「サーバープログラミングインターフェイス」を参照してください。

pgactive を使用したアクティブ/アクティブレプリケーションのサポート

pgactive 拡張は、アクティブ/アクティブレプリケーションを使用して、複数の RDS for PostgreSQL データベースに対する書き込み操作をサポートおよび調整します。Amazon RDS for PostgreSQL は、次のバージョンの pgactive 拡張機能をサポートしています。

- RDS for PostgreSQL バージョン 17.0 以降のバージョン
- RDS for PostgreSQL 16.1 またはそれ以降の 16 バージョン
- RDS for PostgreSQL 15.4-R2 以降のバージョン 15
- RDS for PostgreSQL 14.10 以降のバージョン 14
- RDS for PostgreSQL 13.13 以降のバージョン 13
- RDS for PostgreSQL 12.17 以降のバージョン 12
- RDS for PostgreSQL 11.22

> **Note**
>
> レプリケーション設定に複数のデータベースに対する書き込み操作があると、競合が発生する可能性があります。詳細については、アクティブ/アクティブレプリケーションの競合の処理を参照してください。

トピック

- pgactive 拡張の制限事項
- pgactive 拡張機能の初期化
- RDS for PostgreSQL DB インスタンスのアクティブ/アクティブレプリケーションの設定
- pgactive メンバー間のレプリケーションラグの測定
- pgactive 拡張機能のパラメータ設定
- アクティブ/アクティブ競合について
- pgactive スキーマについて
- pgactive 関数リファレンス
- アクティブ/アクティブレプリケーションの競合の処理
- アクティブ/アクティブレプリケーションでのシーケンスの処理

pgactive 拡張の制限事項

- すべてのテーブルには主キーが必要です。主キーがないと、更新や削除は許可されません。主キー列の値は更新しないでください。
- シーケンスにはギャップがある場合があり、順序に従わないこともあります。シーケンスはレプリケートされません。詳細については、「アクティブ/アクティブレプリケーションでのシーケンスの処理」を参照してください。
- DDL とラージオブジェクトはレプリケートされません。
- セカンダリの一意のインデックスはデータの相違を引き起こす可能性があります。
- 照合順序はグループ内のすべてのノードで同一である必要があります。
- ノード間の負荷分散はアンチパターンです。
- トランザクションが大きいと、レプリケーションの遅延が発生する可能性があります。

pgactive 拡張機能の初期化

RDS for PostgreSQL DB インスタンスの `pgactive` 拡張機能を初期化するには、`rds.enable_pgactive` パラメータの値を `1` に設定し、データベースに拡張を作成します。これを行うと、`rds.logical_replication` パラメータと `track_commit_timestamp` パラメータが自動的に有効になり、`wal_level` の値が `logical` に設定されます。

これらのタスクを実行するには、`rds_superuser` ロールとしてアクセス許可が必要です。

AWS マネジメントコンソール または AWS CLI を使用して、必要な RDS for PostgreSQL DB インスタンスを作成できます。以下のステップでは、RDS for PostgreSQL DB インスタンスがカスタム DB パラメータグループに関連付けられていることを前提としています。カスタム DB パラメータグループの作成については、「Amazon RDS のパラメータグループ」を参照してください。

コンソール

pgactive 拡張機能を初期化するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、RDS for PostgreSQL DB インスタンスを選択します。
3. RDS for PostgreSQL DB インスタンスの [設定] タブを開きます。インスタンスの詳細で、[DB インスタンスパラメータグループ] リンクを見つけます。

4. リンクを選択して、RDS for PostgreSQL DB インスタンスに関連付けられたカスタムパラメータを開きます。
5. `rds.enable_pgactive` パラメータを見つけて 1 に設定し、`pgactive` 機能を初期化します。
6. [Save changes] (変更の保存) をクリックします。
7. Amazon RDS コンソールのナビゲーションペインで、[データベース] を選択します。
8. RDS for PostgreSQL DB インスタンスを選択し、[アクション] メニューから [再起動] を選択します。
9. DB インスタンスの再起動を確定して、変更を有効にします。
10. DB インスタンスが使用可能になったら、`psql` または他の任意の PostgreSQL インスタンスを使用して RDS for PostgreSQL DB インスタンスに接続します。

    次の例では、RDS for PostgreSQL DB インスタンスに *`postgres`* という名前のデフォルトデータベースがあることを前提としています。

    ```
    psql --host=mydb.111122223333.aws-region.rds.amazonaws.com --port=5432 --
    username=postgres --password=PASSWORD --dbname=postgres
    ```

11. pgactive が初期化されていることを確認するには、次のコマンドを実行します。

    ```
    postgres=>SELECT setting ~ 'pgactive'
    FROM pg_catalog.pg_settings
    WHERE name = 'shared_preload_libraries';
    ```

    `pgactive` が `shared_preload_libraries` にある場合、前述のコマンドは以下を返します。

    ```
    ?column?
    ----------
     t
    ```

AWS CLI

pgactive 拡張機能を初期化するには

AWS CLI を使用して pgactive を設定するには、次の手順に示すように、modify-db-parameter-group オペレーションを呼び出してカスタムパラメータグループ内の特定のパラメータを変更します。

1. AWS CLI コマンドを使用して rds.enable_pgactive を 1 に設定し、RDS for PostgreSQL DB インスタンスの pgactive 機能を初期化します。

```
postgres=>aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.enable_pgactive,ParameterValue=1,ApplyMethod=pending-reboot" \
   --region aws-region
```

2. 次の AWS CLI コマンドを使用して RDS for PostgreSQL DB インスタンスを再起動し、pgactive ライブラリを初期化します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. インスタンスが使用可能になったら、psql を使用して RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=mydb.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=master user --password=PASSWORD --dbname=postgres
```

4. pgactive が初期化されていることを確認するには、次のコマンドを実行します。

```
postgres=>SELECT setting ~ 'pgactive'
FROM pg_catalog.pg_settings
WHERE name = 'shared_preload_libraries';
```

pgactive が shared_preload_libraries にある場合、前述のコマンドは以下を返します。

```
?column?
```

```
----------
 t
```

RDS for PostgreSQL DB インスタンスのアクティブ/アクティブレプリケーションの設定

以下の手順は、pgactive が利用可能な場合に 2 つの RDS for PostgreSQL DB インスタンス間でアクティブ/アクティブレプリケーションを開始する方法を示しています。マルチリージョンの高可用性の例を実行するには、2 つの異なるリージョンに Amazon RDS for PostgreSQL インスタンスをデプロイし、VPC ピアリングを設定する必要があります。詳細については、「VPC ピアリング接続」を参照してください。

> **Note**
>
> 複数のリージョン間でトラフィックを送信すると、追加コストが発生する可能性があります。

次の手順では、RDS for PostgreSQL DB インスタンスが pgactive 拡張を使用して有効化されていることを前提としています。詳細については、「pgactive 拡張機能の初期化」を参照してください。

pgactive 拡張を使用して最初の RDS for PostgreSQL DB インスタンスを設定するには

次の例は、pgactive グループの作成方法と、RDS for PostgreSQL DB インスタンスで pgactive 拡張を作成するために必要なその他の手順を示しています。

1. psql または別のクライアントツールを使用して、最初の RDS for PostgreSQL DB インスタンスに接続します。

   ```
   psql --host=firstinstance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password=PASSWORD --dbname=postgres
   ```

2. 次のコマンドを使用して RDS for PostgreSQL インスタンスにデータベースを作成します。

   ```
   postgres=> CREATE DATABASE app;
   ```

3. 次のコマンドを使用して、接続先を新しいデータベースに切り替えます。

   ```
   \c app
   ```

4. 次の SQL ステートメントを使用して、サンプルのテーブルを作成および設定します。

   a. 次の SQL ステートメントを使用してサンプルテーブルを作成します。

```
app=> CREATE SCHEMA inventory;
CREATE TABLE inventory.products (
id int PRIMARY KEY, product_name text NOT NULL,
created_at timestamptz NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

   b. 次の SQL ステートメントを使用して、サンプルデータをテーブルに入力します。

```
app=> INSERT INTO inventory.products (id, product_name)
VALUES (1, 'soap'), (2, 'shampoo'), (3, 'conditioner');
```

   c. 次の SQL ステートメントを使用して、テーブルにデータが存在することを確認します。

```
 app=>SELECT count(*) FROM inventory.products;

 count
-------
 3
```

5. 既存のデータベースで pgactive 拡張を作成します。

```
app=> CREATE EXTENSION pgactive;
```

6. pgactive グループを安全に作成して初期化するには、以下のコマンドを使用します。

```
app=>
-- connection info for endpoint1
CREATE SERVER pgactive_server_endpoint1
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint1>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint1
    OPTIONS (user 'postgres', password '<password>');
      -- connection info for endpoint2
CREATE SERVER pgactive_server_endpoint2
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint2>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint2
    OPTIONS (user 'postgres', password '<password>');
```

これで、レプリケーショングループを初期化し、この最初のインスタンスを追加できます。

```
SELECT pgactive.pgactive_create_group(
    node_name := 'endpoint1-app',
    node_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint1'

);
```

代替手段として、以下のコマンドを使用して pgactive グループを作成および初期化します。ただし、安全性は低くなります。

```
app=> SELECT pgactive.pgactive_create_group(
    node_name := 'node1-app',
    node_dsn := 'dbname=app host=firstinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD');
```

node1-app は、pgactive グループ内のノードを一意に識別するために割り当てる名前です。

> **Note**
>
> パブリックにアクセス可能な DB インスタンスで、このステップを正常に実行するには、rds.custom_dns_resolution パラメータを 1 に設定して有効にする必要があります。

7. DB インスタンスの準備が整っているかどうかを確認するには、次のコマンドを使用します。

```
app=> SELECT pgactive.pgactive_wait_for_node_ready();
```

コマンドが正常に完了した場合は、次の出力が表示されます。

```
pgactive_wait_for_node_ready
-----------------------------
(1 row)
```

2 番目の RDS for PostgreSQL インスタンスを設定して **pgactive** グループに参加させるには

次の例は、RDS for PostgreSQL DB インスタンスを pgactive グループに参加させる方法と、DB インスタンスに pgactive 拡張を作成するために必要なその他のステップを示しています。

次の手順では、RDS for PostgreSQL DB インスタンスが pgactive 拡張を使用して設定されていることを前提としています。詳細については、「pgactive 拡張機能の初期化」を参照してください。

1. psql を使用して、パブリッシャーから更新を受け取るインスタンスに接続します。

```
psql --host=secondinstance.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password=PASSWORD --dbname=postgres
```

2. 次のコマンドを使用して、2 番目の RDS for PostgreSQL DB インスタンスにデータベースを作成します。

```
postgres=> CREATE DATABASE app;
```

3. 次のコマンドを使用して、接続先を新しいデータベースに切り替えます。

```
\c app
```

4. 既存のデータベースに pgactive 拡張を作成します。

```
app=> CREATE EXTENSION pgactive;
```

5. 次に示すように、以下のコマンドを使用して、より安全な方法で RDS for PostgreSQL の 2 番目の DB インスタンスを pgactive グループに参加させます。

```
-- connection info for endpoint1
CREATE SERVER pgactive_server_endpoint1
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint1>', dbname 'app');
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint1
    OPTIONS (user 'postgres', password '<password>');

-- connection info for endpoint2
CREATE SERVER pgactive_server_endpoint2
    FOREIGN DATA WRAPPER pgactive_fdw
    OPTIONS (host '<endpoint2>', dbname 'app');
```

```
CREATE USER MAPPING FOR postgres
    SERVER pgactive_server_endpoint2
    OPTIONS (user 'postgres', password '<password>');
```

```
SELECT pgactive.pgactive_join_group(
    node_name := 'endpoint2-app',
    node_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint2',
    join_using_dsn := 'user_mapping=postgres
 pgactive_foreign_server=pgactive_server_endpoint1'
);
```

代替手段として、以下のコマンドを使用して RDS for PostgreSQL の 2 番目の DB インスタンスを pgactive グループに参加させます。ただし、安全性は低くなります。

```
app=> SELECT pgactive.pgactive_join_group(
node_name := 'node2-app',
node_dsn := 'dbname=app host=secondinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD',
join_using_dsn := 'dbname=app host=firstinstance.111122223333.aws-
region.rds.amazonaws.com user=postgres password=PASSWORD');
```

node2-app は、pgactive グループ内のノードを一意に識別するために割り当てる名前です。

6. DB インスタンスの準備が整っているかどうかを確認するには、次のコマンドを使用します。

```
app=> SELECT pgactive.pgactive_wait_for_node_ready();
```

コマンドが正常に完了すると、次の出力が表示されます。

```
pgactive_wait_for_node_ready
-----------------------------
(1 row)
```

最初の RDS for PostgreSQL データベースが比較的大きい場合は、pgactive.pgactive_wait_for_node_ready() から復元操作の進行状況レポートを出力されることを確認できます。出力は次の例のようになります:

```
NOTICE:  restoring database 'app', 6% of 7483 MB complete
NOTICE:  restoring database 'app', 42% of 7483 MB complete
NOTICE:  restoring database 'app', 77% of 7483 MB complete
NOTICE:  restoring database 'app', 98% of 7483 MB complete
NOTICE:  successfully restored database 'app' from node node1-app in
 00:04:12.274956
 pgactive_wait_for_node_ready
------------------------------
(1 row)
```

この時点から、pgactive は 2 つの DB インスタンス間でデータを同期します。

7. 次のコマンドを使用して、2 番目の DB インスタンスのデータベースにデータがあるかどうかを確認できます。

```
app=> SELECT count(*) FROM inventory.products;
```

データが正常に同期されると、次の出力が表示されます。

```
 count
-------
 3
```

8. 次のコマンドを実行して新しい値を挿入します。

```
app=> INSERT INTO inventory.products (id, product_name) VALUES (4, 'lotion');
```

9. 最初の DB インスタンスのデータベースに接続し、次のクエリを実行します。

```
app=> SELECT count(*) FROM inventory.products;
```

アクティブ/アクティブレプリケーションが初期化されると、出力は次のようになります。

```
count
-------
```

```
4
```

pgactive グループから DB インスタンスをデタッチして削除するには

pgactive グループから DB インスタンスをデタッチして削除するには、次の手順に従います。

1. 次のコマンドを使用して、最初のインスタンスから 2 番目の DB インスタンスをデタッチできます。

```
app=> SELECT * FROM pgactive.pgactive_detach_nodes(ARRAY['node2-app']);
```

2. 次のコマンドを使用して、2 番目の DB インスタンスから pgactive 拡張を削除します。

```
app=> SELECT * FROM pgactive.pgactive_remove();
```

拡張を強制的に削除するには

```
app=> SELECT * FROM pgactive.pgactive_remove(true);
```

3. 次のコマンドを使用して拡張をドロップします。

```
app=> DROP EXTENSION pgactive;
```

pgactive メンバー間のレプリケーションラグの測定

次のクエリを使用して、pgactive メンバー間のレプリケーションラグを表示できます。全体像を把握するには、すべての pgactive ノードでこのクエリを実行します。

```
app=> SELECT * FROM pgactive.pgactive_get_replication_lag_info();
#-[ RECORD 1 ]--------+-----------------------------------------------
#node_name            | node2-app
#node_sysid           | 7481018224801653637
#application_name     | pgactive:7481018224801653637:send
#slot_name            | pgactive_16385_7481018224801653637_0_16385__
#active               | t
#active_pid           | 783486
#pending_wal_decoding | 0
#pending_wal_to_apply | 0
```

```
#restart_lsn          | 0/2108150
#confirmed_flush_lsn  | 0/2154690
#sent_lsn             | 0/2154690
#write_lsn            | 0/2154690
#flush_lsn            | 0/2154690
#replay_lsn           | 0/2154690
#-[ RECORD 2 ]--------+-----------------------------------------
#node_name            | node1-app
#node_sysid           | 7481018033434600853
#application_name     | pgactive:7481018033434600853:send
#slot_name            | pgactive_16385_7481018033434600853_0_16385__
#active               | t
#active_pid           | 783488
#pending_wal_decoding | 0
#pending_wal_to_apply | 0
#restart_lsn          | 0/20F5AD0
#confirmed_flush_lsn  | 0/214EF68
#sent_lsn             | 0/214EF68
#write_lsn            | 0/214EF68
#flush_lsn            | 0/214EF68
#replay_lsn           | 0/214EF68
```

少なくとも次の診断をモニタリングします。

アクティブ

アクティブが false の場合にアラートを設定します。これは、スロットが現在使用されていない (サブスクライバーインスタンスがパブリッシャーから切断された) ことを示します。

Pending_wal_decoding

PostgreSQL の論理レプリケーションでは、WAL ファイルはバイナリ形式で保存されます。パブリッシャーは、これらの WAL の変更をデコードし、論理的な変更 (挿入、更新、削除オペレーションなど) に変換する必要があります。

pending_wal_decoding メトリクスは、パブリッシャー側でデコードを待っている WAL ファイルの数を示します。

この数は、次の要因により増加する可能性があります。

- サブスクライバーが接続されていない場合、アクティブステータスは false になり、pending_wal_decoding は増加する
- スロットはアクティブだが、パブリッシャーは WAL の変更の量に対応できない

pending_wal_to_apply

pending_wal_apply メトリクスは、サブスクライバー側で適用を待っている WAL ファイルの数を示します。

いくつかの要因により、サブスクライバーが変更を適用できなくなり、ディスクがいっぱいになるシナリオが発生する可能性があります。

- スキーマの違い - サンプルという名前のテーブルの WAL ストリームに変更があっても、そのテーブルがサブスクライバー側に存在しない場合など
- プライマリキー列の値が更新された場合
- セカンダリの一意のインデックスはデータの相違を引き起こす可能性がある

pgactive 拡張機能のパラメータ設定

次のクエリを使用すると、pgactive 拡張に関連するすべてのパラメータを表示できます。

```
app=> SELECT * FROM pg_settings WHERE name LIKE 'pgactive.%';
```

pgactive 拡張機能は、さまざまなパラメータを使用して設定できます。これらのパラメータは、AWS マネジメントコンソール または CLI AWS インターフェイスから設定できます。

主な pgactive 拡張機能パラメータ

次の表は、pgactive 拡張機能の主なパラメータのリファレンスです。

パラメータ	単位	デフォルト	説明
pgactive.conflict_logging_include_tuples	boolean	–	pgactive 拡張機能の完全なタプル情報を口グに記録します。 **Note** 変更を有効にするには、サーバー

パラメータ	単位	デフォルト	説明
			の再起動が必要です。
pgactive.log_conflicts_to_table	boolean	–	pgactive 拡張機能が検出された競合を pgactive.pgactive_conflict_history テーブルに口グ記録するかどうかを決定します。詳細については、「競合の口グ記録」を参照してください。 Note 変更を有効にするには、サーバーの再起動が必要です。

パラメータ	単位	デフォルト	説明
pgactive.log_conflicts_to_logfile	boolean	–	pgactive 拡張機能が検出された競合を PostgreSQL ログファイルに記録するかどうかを決定します。詳細については、「競合のログ記録」を参照してください。 Note 変更を有効にするには、サーバーの再起動が必要です。

パラメータ	単位	デフォルト	説明
pgactive.synchronous_commit	boolean	化	pgactive 適用ワーカーのコミット動作を決定します。無効 (オフ) にすると、適用ワーカーは非同期コミットを実行します。これにより、適用オペレーション中の PostgreSQL のスループットが向上しますが、アップストリームへのリプレイ確認が遅延します。off に設定しても常に安全であり、トランザクションの損失やスキップは発生しません。この設定は、ダウンストリームノードでのディスクフラッシュのタイミングと、確認がアップストリームに送信されるタイミングにのみ影響します。システムは、チェッ

パラメータ	単位	デフォルト	説明
			クポイントや定期的な作業などの関連しないオペレーションを通じてコミットがディスクにフラッシュされるまで、リプレイフラッシュ確認の送信を遅らせます。ただし、アップストリームノードの synchronous_standby_names にダウンストリームがリストされている場合、off に設定すると、アップストリームノードでの同期コミットがクライアントに成功を報告するまでの時間が長くなります。この場合、パラメータは on に設定してください。 Note このパラメータ

パラメータ	単位	デフォルト	説明
			が on に設定されていても、synchronous_standby_names にノードがリストされている場合は、アクティブ/アクティブ設定ではレプリケーションの競合が発生する可能性があります。これは、システムにはノード間ロックとグローバルスナップショット管理がないため、異なるノード上の同時トランザク

パラメータ	単位	デフォルト	説明
			ションが同じタプルを変更できるためです。さらに、トランザクションはアップストリームノードでコミットした後にのみレプリケーションを開始します。同期コミットを有効にしても、pgactive 拡張機能が常時一貫性のあるシステムになるわけではありません。

パラメータ	単位	デフォルト	説明
pgactive.temp_dump_directory	string	–	初期設定時のデータベースクローン作成オペレーションに必要な一時ストレージパスを定義します。このディレクトリは postgres ユーザーによって書き込み可能で、完全なデータベースダンプを格納するのに十分なストレージ容量を持っている必要があります。システムは、論理コピーオペレーションによるデータベースの初期セットアップ中にのみ、この場所を使用します。このパラメータは、pgactive_init_copy command では使用されません。

パラメータ	単位	デフォルト	説明
pgactive.max_ddl_lock_delay	milliseconds	-1	同時書き込みトランザクションを強制的に中止するまでの DDL ロックの最大待機時間を指定します。デフォルト値は -1 で、max_standby_streaming_delay で設定された値が使用されます。このパラメータには時間単位を指定できます。例えば、10 秒の場合は 10s と設定できます。この待機期間中、システムは進行中の書き込みトランザクションがコミットまたはロールバックされるのを待つ間に DDL ロックの取得を試みます。詳細については、「DDL ロック」を参照してください。

パラメータ	単位	デフォルト	説明
pgactive.ddl_lock_timeout	milliseconds	-1	DDL ロック試行がロックの取得を待機する時間を指定します。デフォルト値は -1 で、lock_timeout で指定された値を使用します。このパラメータは、10 秒の場合は 10s などの時間単位で設定できます。このタイマーは、DDL ロックを取得するための待機期間のみを制御します。システムがロックを取得し、DDL オペレーションを開始すると、タイマーは停止します。このパラメータは、DDL ロックを保持できる合計期間や全体的な DDL オペレーション時間を制限するものではありません。オペレーションの合計期間を制御

パラメータ	単位	デフォルト	説明
			するには、代わりに statement_timeout を使用します。詳細については、「DDL ロック」を参照してください。

パラメータ	単位	デフォルト	説明
pgactive.debug_trace_ddl_locks_level	boolean	–	pgactive 拡張機能の DDL ロックオペレーションのデフォルトのデバッグログレベルを上書きします。この設定を行うと、DDL ロック関連のメッセージがデフォルトレベルではなく LOG デバッグレベルで出力されます。このパラメータを使用して、サーバー全体で詳細な DEBUG1 または DEBUG2 ログレベルを有効にせずに、DDL ロックアクティビティをモニタリングできます。 詳細度の高い順序で使用可能なログレベル。 • none – DDL ロックメッセージは DEBUG1 以下のサーバーロ

パラメータ	単位	デフォルト	説明
			グレベルにのみ表示されます。 • statement - DDL ロックの取得試行の LOG 出力を追加します。 • acquire_release - ロック取得、リリース、拒否イベント、およびピアノードによるリモート DDL ロックの適用を記録します。 • peers - ピアノード間の DDL ロックネゴシエーションに関する追加の詳細を提供します。 • debug - すべての DDL ロック関連のアクティビティを LOG レベルでログに記録します。

パラメータ	単位	デフォルト	説明
			モニタリングオプションの詳細については、「グローバル DDL ロックのモニタリング」を参照してください。 Note この設定の変更は、設定を再読み込みすると有効になります。サーバーを再起動する必要はありません。

追加の pgactive 拡張機能パラメータ

次の表は、pgactive 拡張機能で利用可能な、あまり使用されない内部設定オプションを示しています。

パラメータ	単位	デフォルト	説明
pgactive.debug_apply_delay	integer	–	pgactive.pgactive_connections エントリに明示的に適用遅延が指定されていない設定済み接続の適用遅延 (ミリ秒単位) を設定します。この遅延はノード作成時または参加時に設定され、pgactive はコミットされてから指定されたミリ秒数が経過するまで、ピアノードでトランザクションをリプレイしません。 主に、テスト環境で高レイテンシーのネットワークをシミュレートして、競合を簡単に作成するために使用します。例えば、ノード A とノード B に 500 ミリ秒の遅延が

パラメータ	単位	デフォルト	説明
			設定されている場合、ノード A に値を挿入した後、ノード B で競合する挿入を実行するために少なくとも 500 ミリ秒の猶予があります。 Note 適用ワーカーを有効にするには、サーバーの再ロードまたは再起動が必要です。

パラメータ	単位	デフォルト	説明
pgactive.connectability_check_duration	integer	–	データベースワーカーが接続の確立に失敗した場合に、接続を確立しようとする時間 (秒単位) を指定します。ワーカーは、成功するかこのタイムアウト値に達するまで、1 秒ごとに接続を試行します。この設定は、ワーカーが接続を確立する準備ができる前にデータベースエンジンが起動する場合に便利です。

パラメータ	単位	デフォルト	説明
pgactive.skip_ddl_replication	boolean	on	pgactive が有効になっている Amazon RDS で DDL の変更をレプリケートまたは処理する方法を制御します。on に設定すると、ノードは非 pgcctive ノードと同様に DDL 変更を処理します。このパラメータを使用する場合、以下の要件が適用されます。 • 新しいノードは、skip_ddl_replication 値がアップストリームノードと異なる場合、pgactive グループに参加できません。 • 既存のノードは、パラメータ値がアップストリームノードと一致しない場合、p

パラメータ	単位	デフォルト	説明
			gactive ワーカーを起動できません。 • すべての pgactive メンバーは同じパラメータ値を使用する必要があります。 このパラメータは、スーパーユーザー権限を使用してグローバル、ローカル (セッションレベル) の 2 つの方法で変更できます。 Note このパラメータを誤って変更すると、レプリケーションのセットアップが中断される可能性

パラメータ	単位	デフォルト	説明
			があります。
pgactive.do_not_replicate	boolean	–	このパラメータは内部使用のみを目的としています。トランザクションでこのパラメータを設定すると、変更は DB クラスター内の他のノードにレプリケートされません。 Note このパラメータを誤って変更すると、レプリケーションのセットアップが中断される可能性があります。

パラメータ	単位	デフォルト	説明
pgactive.discard_mismatched_row_attributes	boolean	–	このパラメータは専門家による使用のみを目的としています。このパラメータは、特定のレプリケーションの問題をトラブルシューティングする場合にのみ使用することをお勧めします。このパラメータは、次の場合に使用します。 • 受信レプリケーションストリームに、ローカルテーブルよりも多くの列を持つ行が含まれている場合。 • これらのリモート行に NULL 以外の値が含まれている場合。 この設定により、以下のエラーメッセージがオーバー

パラメータ	単位	デフォルト	説明
			ライドされ、データの不一致が発生してもレプリケーションは続行されます。「cannot right-pad mismatched attributes; attno %u is missing in local table and remote row has non-null, non-dropped value for this attribute 」 **Note** このパラメータを誤って変更すると、レプリケーションのセットアップが中断される可能性

パラメータ	単位	デフォルト	説明
			があります。

パラメータ	単位	デフォルト	説明
pgactive.debug_trace_replay	boolean	–	on に設定すると、ダウンストリームの適用ワーカーが処理するリモートアクションごとにログメッセージが出力されます。ログには以下が含まれます。 • 変更の種類 • 影響を受けるテーブル名 • トランザクション開始以降の変更の数 • トランザクションコミット LSN • コミットタイムスタンプ • アップストリームノード識別子 • 転送ノード識別子 (該当する場合) ログには、キューに登録された DDL コマン

パラメータ	単位	デフォルト	説明
			ドとテーブルの DROP も記録されます。 para> デフォルトでは、ログには行フィールドの内容は含まれません。ログに行の値を含めるには、次のフラグを有効にして再コンパイルする必要があります。 • VERBOSE_INSERT • VERBOSE_UPDATE • VERBOSE_DELETE Note このログ記録設定を有効にすると、パフォーマンスに影響する可能性があります

パラメータ	単位	デフォルト	説明
			。トラブルシューティングに必要な場合にのみ有効にすることをお勧めします。この設定の変更は、設定を再読み込みすると有効になります。サーバーを再起動する必要はありません。

パラメータ	単位	デフォルト	説明
pgactive.extra_apply_connection_options		–	pgactive ノードとのすべてのピアノード接続の接続パラメータを設定できます。これらのパラメータは、keepalives や SSL モードなどの設定を制御します。デフォルトでは、pgactive は次の接続パラメータを使用します。 • connect_timeout=30 • keepalives=1 • keepalives_idle=20 • keepalives_interval=20 • keepalives_count=5 デフォルトのパラメータを上書きするには、以下の同様のコマンドを使用します。pgactive.extra_app

パラメータ	単位	デフォルト	説明
			`ly_connection_options = 'keepalives=0'` 個々のノード接続文字列は、これらの設定と pgactive の組み込み接続オプションの両方よりも優先されます。接続文字列形式の詳細については、「libpq connection strings」を参照してください。 デフォルトの keepalives 設定は有効のままにすることをお勧めします。信頼性の低いネットワーク経由で大規模なトランザクションが完了する際に問題が発生する場合にのみ、keepalives を無効にしてください。

パラメータ	単位	デフォルト	説明
			Note デフォルトのkeepalives設定は有効のままにすることをお勧めします。信頼性の低いネットワーク経由で大規模なトランザクションが完了する際に問題が発生する場合にのみ、keepalives を無効にしてください。この設定の変更は、設定を再読み込みすると有効になります。サー

パラメータ	単位	デフォルト	説明
			バーを再起動する必要はありません。
pgactive.init_node_parallel_jobs (int)		–	pgactive.pgactive_join_group 関数を使用して、論理ノードの結合時に pg_dump と pg_restore が使用できる並列ジョブの数を指定します。 この設定の変更は、設定を再読み込みすると有効になります。サーバーを再起動する必要はありません。

パラメータ	単位	デフォルト	説明
pgactive.max_nodes	int	4	pgactive 拡張グループで許可されるノードの最大数を指定します。デフォルト値は 4 ノードです。このパラメータの値を設定するときは、次の点を考慮する必要があります。 • pgactive 拡張グループ内のすべてのノードは、同じパラメータ値を使用する必要があります。 • パラメータ値がアップストリームノードと異なる場合、新しいノードは結合できません。 • パラメータ値がアップストリームノードと異なる場合、既存のノードは pgactive 拡

パラメータ	単位	デフォルト	説明
			張ワーカーを開始できません。 • 大規模なグループでは、追加のモニタリングとメンテナンス作業が必要になるため、このパラメータの値は慎重に設定してください。 このパラメータは、設定ファイルで設定する方法と、ALTER SYSTEM SET コマンドを使用する方法の 2 つの方法で設定できます。 このパラメータのデフォルト値は 4 です。つまり、pgactive 拡張グループには、いつでも最大 4 ノードまでしか参加できません。

パラメータ	単位	デフォルト	説明
			Note この変更は、サーバーを再起動した後に有効になります。
pgactive.permit_node_identifier_getter_function_creation	boolean	–	このパラメータは、内部使用のみを目的としています。有効にすると、pgactive 拡張機能により、pgactive ノード識別子 getter 関数の作成が可能になります。

アクティブ/アクティブ競合について

pgactive をアクティブ/アクティブモードで使用する場合、複数のノードから同じテーブルに書き込むと、データの競合が発生する可能性があります。一部のクラスタリングシステムでは、同時アクセスを防ぐために分散ロックを使用していますが、pgactive は地理的に分散されたアプリケーションに適した楽観的なアプローチを採用しています。

一部のデータベースクラスタリングシステムは、分散ロックを使用して同時データアクセスを防止します。このアプローチはサーバーが近接している場合に機能しますが、優れたパフォーマンスのためには非常に低いレイテンシーが必要となるため、地理的に分散したアプリケーションはサポートして

いません。pgactive 拡張機能は、分散ロック (悲観的アプローチ) を使用する代わりに、楽観的アプローチを使用します。つまり、次のことを意味します。

- 可能な場合には競合を回避できます。
- 特定のタイプの競合の発生を許可します。
- 競合が発生した場合に競合を解決します。

このアプローチにより、分散アプリケーションを構築する際の柔軟性が向上します。

競合の発生方法

ノード間の競合は、関連するすべてのトランザクションが同じノードで同時に発生した場合には発生しない一連のイベントから発生します。ノード間の変更はトランザクションのコミット後にのみ行われるため、各トランザクションはコミットされたノードで個別に有効ですが、その間に他の作業を行った別のノードで実行された場合は有効になりません。pgactive 適用は基本的に他のノードでトランザクションを再実行するため、適用されているトランザクションと受信側ノードでコミットされたトランザクションの間に競合がある場合、再実行オペレーションが失敗する可能性があります。

すべてのトランザクションが単一のノードで実行されている場合にほとんどの競合が発生しないのは、PostgreSQL にそれを防ぐためのトランザクション間通信メカニズムがあるからです。これには、以下が含まれます。

- UNIQUE インデックス
- SEQUENCE
- 行とリレーションのロック
- SERIALIZABLE 依存関係の追跡

これらのメカニズムはすべて、望ましくない同時実行の問題を防ぐためにトランザクション間で通信する方法です。

pgactive は、分散トランザクションマネージャーやロックマネージャーを使用しないため、低レイテンシーを実現し、ネットワークパーティションを適切に処理します。ただし、これは、異なるノード上のトランザクションが相互に完全に分離して実行されることを意味します。通常、分離はデータベースの一貫性を向上させますが、この場合は、競合を防ぐために分離を減らす必要があります。

競合のタイプ

発生する可能性のある競合には、以下が含まれます。

トピック

- PRIMARY KEY または UNIQUE の競合
- INSERT/INSERT の競合
- 複数の UNIQUE 制約に違反する INSERT
- UPDATE/UPDATE の競合
- PRIMARY KEY の UPDATE 競合
- 複数の UNIQUE 制約に違反するUPDATE
- UPDATE/DELETE の競合
- INSERT/UPDATE の競合
- DELETE/DELETE の競合
- 外部キー制約の競合
- 排他制約の競合
- グローバルデータの競合
- ロックの競合とデッドロックの中止
- 相違による競合

PRIMARY KEY または UNIQUE の競合

行の競合は、複数のオペレーションが単一のノードで同じ行キーを変更しようとしたときに発生します。これらの競合は、最も一般的なタイプのデータ競合です。

pgactive は検出された競合を、最終更新を優先する処理またはカスタム競合ハンドラーで解決します。

行の競合には以下が含まれます。

- INSERT と INSERT
- INSERT と UPDATE
- UPDATE と DELETE
- INSERT と DELETE
- DELETE と DELETE
- INSERT と DELETE

INSERT/INSERT の競合

この最も一般的な競合は、2 つの異なるノードの INSERT が同じ PRIMARY KEY 値 (または PRIMARY KEY が存在しない場合は同じ UNIQUE 制約値) を持つタプルを作成するときに発生します。

pgactivelink は、発信元ホストのタイムスタンプを使用して最新のタプルを保持することで、INSERT の競合を解決します。このデフォルトの動作は、カスタム競合ハンドラーで上書きできます。このプロセスでは特別な管理者アクションは必要ありませんが、pgactivelink はノード全体で 1 つの INSERT オペレーションを破棄することに注意してください。カスタムハンドラーが実装しない限り、自動データマージは発生しません。

pgactivelink は、単一の制約違反に関連する競合のみを解決できます。INSERT が複数の UNIQUE 制約に違反している場合は、追加の競合解決戦略を実装する必要があります。

複数の UNIQUE 制約に違反する INSERT

INSERT/INSERT 競合は、PRIMARY KEY を含む複数の UNIQUE 制約に違反する可能性があります。pgactivelink は、単一の UNIQUE 制約を含む競合のみを処理できます。競合が複数の UNIQUE 制約に違反すると、適用ワーカーは失敗し、次のエラーを返します。

```
multiple unique constraints violated by remotely INSERTed tuple.
```

古いバージョンでは、この状況では代わりに「一意性の相違による競合」エラーが発生しました。

これらの競合を解決するには、手動でアクションを実行する必要があります。競合するローカルタプルを DELETE するか、UPDATE して新しいリモートタプルとの競合を削除します。複数の競合するタプルに対処する必要がある場合があることに注意してください。現在、pgactivelink には、複数の一意の制約に違反するタプルを無視、破棄、またはマージする組み込み機能はありません。

> **Note**
>
> 詳細については、「複数の UNIQUE 制約に違反する UPDATE」を参照してください。

UPDATE/UPDATE の競合

この競合は、2 つのノードが PRIMARY KEY を変更せずに同じタプルを同時に変更した場合に発生します。pgactivelink は、定義されている場合には、最終更新を優先するロジックまたはカスタム競合ハンドラーを使用してこれらの競合を解決します。PRIMARY KEY は、タプルマッチングと競合

の解決に不可欠です。PRIMARY KEY のないテーブルの場合、pgactivelink は次のエラーで UPDATE オペレーションを拒否します。

```
Cannot run UPDATE or DELETE on table (tablename) because it does not have a primary key.
```

PRIMARY KEY の UPDATE 競合

pgactive は、PRIMARY KEY 更新を処理する際に制限があります。PRIMARY KEY で UPDATE オペレーションを実行することはできますが、pgactive はこれらのオペレーションに最終更新を優先するロジックを使用して競合を自動的に解決することはできません。PRIMARY KEY の更新が既存の値と競合しないようにする必要があります。PRIMARY KEY の更新中に競合が発生した場合、相違による競合が発生し、手動による介入が必要になります。これらの状況の処理の詳細については、「相違による競合」を参照してください。

複数の UNIQUE 制約に違反するUPDATE

pgactivelink は、受信する UPDATE が複数の UNIQUE 制約または PRIMARY KEY 値に違反している場合、最終更新を優先する競合解決を適用できません。この動作は、複数の制約違反がある INSERT オペレーションに似ています。このような状況では、手動による介入を必要とする相違による競合が発生します。詳細については、「相違による競合」を参照してください。

UPDATE/DELETE の競合

この競合は、あるノードが行を UPDATE し、別のノードが同時にその行を DELETE した場合に発生します。この場合、再実行時に UPDATE/DELETE 競合が発生します。解決策は、カスタム競合ハンドラーが特に指定しない限り、DELETE の後に送信される UPDATE をすべて破棄することです。

pgactivelink は、タプルを照合して競合を解決するために PRIMARY KEY を必要とします。PRIMARY KEY のないテーブルの場合、次のエラーで DELETE オペレーションが拒否されます。

```
Cannot run UPDATE or DELETE on table (tablename) because it does not have a primary key.
```

> **Note**
>
> pgactivelink は、UPDATE/DELETE と INSERT/UPDATE の競合を区別できません。どちらの場合も、UPDATE は存在しない行に影響します。非同期レプリケーションおよびノード間に再実行順序がないことにより、pgactivelink は UPDATE が新しい行 (INSERT をまだ受信

していない) に対するものか、削除された行に対するものかを判断できません。どちらのシナリオでも、pgactivelink は UPDATE を破棄します。

INSERT/UPDATE の競合

この競合は、マルチノード環境で発生する可能性があります。これは、あるノードが行を INSERT し、2 番目のノードが行を UPDATE し、3 番目のノードが元の INSERT の前に UPDATE を受信したときに発生します。デフォルトでは、カスタム競合トリガーで特に指定されていない限り、pgactivelink は UPDATE を破棄することで、これらの競合を解決します。この解決方法により、ノード間でデータの不整合が発生する可能性があることに注意してください。同様のシナリオとその処理の詳細については、「UPDATE/DELETE の競合」を参照してください。

DELETE/DELETE の競合

この競合は、2 つの異なるノードが同じタプルを同時に削除した場合に発生します。pgactivelink は、両方の DELETE オペレーションの結果が同じであるため、これらの競合を無害と見なします。このシナリオでは、pgactivelink はデータ整合性に影響を与えることなく、一方の DELETE オペレーションを安全に無視します。

外部キー制約の競合

FOREIGN KEY の制約により、既存のローカルデータにリモートトランザクションを適用するときに競合が発生する可能性があります。これらの競合は通常、トランザクションが発信元ノードの論理順序とは異なる順序で適用された場合に発生します。

デフォルトでは、pgactive は session_replication_role を `replica` として変更を適用し、レプリケーション中に外部キーチェックをバイパスします。アクティブ/アクティブ設定では、これが外部キー違反につながる可能性があります。ほとんどの違反は一時的なものであり、レプリケーションが追いつくと解決されます。ただし、pgactive はクロスノードの行ロックをサポートしていないため、外部キーのダングリングが発生する可能性があります。

この動作は、分断耐性のある非同期アクティブ/アクティブシステムに固有のものです。例えば、ノード A が新しい子行を挿入すると同時に、ノード B がその親行を削除する場合があります。システムは、ノード間でのこのタイプの同時変更を防ぐことはできません。

外部キーの競合を最小限に抑えるには、以下のことをお勧めします。

- 外部キー関係を密接に関連するエンティティに制限します。

- 可能な場合は、単一のノードから関連エンティティを変更します。
- ほとんど変更を必要としないエンティティを選択します。
- 変更に対してアプリケーションレベルの同時実行制御を実装します。

排他制約の競合

pgactivelink は排他制約をサポートしておらず、作成を制限します。

> **Note**
>
> 既存のスタンドアロンデータベースを pgactivelink データベースに変換する場合は、すべての排他制約を手動で削除します。

分散非同期システムでは、制約に違反する行のセットが存在しないことは保証できません。これは、異なるノード上のすべてのトランザクションが完全に分離されているためです。排他制約は、排他制約違反のためにどのノードからも別のノードに再実行できなくなる、再実行デッドロックにつながる可能性があります。

pgactivelink に排他制約の作成を強制した場合、またはスタンドアロンデータベースを pgactivelink に変換するときに既存の排他制約を削除しない場合、レプリケーションが中断する可能性があります。レプリケーションの進行状況を回復するには、受信リモートタプルと競合するローカルタプルを削除または変更し、リモートトランザクションを適用できるようにします。

グローバルデータの競合

pgactivelink を使用する場合、ノード間でロールなどの PostgreSQL システム全体のグローバルデータが異なると、競合が発生する可能性があります。これらの競合により、オペレーション (主に DDL) が 1 つのノードで成功してコミットされるものの、他のノードには適用されない可能性があります。

ユーザーが 1 つのノードに存在するものの、別のノードには存在しない場合、レプリケーションの問題が発生する可能性があります。

- Node1 には `fred` という名前のユーザーがいますが、このユーザーは Node2 には存在しません。
- `fred` が Node1 にテーブルを作成すると、テーブルは `fred` を所有者としてレプリケートされます。
- この DDL コマンドを Node2 に適用すると、ユーザー `fred` が存在しないため、失敗します。

- この失敗により、Node2 の PostgreSQL ログに ERROR が生成され、`pgactive.pgactive_stats.nr_rollbacks` カウンターが増分されます。

解決策: Node2 でユーザー `fred` を作成します。ユーザーが同じアクセス許可を持つ必要はありませんが、両方のノードに存在する必要があります。

1 つのノードにテーブルが存在するものの、別のノードには存在しない場合、データ変更オペレーションは失敗します。

- Node1 には、Node2 に存在しない `foo` という名前のテーブルがあります。
- Node1 の `foo` テーブルに対する DML オペレーションは、Node2 にレプリケートされると失敗します。

解決策: 同じ構造で Node2 にテーブル `foo` を作成します。

> **Note**
>
> pgactivelink は現在、CREATE USER コマンドまたは DDL オペレーションをレプリケートしません。DDL レプリケーションは将来のリリースが予定されています。

ロックの競合とデッドロックの中止

pgactive 適用プロセスは通常のユーザーセッションのように動作するため、標準的な行とテーブルのロックルールに従います。そのため、pgactivelink 適用プロセスは、ユーザートランザクションまたは他の適用プロセスによって保持されているロックを待機する可能性があります。

以下のタイプのロックは適用プロセスに影響を与える可能性があります。

- ユーザーセッションによる明示的なテーブルレベルのロック (LOCK TABLE...)
- ユーザーセッションによる明示的な行レベルのロック (SELECT ... FOR UPDATE/FOR SHARE)
- 外部キーからのロック
- ローカルアクティビティまたは他のサーバーからの行の UPDATE、INSERT、または DELETE による暗黙的なロック

デッドロックは、次の間で発生する可能性があります。

- pgactivelink 適用プロセスとユーザートランザクション

- 2 つの適用プロセス

デッドロックが発生すると、PostgreSQL のデッドロック検出機能は問題のトランザクションのうち 1 つを終了します。pgactivelink 適用ワーカーのプロセスが終了すると、自動的に再試行され、通常は成功します。

> **Note**
>
> - これらの問題は一時的なものであり、通常は管理者の介入は必要ありません。適用プロセスがアイドル状態のユーザーセッションのロックによって長期間ブロックされている場合は、ユーザーセッションを終了してレプリケーションを再開できます。この状況は、ユーザーが別のユーザーセッションに影響を与える長期間にわたるロックを保持する場合に似ています。
> - ロック関連の再生遅延を特定するには、PostgreSQL で `log_lock_waits` 機能を有効にします。

相違による競合

ノード間で同一であるはずのデータが予期せず異なる場合、相違による競合が発生します。これらの競合は発生すべきではないものの、現在の実装ではすべてを確実に防止できるわけではありません。

> **Note**
>
> 行の PRIMARY KEY を変更すると、すべてのノードが変更を処理する前に別のノードが同じ行のキーを変更した場合、相違による競合が発生する可能性があります。プライマリキーの変更を避けるか、変更を指定された 1 つのノードに制限してください。詳細については、「the section called “PRIMARY KEY の UPDATE 競合”」を参照してください。

行データが関係した相違による競合には通常、管理者の介入が必要です。これらの競合を解決するには、`pgactive.pgactive_do_not_replicate` を使用してレプリケーションを一時的に無効にし、一方のノードのデータをもう一方のノードに合わせて手動で調整する必要があります。これらの競合は、pgactive を記載通りに使用し、安全でないとマークされた設定や関数を回避すれば、発生しません。

管理者は、これらの競合を手動で解決する必要があります。競合タイプによっては、`pgactive.pgactive_do_not_replicate` などの高度なオプションを使用する必要がありま

す。不適切に使用すると状況が悪化する可能性があるため、これらのオプションは慎重に使用してください。競合にはさまざまな可能性が考えられるため、普遍的な解決の手順を提供することはできません。

異なるノード間で同一であるはずのデータが予期せず異なる場合、相違による競合が発生します。これらの競合は発生すべきではないものの、現在の実装ではこのような競合をすべて確実に防止できるわけではありません。

競合の回避または許容

ほとんどの場合、適切なアプリケーション設計を使用することで、競合を回避したり、アプリケーションが競合を許容するようにしたりできます。

競合は、複数のノードで同時操作が行われた場合にのみ発生します。競合を回避するには、以下を行います。

- 1 つのノードにのみ書き込む
- 各ノードの独立したデータベースサブセットに書き込む (各ノードに個別のスキーマを割り当てるなど)

INSERT と INSERT の競合の場合は、グローバルシーケンスを使用して競合を完全に防止します。

競合が許容できないユースケースの場合は、アプリケーションレベルで分散ロックを実装することを検討してください。多くの場合、最善の方法は、すべての競合を防ぐのではなく、pgactive の競合解決メカニズムと連携するようにアプリケーションを設計することです。詳細については、「the section called “競合のタイプ”」を参照してください。

競合のログ記録

pgactivelink は、競合インシデントを `pgactive.pgactive_conflict_history` テーブルに記録し、アクティブ/アクティブ競合の診断と処理に役立つようにします。このテーブルへの競合ログは、`pgactive.log_conflicts_to_table` を true に設定した場合にのみ記録されます。pgactive 拡張機能は、log_min_messages が `LOG` または `lower` に設定されている場合に、`pgactive.log_conflicts_to_table` 設定に関係なく、競合を PostgreSQL ログファイルにも記録します。

競合履歴テーブルは、以下の目的で使用します。

- アプリケーションに競合が起きる頻度を測定する
- 競合が発生する場所を特定する

- アプリケーションを改善し、競合率を低減する
- 競合の解決策により望ましい結果が得られないケースを検出する
- ユーザー定義の競合トリガーまたはアプリケーション設計の変更が必要な場所を特定する

行が競合した場合、オプションで行の値をログに記録できます。これは `pgactive.log_conflicts_to_table` 設定によって制御されます。以下の点に注意してください。

- これはデータベース全体のグローバルオプションです。
- 行値のログ記録をテーブル別に制御することはできません。
- フィールド番号、配列要素、またはフィールド長に制限はありません。
- 競合を引き起こす可能性のある数メガバイトの行を使用する場合は、この機能を有効にすることはお勧めできません。

競合履歴テーブルには、データベース内のすべてのテーブルのデータ (それぞれ異なるスキーマを持つ可能性がある) が含まれているため、ログに記録された行値は JSON フィールドとして保存されます。JSON は、SQL から直接呼び出す場合と同様に、`row_to_json` を使用して作成されます。PostgreSQL では `json_to_row` 関数が提供されていないため、ログに記録された JSON から複合型のタプルを再構築するには、テーブル固有のコード (PL/pgSQL、PL/Python、PL/Perl など) が必要です。

> **Note**
>
> ユーザー定義の競合のサポートは、今後の拡張機能として計画されています。

pgactive スキーマについて

pgactive スキーマは、RDS for PostgreSQL のアクティブ/アクティブレプリケーションを管理します。このスキーマには、レプリケーション設定とステータス情報を保存するテーブルが含まれています。

> **Note**
>
> pgactive スキーマは進化しており、変更される可能性があります。これらのテーブルのデータを直接変更しないでください。

pgactive スキーマのキーテーブルには以下のものが含まれます。

- `pgactive_nodes` – アクティブ/アクティブレプリケーショングループ内のノードに関する情報を保存します。
- `pgactive_connections` – 各ノードの接続の詳細を保存します。

pgactive_nodes

pgactive_nodes は、アクティブ/アクティブレプリケーショングループに参加しているノードに関する情報を保存します。

列	タイプ	照合	Nullable	デフォルト
node_sysid	text	–	NOT NULL	–
node_timeline	oid	–	NOT NULL	–
node_dboid	oid	–	NOT NULL	–
node_status	char	–	NOT NULL	–
node_name	text	–	NOT NULL	–
node_dsn	text	–	NOT NULL	–
node_init_from_dsn	text	–	NOT NULL	–
node_read_only	ブール値	–	–	false
node_seq_id	smallint	–	NOT NULL	–

node_sysid

ノードの一意の ID。`pgactive_create_group` または `pgactive_join_group` 中に生成されます。

node_status

ノードの準備状況:

- b - セットアップの開始
- i - 初期化
- c - キャッチアップ
- o - アウトバウンドスロットの作成
- r - 準備完了
- k - 強制終了

> **Note**
> この列は、ノードが接続されているか切断されているかを示しません。

node_name

ユーザーが指定した一意のノード名。

node_dsn

接続文字列またはユーザーマッピング名

node_init_from_dsn

このノードが作成された DSN。

pgactive_connection

pgactive_connections は、各ノードの接続の詳細を保存します。

列	タイプ	照合	Nullable	デフォルト	
conn_sysid	text	なし	NOT NULL	なし	
conn_timeline	oid	なし	NOT NULL	なし	
conn_dboid	oid	なし	NOT NULL	なし	

列	タイプ	照合	Nullable	デフォルト
conn_dsn	text	なし	NOT NULL	なし
conn_apply_delay	integer	なし	なし	なし
conn_replication_sets	text	なし	なし	なし

conn_sysid

このエントリが参照するノードのノード識別子。

conn_dsn

pgactive.pgactive_nodes node_dsn と同じです。

conn_apply_delay

設定されている場合、リモートノードから各トランザクションを適用するまで待機するミリ秒数。主にデバッグ用。null の場合、グローバルデフォルトが適用されます。

レプリケーションセットの使用

レプリケーションセットは、レプリケーションオペレーションに含めるテーブルと除外するテーブルを決定します。デフォルトでは、次の関数を使用して指定しない限り、すべてのテーブルがレプリケートされます。

- `pgactive_exclude_table_replication_set()` - 指定されたテーブルをレプリケーションから除外する
- `pgactive_include_table_replication_set()` - 指定されたテーブルをレプリケーションに含める

> **Note**
>
> レプリケーションセットを設定する前に、次の点を考慮してください。
>
> - テーブルの包含または除外は、`pgactive_create_group()` の実行後、`pgactive_join_group()` を実行するより前にのみ設定できます。

- pgactive_exclude_table_replication_set() を使用した後、pgactive_include_table_replication_set() を使用することはできません。
- pgactive_exclude_table_replication_set() を使用した後、pgactive_include_table_replication_set() を使用することはできません。

システムは、初期設定に基づいて新しく作成されたテーブルを異なる方法で処理します。

- テーブルを除外した場合: pgactive_join_group() 以降に作成された新しいテーブルは、自動的にレプリケーションに含まれます。
- テーブルを含めた場合: pgactive_join_group() 以降に作成された新しいテーブルは、自動的にレプリケーションから除外されます。

特定のテーブルのレプリケーションセット設定を表示するには、pgactive.pgactive_get_table_replication_sets() 関数を使用します。

pgactive 関数リファレンス

以下に、pgactive 関数のリストとそのパラメータ、戻り値、実用的な使用上の注意事項を示します。

get_last_applied_xact_info

指定されたノードに最後に適用されたトランザクション情報を取得します。

引数

- sysid (テキスト) - タイムライン OID
- dboid (OID)

戻り型

以下を記録します。

- last_applied_xact_id (OID)
- last_applied_xact_committs (タイムゾーン付きのタイムスタンプ)
- last_applied_xact_at (タイムゾーン付きのタイムスタンプ)

使用に関する注意事項

この関数を使用して、指定されたノードに最後に適用されたトランザクション情報を取得します。

pgactive_apply_pause

レプリケーション適用プロセスを一時停止します。

引数

なし

戻り型

boolean

使用に関する注意事項

この関数を呼び出して、レプリケーション適用プロセスを一時停止します。

pgactive_apply_resume

レプリケーション適用プロセスを再開します。

引数

なし

戻り型

void

使用に関する注意事項

この関数を呼び出して、レプリケーション適用プロセスを再開します。

pgactive_is_apply_paused

レプリケーション適用が現在一時停止されているかどうかを確認します。

引数

なし

戻り型

boolean

使用に関する注意事項

この関数を使用して、レプリケーション適用が現在一時停止されているかどうかを確認します。

pgactive_create_group

スタンドアロンデータベースを初期ノードに変換して pgactive グループを作成します。

引数

- node_name (テキスト)
- node_dsn (テキスト)
- apply_delay integer DEFAULT NULL::integer - replication_sets text[] DEFAULT ARRAY[‘default’::text]

戻り型

void

使用に関する注意事項

スタンドアロンデータベースを初期ノードに変換して pgactive グループを作成します。この関数は、ノードを pgactive ノードに変換する前に、サニティチェックを実行します。この関数を使用する前に、PostgreSQL クラスターに pgactive バックグラウンドワーカーをサポートするのに十分な `max_worker_processes` が使用可能であることを確認してください。

pgactive_detach_nodes

pgactive グループから指定されたノードを削除します。

引数

- p_nodes (text[])

戻り型

void

使用に関する注意事項

この関数を使用して、pgactive グループから指定されたノードを削除します。

pgactive_exclude_table_replication_set

レプリケーションから特定のテーブルを除外します。

引数

- p_relation (regclass)

戻り型

void

使用に関する注意事項

この関数を使用して、レプリケーションから特定のテーブルを除外します。

pgactive_get_replication_lag_info

ノードの詳細、WAL ステータス、LSN 値など、詳細なレプリケーションラグ情報を取得します。

引数

なし

戻り型

SETOF record - node_name text - node_sysid text - application_name text - slot_name text - active boolean - active_pid integer - pending_wal_decoding bigint - Approximate size of WAL in bytes to be decoded on the sender node - pending_wal_to_apply bigint - Approximate size of WAL in bytes to be applied on receiving node - restart_lsn pg_lsn - confirmed_flush_lsn pg_lsn - sent_lsn pg_lsn - write_lsn pg_lsn - flush_lsn pg_lsn - replay_lsn pg_lsn

使用に関する注意事項

この関数を呼び出して、ノードの詳細、WAL ステータス、LSN 値などのレプリケーションラグ情報を取得します。

pgactive_get_stats

pgactive レプリケーションの統計を取得します。

引数

なし

戻り型

SETOF record - rep_node_id oid - rilocalid oid - riremoteid text - nr_commit bigint - nr_rollback bigint - nr_insert bigint - nr_insert_conflict bigint - nr_update bigint - nr_update_conflict bigint - nr_delete bigint - nr_delete_conflict bigint - nr_disconnect bigint

使用に関する注意事項

この関数を使用して、pgactive レプリケーションの統計を取得します。

pgactive_get_table_replication_sets

特定のリレーションのレプリケーションセット設定を取得します。

引数

- リレーション (regclass)

戻り型

SETOF レコード

使用に関する注意事項

この関数を呼び出して、特定のリレーションのレプリケーションセット設定を取得します。

pgactive_include_table_replication_set

レプリケーションに特定のテーブルを含めます。

引数

- p_relation (regclass)

戻り型

void

使用に関する注意事項

この関数を使用して、レプリケーションに特定のテーブルを含めます。

pgactive_join_group

既存の pgactive グループにノードを追加します。

引数

- node_name (テキスト)
- node_dsn (テキスト)
- join_using_dsn (テキスト)

- apply_delay (整数、オプション)
- replication_sets (text[]、デフォルト: ['default'])
- bypass_collation_check (ブール値、デフォルト: false)
- bypass_node_identifier_creation (ブール値、デフォルト: false)
- bypass_user_tables_check (ブール値、デフォルト: false)

戻り型

void

使用に関する注意事項

この関数を呼び出して、既存の pgactive グループにノードを追加します。PostgreSQL クラスターに pgactive バックグラウンドワーカー用の十分な max_worker_processes があることを確認します。

pgactive_remove

ローカルノードからすべての pgactive コンポーネントを削除します。

引数

- force (ブール値、デフォルト: false)

戻り型

void

使用に関する注意事項

この関数を呼び出して、ローカルノードからすべての pgactive コンポーネントを削除します。

pgactive_snowflake_id_nextval

ノード固有の一意のシーケンス値を生成します。

引数

- regclass

戻り型

bigint

使用に関する注意事項

この関数を使用して、ノード固有の一意のシーケンス値を生成します。

pgactive_update_node_conninfo

pgactive ノードの接続情報を更新します。

引数

- node_name_to_update (テキスト)
- node_dsn_to_update (テキスト)

戻り型

void

使用に関する注意事項

この関数を使用して、pgactive ノードの接続情報を更新します。

pgactive_wait_for_node_ready

グループの作成または参加オペレーションの進行状況をモニタリングします。

引数

- タイムアウト (整数、デフォルト: 0)
- progress_interval (整数、デフォルト: 60)

戻り型

void

使用に関する注意事項

この関数を呼び出して、グループの作成または参加オペレーションの進行状況をモニタリングします。

アクティブ/アクティブレプリケーションの競合の処理

pgactive 拡張は、クラスターごとではなく、データベースごとに機能します。pgactive を使用する各 DB インスタンスは、独立したインスタンスであり、あらゆるソースからのデータ変更を受け入れることができます。変更が DB インスタンスに送信されると、PostgreSQL は変更をローカルに

コミットし、pgactive を使用して他の DB インスタンスに非同期に変更をレプリケートします。2 つの PostgreSQL DB インスタンスが同じレコードをほぼ同時に更新すると、競合が発生する可能性があります。

pgactive 拡張は、競合の検出と自動解決のためのメカニズムを提供します。両方の DB インスタンスでトランザクションがコミットされた時点のタイムスタンプを追跡し、最新のタイムスタンプで変更を自動的に適用します。また、pgactive 拡張は、pgactive.pgactive_conflict_history テーブルで競合が発生した場合もログに記録します。

pgactive.pgactive_conflict_history は継続的に増大します。パージポリシーを定義するとよいでしょう。これを行うには、一部のレコードを定期的に削除するか、この関係のパーティションスキームを定義します (その後で対象のパーティションをデタッチ、ドロップ、切り捨てることができます)。パージポリシーを定期的に実装するには、 pg_cron 拡張機能を使用するというオプションがあります。pg_cron 履歴テーブルの例については、「PostgreSQL pg_cron 拡張機能を使用したメンテナンスのスケジュール」の次の情報を参照してください。

アクティブ/アクティブレプリケーションでのシーケンスの処理

pgactive 拡張を使用した RDS for PostgreSQL DB インスタンスは、2 つの異なるシーケンスメカニズムを使用して固有の値を生成します。

グローバルシーケンス

グローバルシーケンスを使用するには、CREATE SEQUENCE ステートメントを使用してローカルシーケンスを作成します。usingnextval(seqname) の代わりに pgactive.pgactive_snowflake_id_nextval(seqname) を使用すると、シーケンスの次の固有な値を取得できます。

次の例では、グローバルシーケンスを作成します。

```
app=> CREATE TABLE gstest (
      id bigint primary key,
      parrot text
    );
```

```
app=>CREATE SEQUENCE gstest_id_seq OWNED BY gstest.id;
```

```
app=> ALTER TABLE gstest \
```

```
    ALTER COLUMN id SET DEFAULT \
    pgactive.pgactive_snowflake_id_nextval('gstest_id_seq');
```

分割シーケンス

分割ステップまたは分割シーケンスでは、通常の PostgreSQL シーケンスをノードごとに使用します。各シーケンスは同じ量ずつインクリメントされ、異なるオフセットから始まります。例えば、ステップ 100 の場合、ノード 1 は 101、201、301 などとしてシーケンスを生成し、ノード 2 は 102、202、302 などとしてシーケンスを生成します。このスキームは、ノードが長時間通信できない場合でも適切に機能しますが、設計者はスキーマを確立するときに最大ノード数を指定する必要があり、ノードごとの設定が必要になります。間違えると、シーケンスが重複しやすくなります。

次に示すように、ノードで目的のシーケンスを作成することで、このアプローチを pgactive で比較的簡単に設定できます。

```
CREATE TABLE some_table (generated_value bigint primary key);
```

```
app=> CREATE SEQUENCE some_seq INCREMENT 100 OWNED BY some_table.generated_value;
```

```
app=> ALTER TABLE some_table ALTER COLUMN generated_value SET DEFAULT
 nextval('some_seq');
```

次に、各ノードで setval を呼び出して、次のように異なるオフセットの開始値を指定します。

```
app=>
-- On node 1
SELECT setval('some_seq', 1);

-- On node 2
SELECT setval('some_seq', 2);
```

pg_repack 拡張機能を使用して、テーブルやインデックスの膨張を抑制する

pg_repack 拡張機能を使用して、VACUUM FULL の代わりとしてテーブルやインデックスの肥大化を取り除くことができます。このエクステンションは、RDS for PostgreSQL のバージョン 9.6.3 以降でサポートされています。pg_repack 拡張機能の詳細については、GitHub プロジェクトのドキュメントをご覧ください。

VACUUM FULL とは異なり、pg_repack 拡張機能では、次の場合にテーブルの再構築オペレーション中に短期間だけ排他的ロック (AccessExclusiveLock) が必要です。

- ログテーブルの初回作成 – 次の例に示すように、データの初回コピー中に発生した変更を記録するログテーブルが作成されます。

```
postgres=>\dt+ repack.log_*
List of relations
-[ RECORD 1 ]-+----------
Schema        | repack
Name          | log_16490
Type          | table
Owner         | postgres
Persistence   | permanent
Access method | heap
Size          | 65 MB
Description   |
```

- 最終スワップアンドドロップフェーズ。

再構築オペレーションの残りの部分で必要なのは、元のテーブルから新しいテーブルに行をコピーするための ACCESS SHARE ロックのみです。これにより、INSERT、UPDATE、DELETE オペレーションを通常どおりに進めることができます。

推奨事項

次の推奨事項は、pg_repack 拡張機能を使用してテーブルとインデックスの肥大化を取り除く場合に適用されます。

- 業務時間外または他のデータベースアクティビティのパフォーマンスへの影響を最小限に抑えるために、メンテナンスウィンドウで再パックを実行します。

- 再構築アクティビティ中にブロッキングセッションを注意深くモニタリングし、pg_repack をブロックする可能性のあるアクティビティが元のテーブルにないことを確認します。特に、元のテーブルで排他的ロックが必要なときは、最後のスワップアンドドロップフェーズ中にアクティビティがないことを確認します。詳細については、「クエリをブロックしているものの特定」を参照してください。

  ブロッキングセッションが表示された場合は、慎重に検討した後、次のコマンドを使用してセッションを終了できます。これは、pg_repack の継続によって再構築を完了するのに役立ちます。

  ```
  SELECT pg_terminate_backend(pid);
  ```

- トランザクション率が非常に高いシステムで pg_repack's ログテーブルから蓄積された変更を適用すると、適用プロセスが変更の速度に対して遅れる可能性があります。このような場合、pg_repack は適用プロセスを完了できません。詳細については、「再パック中の新しいテーブルのモニタリング」を参照してください。インデックスが著しく肥大化している場合、代替の解決策は、インデックスのみの再パックを実行することです。これにより、VACUUM のインデックスクリーンアップサイクルをより速く完了させることもできます。

  PostgreSQL バージョン 12 の手動 VACUUM を使用してインデックスのクリーンアップフェーズをスキップできます。また、PostgreSQL バージョン 14 の緊急自動バキューム中は自動的にスキップされます。これにより、VACUUM はインデックスの肥大化を取り除くことなくより迅速に完了します。これは、循環 VACUUM の防止などの緊急時にのみ使用されます。詳細については、Amazon Aurora ユーザーガイドの「インデックスの肥大化の回避」を参照してください。

前提条件

- テーブルには、PRIMARY KEY 制約または null 以外の UNIQUE 制約が必要です。
- 拡張機能のバージョンは、クライアントとサーバーの両方で同じである必要があります。
- RDS インスタンスに、肥大化がないテーブルの合計サイズ以上の FreeStorageSpace があることを確認します。例として、TOAST とインデックスを含むテーブルの合計サイズが 2TB で、テーブルの肥大化の合計が 1TB であるとします。必須の FreeStorageSpace は、次の計算によって返される値よりも大きくなければなりません。

  ```
  2TB (Table size) - 1TB (Table bloat) = 1TB
  ```

  次のクエリを使用してテーブルの合計サイズを確認し、pgstattuple を使用して肥大化を導き出すことができます。詳細については、Amazon Aurora ユーザーガイドの「テーブルとインデックスの肥大化の診断」を参照してください。

```
SELECT pg_size_pretty(pg_total_relation_size('table_name')) AS total_table_size;
```

このスペースは、アクティビティの完了後に再利用されます。

- RDS インスタンスに再パックオペレーションを処理するのに十分なコンピューティング容量と IO 容量があることを確認します。パフォーマンスのバランスを最適化するために、インスタンスクラスをスケールアップすることを検討してください。

pg_repack 拡張機能を使用するには

1. 次のコマンドを実行して、RDS for PostgreSQL DB インスタンスに pg_repack エクステンションをインストールします。

```
CREATE EXTENSION pg_repack;
```

2. 次のコマンドを実行して、pg_repack によって作成されたテンポラリログテーブルへの書き込みアクセス権を付与します。

```
ALTER DEFAULT PRIVILEGES IN SCHEMA repack GRANT INSERT ON TABLES TO PUBLIC;
ALTER DEFAULT PRIVILEGES IN SCHEMA repack GRANT USAGE, SELECT ON SEQUENCES TO
 PUBLIC;
```

3. pg_repack クライアントユーティリティを使用してデータベースに接続します。rds_superuser 権限を持つアカウントを使用します。例として、rds_test ロールに rds_superuser 権限があるとします。次の構文は、postgres データベース内のすべてのテーブルインデックスを含む完全なテーブルに対して pg_repack を実行します。

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 -k postgres
```

> **Note**
>
> -k オプションを使用して接続する必要があります。-a オプションはサポートされていません。

pg_repack クライアントからのレスポンスにより、再パッケージされる DB インスタンスのテーブルに関する情報が提供されます。

```
INFO: repacking table "pgbench_tellers"
INFO: repacking table "pgbench_accounts"
INFO: repacking table "pgbench_branches"
```

4. 次の構文は、postgres データベース内のインデックスを含む単一のテーブル orders を再パックします。

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 --table orders -k postgres
```

次の構文では、postgres データベース内の orders テーブルのインデックスのみを再パックします。

```
pg_repack -h db-instance-name.111122223333.aws-region.rds.amazonaws.com -U rds_test
 --table orders --only-indexes -k postgres
```

再パック中の新しいテーブルのモニタリング

- データベースのサイズは、再パックのスワップアンドドロップフェーズまで、テーブルの合計サイズから肥大化を引いた数だけ増加します。データベースサイズの増加率をモニタリングし、再パックの速度を計算して、最初のデータ転送の完了にかかる時間を概算で見積もることができます。

  例えば、テーブルの合計サイズを 2TB、データベースのサイズを 4TB、テーブルの合計肥大化を 1TB とします。再パックオペレーションの最後に計算によって返されるデータベースの合計サイズ値は次のとおりです。

  2TB (Table size) + 4 TB (Database size) - 1TB (Table bloat) = 5TB

  再パックオペレーションの速度を概算で見積もるには、2 つの時点の間の増加率をバイト単位でサンプリングします。増加率が 1GB の場合、最初のテーブル構築オペレーションが完了するまでに 1000 分または 16.6 時間かかることがあります。最初のテーブル構築に加えて、pg_repack は蓄積された変更を適用する必要があります。所要時間は、進行中の変更と蓄積された変更の適用速度によって異なります。

> **Note**
>
> pgstattuple 拡張機能を使用して、テーブルの肥大化を計算できます。詳細については、「pgstattuple」を参照してください。

- 再パックスキーマの下の pg_repack's ログテーブルの行数は、最初のロード後に新しいテーブルに適用される保留中の変更の量を表します。

  pg_stat_all_tables の pg_repack's ログテーブルをチェックして、新しいテーブルに適用される変更をモニタリングできます。pg_stat_all_tables.n_live_tup は、新しいテーブルに適用される保留中のレコードの数を示します。詳細については、「pg_stat_all_tables」を参照してください。

```
postgres=>SELECT relname,n_live_tup FROM pg_stat_all_tables WHERE schemaname =
 'repack' AND relname ILIKE '%log%';

-[ RECORD 1 ]---------
relname    | log_16490
n_live_tup | 2000000
```

- pg_stat_statements 拡張機能を使用して、再パックオペレーションの各ステップにかかる時間を調べることができます。これは、本番環境で同じ再パックオペレーションを適用する準備に役立ちます。出力をさらに拡張するように LIMIT 句を調整できます。

```
postgres=>SELECT
     SUBSTR(query, 1, 100) query,
     round((round(total_exec_time::numeric, 6) / 1000 / 60),4)
 total_exec_time_in_minutes
 FROM
     pg_stat_statements
 WHERE
     query ILIKE '%repack%'
 ORDER BY
     total_exec_time DESC LIMIT 5;

 query                                                                 |
 total_exec_time_in_minutes
-----------------------------------------------------------------------
+---------------------------
```

```
 CREATE UNIQUE INDEX index_16493 ON repack.table_16490 USING btree (a) |
        6.8627
 INSERT INTO repack.table_16490 SELECT a FROM ONLY public.t1           |
        6.4150
 SELECT repack.repack_apply($1, $2, $3, $4, $5, $6)                    |
        0.5395
 SELECT repack.repack_drop($1, $2)                                     |
        0.0004
 SELECT repack.repack_swap($1)                                         |
        0.0004
(5 rows)
```

再パックは完全にアウトオブプレースオペレーションであるため、元のテーブルは影響を受けず、元のテーブルの復元を必要とする予期しない課題は予想されません。再パックが予期せず失敗した場合は、エラーの原因を調べて解決する必要があります。

問題が解決したら、テーブルが存在するデータベースに pg_repack 拡張機能を削除して再作成し、pg_repack ステップを再試行してください。さらに、コンピューティングリソースの可用性とテーブルの同時アクセシビリティは、再パックオペレーションをタイムリーに完了させる上で重要な役割を果たします。

PLV8 拡張機能のアップグレードおよび使用

PLV8 は、信頼できる JavaScript 言語の PostgreSQL 用エクステンションです。ストアドプロシージャ、トリガー、SQL から呼び出し可能なその他のプロシージャルコードに使用できます。この言語のエクステンションは、PostgreSQL のすべての最新リリースでサポートされています。

PLV8 を使用しており、PostgreSQL を新しい PLV8 バージョンにアップグレードする場合は、新しいエクステンションをすぐに利用できるようになります。次のステップを実行して、カタログメタデータを PLV8 の新しいバージョンと同期させます。これらの手順はオプションですが、メタデータ不一致の警告を回避するために実行することを強くお勧めします。

アップグレードプロセスでは、既存の PLV8 機能がすべて削除されます。そのため、アップグレードする前に、RDS for PostgreSQL DB インスタンスのスナップショットを作成しておくことをお勧めします。詳細については、「Amazon RDS のシングル AZ DB インスタンスの DB スナップショットの作成」を参照してください。

> **Important**
>
> PostgreSQL バージョン 18 以降の Amazon RDS for PostgreSQL では、`plcoffee` および `plls` の PostgreSQL 拡張機能が廃止されます。エンジンバージョンがアップグレードされる場合に備えて、今後はアプリケーションで CoffeeScript や LiveScript を使用しないことをお勧めします。

カタログメタデータを新しいバージョンの PLV8 と同期させるには

1. 更新する必要があることを確認します。そのためには、インスタンスに接続されている間に以下のコマンドを実行します。

   ```
   SELECT * FROM pg_available_extensions WHERE name IN ('plv8','plls','plcoffee');
   ```

   インストールされているバージョンとしてデフォルトのバージョンより低いバージョンが表示された場合は、この手順を実行して、エクステンションを更新する必要があります。例えば、以下の結果セットは更新の必要があることを表します。

   ```
   name    | default_version | installed_version |                    comment
   --------+-----------------+-------------------
   +-------------------------------------------------
   ```

```
plls    | 2.1.0           | 1.5.3             | PL/LiveScript (v8) trusted
 procedural language
plcoffee| 2.1.0           | 1.5.3             | PL/CoffeeScript (v8) trusted
 procedural language
plv8    | 2.1.0           | 1.5.3             | PL/JavaScript (v8) trusted
 procedural language
(3 rows)
```

2. RDS for PostgreSQL DB インスタンスのスナップショットを作成していない場合は、作成してください。次のステップは、スナップショットの作成中も続行できます。
3. DB インスタンスの PLV8 関数の数を取得し、アップグレード後にすべて揃っていることを確認できるようにします。例えば次の SQL クエリ では、plv8、plcoffee、plls で記述されている関数の数が返ります。

```
SELECT proname, nspname, lanname
FROM pg_proc p, pg_language l, pg_namespace n
WHERE p.prolang = l.oid
AND n.oid = p.pronamespace
AND lanname IN ('plv8','plcoffee','plls');
```

4. pg_dump を使用して、スキーマのみのダンプファイルを作成します。例えば、クライアントマシンの /tmp ディレクトリに、ファイルを作成します。

```
./pg_dump -Fc --schema-only -U master postgres >/tmp/test.dmp
```

この例では、以下のオプションを使用します。

- -Fc – カスタム形式
- -- スキーマのみ – スキーマの作成に必要なコマンド (ここでは関数) のみをダンプする
- -U – RDS マスターユーザー名
- database – DB インスタンスのデータベース名

pg_dump の詳細については、「PostgreSQL ドキュメント」の「pg_dump」を参照してください。

5. ダンプファイルに存在する "CREATE FUNCTION" DDL ステートメントを抽出します。次の例では grep コマンドを実行して、関数を作成する DDL ステートメントを抽出し、ファイルに保存します。この ddl は後続のステップで関数を再作成するために使用します。

```
./pg_restore -l /tmp/test.dmp | grep FUNCTION > /tmp/function_list
```

pg_restore の詳細については、「PostgreSQL ドキュメント」の「pg_restore」を参照してください。

6. 関数およびエクステンションを削除します。次の例では、PLV8 ベースのオブジェクトを削除します。CASCADE オプションでは、すべての依存が削除されます。

```
DROP EXTENSION plv8 CASCADE;
```

plcoffee または plls に基づくオブジェクトが PostgreSQL インスタンスに含まれている場合は、それらのエクステンションに対してこのステップを繰り返します。

7. エクステンションを作成します。次の例では、plv8、plcoffee、plls のエクステンションが作成されます。

```
CREATE EXTENSION plv8;
CREATE EXTENSION plcoffee;
CREATE EXTENSION plls;
```

8. ダンプファイルおよび "ドライバ" ファイルを使用して関数を作成します。

次の例では、前に抽出した関数が再作成されます。

```
./pg_restore -U master -d postgres -Fc -L /tmp/function_list /tmp/test.dmp
```

9. 次のクエリを使用して、すべての関数が再作成されたことを確認します。

```
SELECT * FROM pg_available_extensions WHERE name IN ('plv8','plls','plcoffee');
```

PLV8 バージョン 2 では、次の行が結果セットに追加されます。

```
    proname    |  nspname   | lanname
---------------+------------+----------
 plv8_version  | pg_catalog | plv8
```

PL/Rust を使って Rust 言語で PostgreSQL 関数を記述する

PL/Rust は、PostgreSQL のための信頼できる Rust 言語エクステンションです。ストアドプロシージャ、関数、SQL から呼び出し可能なその他のプロシージャルコードに使用できます。PL/Rust 言語拡張は次のバージョンで利用可能です。

- RDS for PostgreSQL 17.1 またはそれ以降の 17 バージョン
- RDS for PostgreSQL 16.1 またはそれ以降の 16 バージョン
- RDS for PostgreSQL 15.2-R2 またはそれ以降の 15 バージョン
- RDS for PostgreSQL 14.9 またはそれ以降の 14 バージョン
- RDS for PostgreSQL 13.12 またはそれ以降の 13 バージョン

詳細については、GitHub の「PL/Rust」を参照してください。

> **重要**
>
> PostgreSQL 18 以降、Amazon RDS for PostgreSQL は `plrust` 拡張機能をサポートしなくなりました。今後のエンジンバージョンにアップグレードできるように、アプリケーションで PL/Rust の使用を停止することをお勧めします。

トピック

- PL/Rust の設定
- PL/Rust を使った関数の作成
- PL/Rust の入ったクレートを使用する
- PL/Rust の制限事項

PL/Rust の設定

DB インスタンスに plrust 拡張機能をインストールするには、DBインスタンスに関連付けられた DB パラメータグループの `shared_preload_libraries` パラメータに plrust を追加します。plrust 拡張機能をインストールすると、関数を作成できます。

`shared_preload_libraries` パラメータを変更するには、DB インスタンスをカスタムパラメータグループに関連付ける必要があります。カスタム DB パラメータグループの作成については、「Amazon RDS のパラメータグループ」を参照してください。

plust 拡張機能は、AWS マネジメントコンソール または AWS CLI を使用してインストールできます。

以下のステップでは、DB インスタンスがカスタム DB パラメータグループに関連付けられていることを前提としています。

コンソール

plust 拡張機能を **shared_preload_libraries** パラメータにインストールする

rds_superuser グループ (ロール) のメンバーであるアカウントを使用して、次のステップを完了します。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. DB インスタンスの名前を選択して、その詳細を表示します。
4. DB インスタンスの [設定] タブを開き、DB インスタンスパラメータグループのリンクを探します。
5. リンクを選択して、DB クラスターに関連付けられたカスタムパラメータを開きます。
6. [パラメータ] 検索フィールドに、shared_pre を入力して **shared_preload_libraries** パラメータを検索します。
7. プロパティ値にアクセスするには、[Edit parameters] (パラメータの編集) を選択します。
8. [値] フィールドのリストに plrust を追加します。値のリスト内の項目を区切るにはカンマを使用します。
9. DB インスタンスを再起動して、shared_preload_libraries パラメータの変更を有効にします。最初の再起動が完了するまでにさらに時間がかかる場合があります。
10. インスタンスが使用可能になったら、plrust が初期化されていることを確認します。psql を使用して DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
```

出力は以下のようになります。

```
shared_preload_libraries
--------------------------
rdsutils,plrust
```

```
(1 row)
```

AWS CLI

shared_preload_libraries パラメータに pltrust 拡張機能をインストールする

`rds_superuser` グループ (ロール) のメンバーであるアカウントを使用して、次のステップを完了します。

1. `shared_preload_libraries` パラメータに plrust を追加するには、modify-db-parameter-group AWS CLI コマンドを使用します。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=plrust,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. reboot-db-instance AWS CLI コマンドを使用して DB インスタンスを再起動し、plrust ライブラリを初期化します。最初の再起動が完了するまでにさらに時間がかかる場合があります。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. インスタンスが使用可能になったら、plrust が初期化されていることを確認できます。`psql` を使用して DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
```

出力は以下のようになります。

```
shared_preload_libraries
--------------------------
rdsutils,plrust
(1 row)
```

PL/Rust を使った関数の作成

PL/Rust は関数を動的ライブラリとしてコンパイルし、ロードして実行します。

次の Rust 関数は、配列から複数を除外します。

```
postgres=> CREATE LANGUAGE plrust;
CREATE EXTENSION
```

```
CREATE OR REPLACE FUNCTION filter_multiples(a BIGINT[], multiple BIGINT) RETURNS
 BIGINT[]
    IMMUTABLE STRICT
    LANGUAGE PLRUST AS
$$
    Ok(Some(a.into_iter().filter(|x| x.unwrap() % multiple != 0).collect()))
$$;

WITH gen_values AS (
SELECT ARRAY(SELECT * FROM generate_series(1,100)) as arr)
SELECT filter_multiples(arr, 3)
from gen_values;
```

PL/Rust の入ったクレートを使用する

RDS for PostgreSQL バージョン 16.3-R2 以降、15.7-R2 以降の 15 バージョン、14.12-R2 以降の 14 バージョン、および 13.15-R2 以降の 13 バージョンでは、PL/Rust は追加のクレートをサポートしています。

- `url`
- `regex`
- `serde`
- `serde_json`

RDS for PostgreSQL バージョン 15.5-R2 以降、14.10-R2 以降の 14 バージョン、および 13.13-R2 以降の 13 バージョンでは、PL/Rust は 2 つの追加のクレートをサポートしています。

- `croaring-rs`
- `num-bigint`

Amazon RDS for PostgreSQL バージョン 15.4、14.9、13.12 以降、PL/Rust は、次のクレートをサポートします。

- `aes`
- `ctr`
- `rand`

これらのクレートではデフォルト機能のみがサポートされています。新しい RDS for PostgreSQL バージョンには、更新されたバージョンのクレートが含まれているため、古いバージョンのクレートはサポートされなくなる可能性があります。

メジャーバージョンアップグレードを行う際のベストプラクティスに従って、お使いの PL/Rust 関数が新しいメジャーバージョンと互換性があるかどうかをテストしてください。詳細については、「Amazon RDS ユーザーガイド」のブログ「Amazon RDS を PostgreSQL のメジャーバージョンとマイナーバージョンにアップグレードするためのベストプラクティス」と「Amazon RDS の PostgreSQL DB エンジンのアップグレード」を参照してください。

PL/Rust 関数を作成する際の依存関係の使用例については、「依存関係を使う」を参照してください。

PL/Rust の制限事項

デフォルトでは、データベースユーザーは PL/Rust を使用できません。PL/Rust へのアクセスを提供するには、rds_superuser 権限を持つユーザーとして接続し、次のコマンドを実行します。

```
postgres=> GRANT USAGE ON LANGUAGE PLRUST TO user;
```

PostGIS 拡張機能を使用した空間データの管理

PostGIS は PostgreSQL の拡張機能であり、空間情報の保存と管理に使用します。PostGIS の詳細については、「Postgis.net」を参照してください。

バージョン 10.5 以降の PostgreSQL では、PostGIS がマップボックスのベクトルタイルデータを操作するために使用する libprotobuf 1.3.0 ライブラリがサポートされています。

PostGIS 拡張機能のセットアップには、`rds_superuser` 権限が必要です。PostGIS 拡張機能と空間データを管理するためのユーザー (ロール) を作成することをお勧めします。PostGIS 拡張機能とその関連コンポーネントは PostgreSQL に数千もの関数を追加します。ユースケースに適している場合は、PostGIS エクステンションを独自のスキーマで作成することを検討してください。次の例は、拡張機能を独自のデータベースにインストールする方法を示していますが、これは必須ではありません。

トピック

- ステップ 1: PostGIS 拡張機能を管理するユーザー (ロール) を作成する
- ステップ 2: PostGIS エクステンションを読み込む
- ステップ 3: 拡張機能スキーマの所有権を移管する
- ステップ 4: PostGIS テーブルの所有権を移管する
- ステップ 5: エクステンションをテストする
- ステップ 6: PostGIS 拡張機能を更新する
- PostGIS 拡張バージョン
- PostGIS 2 から PostGIS 3 へのアップグレード

ステップ 1: PostGIS 拡張機能を管理するユーザー (ロール) を作成する

まず、`rds_superuser` 権限があるユーザーとして RDS for PostgreSQL DB インスタンスに接続します。インスタンスの設定時にデフォルトの名前を保持している場合は、次のように `postgres` として接続します。

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

PostGIS 拡張機能を管理する別のロール (ユーザー) を作成します。

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
```

```
CREATE ROLE
```

このロールに rds_superuser 権限を付与して、ロールが拡張機能をインストールできるようにします。

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

PostGIS アーティファクトに使用するデータベースを作成します。この手順は省略可能です。または、ユーザーデータベースに PostGIS 拡張機能用のスキーマを作成することもできますが、これも必須ではありません。

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

gis_admin に lab_gis データベース上のすべての特権を付与します。

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

セッションを終了し、RDS for PostgreSQL DB インスタンスに gis_admin として再接続します。

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

次の手順の説明に従って、拡張機能のセットアップを続けます。

ステップ 2: PostGIS エクステンションを読み込む

PostGIS 拡張機能には複数の関連する拡張機能があり、それらが連携することで地理空間機能を提供しています。ユースケースによっては、このステップで作成した拡張機能の一部が必要ない場合があります。

CREATE EXTENSION ステートメントを使用して PostGIS エクステンションをロードします。

```
CREATE EXTENSION postgis;
CREATE EXTENSION
```

```
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION
```

次の例に示されている SQL クエリを実行すると、拡張子とその所有者がリストアップされ、結果を確認することができます。

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

ステップ 3: 拡張機能スキーマの所有権を移管する

ALTER SCHEMA ステートメント使用して、gis_admin ロールにスキーマの所有権を移転します。

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

次の SQL クエリを実行して、所有権の変更を確認できます。または、psql コマンドラインの \dn メタコマンドを使用します。

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;

       List of schemas
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

ステップ 4: PostGIS テーブルの所有権を移管する

> **Note**
>
> PostGIS 関数の所有権を変更しないでください。PostGIS の適切な運用と今後のアップグレードでは、これらの関数が元の所有権を保持する必要があります。PostGIS アクセス許可の詳細については、「PostgreSQL Security」を参照してください。

次の関数を使用して、gis_admin ロールに PostGIS テーブルの所有権を移管します。psql プロンプトから次のステートメントを実行して関数を作成します。

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

続いて、次のクエリを実行して exec 関数を実行すると、ステートメントが実行されてアクセス許可が変更されます。

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
```

```
s;
```

ステップ 5: エクステンションをテストする

スキーマ名の指定を不要とするには、次のコマンドを使用して検索パスに tiger スキーマを追加します。

```
SET search_path=public,tiger;
SET
```

次の SELECT ステートメントを使用して、tiger スキーマをテストします。

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

この拡張機能の詳細については、PostGIS ドキュメントの「Tiger Geocoder」を参照してください。

次の topology ステートメントを使用して SELECT スキーマへのアクセスをテストします。これにより、createtopology 関数を呼び出して、指定された空間参照識別子 (26986) とデフォルトの許容誤差 (0.5) を持つ新しいトポロジーオブジェクト (my_new_topo) を登録します。詳細については、PostGIS ドキュメントの「CreateTopology」を参照してください。

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

ステップ 6: PostGIS 拡張機能を更新する

PostgreSQL の新しいリリースでは、それぞれのリリースと互換性のある 1 つまたは複数のバージョンの PostGIS 拡張機能をサポートしています。PostgreSQL エンジンを新しいバージョンにアップグレードしても、PostGIS 拡張機能は自動的にアップグレードされません。PostgreSQL エンジンをアップグレードする前に、通常 PostGIS を現在の PostgreSQL バージョンで使用可能な最新バージョンにアップグレードします。詳細については、「PostGIS 拡張バージョン」を参照してください。

PostgreSQL エンジンのアップグレード後、PostGIS 拡張機能を再度アップグレードして、新しくアップグレードした PostgreSQL エンジンバージョンでサポートされているバージョンにアップグレードします。PostgreSQL のアップグレードの詳細については、「RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法」を参照してください。

RDS for PostgreSQL DB インスタンスでは、利用可能な PostGIS 拡張機能のバージョンアップを常時確認できます。そうするには、以下のコマンドを実行します。この関数は、PostGIS 2.5.0 以降のバージョンで使用できます。

```
SELECT postGIS_extensions_upgrade();
```

アプリケーションが最新で PostGIS バージョンがサポートされていない場合でも、次のように、メジャーバージョンで使用できる古いバージョンの PostGIS をインストールできます。

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

古いバージョンから特定の PostGIS バージョンにアップグレードする場合は、次のコマンドも使用できます。

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

アップグレード前のバージョンによっては、この関数をもう一度実行する必要があります。初期に関数を実行した結果によって、追加のアップグレード関数が必要かどうかが決まります。例えば、PostGIS 2 から PostGIS 3 にアップグレードする場合がこれに該当します。(詳しくは、「PostGIS 2 から PostGIS 3 へのアップグレード」を参照してください。)

PostgreSQL エンジンのメジャーアップグレードの準備のためにこの拡張機能をアップグレードした場合は、他の準備作業を継続できます。詳細については、「RDS for PostgreSQL のメジャーバージョンアップグレードを実行する方法」を参照してください。

PostGIS 拡張バージョン

に記載されている PostGIS など、すべての拡張機能バージョンをインストールすることをお勧めします。「Amazon RDS for PostgreSQL リリースノート」の Amazon RDS for PostgreSQL の拡張バージョン。リリースで利用可能なバージョンのリストを取得するには、次のコマンドを使用します。

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

バージョン情報は、Amazon RDS for PostgreSQL リリースノートの次のセクションで確認できます。

- Amazon RDS でサポートされる PostgreSQL バージョン 16 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 15 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 14 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 13 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 12 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 11 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 10 の拡張機能
- Amazon RDS でサポートされる PostgreSQL バージョン 9.6.x の拡張機能

PostGIS 2 から PostGIS 3 へのアップグレード

バージョン 3.0 以降、PostGIS ラスター機能は別の postgis_raster という拡張機能になりました。この拡張機能には、独自のインストールとアップグレードパスがあります。これにより、ラスター画像処理に必要な多くの関数、データ型などのアーティファクトがコア postgis 拡張機能から削除されます。つまり、ユースケースにラスター処理が必要ない場合は、postgis_raster 拡張機能をインストールする必要はありません。

次のアップグレード例では、最初のアップグレードコマンドは、ラスター機能を postgis_raster 拡張機能に展開します。次に、postgis_raster を新しいバージョンにアップグレードするには 2 つ目のアップグレードコマンドが必要です。

PostGIS 2 から PostGIS 3 にアップグレードするには

1. お使いの の PostgreSQL バージョンで利用可能な PostGIS のデフォルトバージョンを確認します。RDS for PostgreSQL DB インスタンス。確認するために、以下のクエリを実行します。

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. 、RDS for PostgreSQL DB インスタンスの各データベースにインストールされている PostGIS のバージョンを確認します。つまり、各ユーザーデータベースを次のようにクエリします。

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                                Description
---------+---------+--------
+-------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

このようにデフォルトバージョン (PostGIS 3.1.4) とインストールされているバージョン (PostGIS 2.3.7) が一致しない場合は、PostGIS 拡張機能をアップグレードする必要があることになります。

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. 次のクエリを実行して、ラスター機能が独自のパッケージに組み込まれていることを確認します。

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
```

```
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

出力を確認すれば、バージョンの間にまだ差があることがわかります。PostGIS 関数はバージョン 3 (postgis-3) で、ラスター関数 (rtpostgis) はバージョン 2 (rtpostgis-2.3) です。アップグレードを完了するには、次のようにアップグレードコマンドを再度実行します。

```
postgres=> SELECT postgis_extensions_upgrade();
```

警告メッセージは無視しても問題ありません。次のクエリを再度実行して、アップグレードが完了していることを確認します。PostGIS と関連するすべての拡張機能に対してアップグレードが必要と表示されていなければ、アップグレードは完了です。

```
SELECT postgis_full_version();
```

4. 次のクエリを使用して、完了したアップグレードプロセスと個別にパッケージ化された拡張機能を確認し、それぞれのバージョンが一致していることを確認します。

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                                Description
----------------+---------+--------
+-------------------------------------------------------------------------
```

```
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

出力には、PostGIS 2 拡張機能が PostGIS 3 にアップグレードされ、postgis と現在は分離された postgis_raster 拡張機能の両方がバージョン 3.1.5 であることが表示されます。

このアップグレード完了後にラスター機能を使用する予定がない場合は、次のように拡張機能を削除できます。

```
DROP EXTENSION postgis_raster;
```

Amazon RDS for PostgreSQL でサポートされている外部データラッパーを使用する

外部データラッパー (FDW) は、外部データへのアクセスを提供する特定のタイプの拡張機能です。例えば、oracle_fdw 拡張機能を使用すると、 RDS for PostgreSQL DB クラスターが Oracle データベースと連動できるようになります。別の例では、PostgreSQL ネイティブの postgres_fdw 拡張機能を使用すると、RDS for PostgreSQL DB インスタンスの外部に置かれた PostgreSQL DB インスタンスに保存されているデータにアクセスできます。

以下で、PostgreSQL でサポートされている、いくつかの 外部データラッパーについての情報を確認できます。

トピック

- SQL を使用した DB ログのアクセスのための log_fdw 拡張機能の使用
- 外部データへのアクセスのための postgres_fdw 拡張機能の使用
- mysql_fdw 拡張機能による MySQL データベースの操作
- oracle_fdw 拡張機能による Oracle データベースの操作
- tds_fdw 拡張機能による SQL Server データベースの操作

SQL を使用した DB ログのアクセスのための log_fdw 拡張機能の使用

RDS for PostgreSQL DB インスタンスは、SQL インターフェイスを通じてデータベースエンジンのログにアクセスする際に使用できる、log_fdw 拡張機能をサポートしています。log_fdw エクステンションは、データベースログ用の外部テーブルの作成を容易にする 2 つの関数を提供します。

- list_postgres_log_files - データベースログディレクトリのファイルとファイルサイズ (バイト単位) を一覧表示します。
- create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text) - 現在のデータベースで指定されたファイルの外部テーブルを構築します。

log_fdw によって作成されたすべての関数は、rds_superuser によって所有されます。rds_superuser ロールのメンバーは、これらの関数へのアクセス権限を他のデータベースユーザーに付与することができます。

デフォルトでは、ログファイルは、log_destination パラメータで指定されたように、Amazon RDS によって stderr (標準エラー) 形式で生成されます。このパラメータには、stderr と csvlog (カンマ区切り値、CSV) の 2 つのオプションしかありません。パラメータに csvlog オプションを追加すると、Amazon RDS は stderr と csvlog 両方のログを生成します。これは DB クラスターのストレージ容量に影響を与える可能性があるため、ログ処理に影響を与える他のパラメータに注意する必要があります。詳細については、「ログの送信先の設定 (stderr、csvlog)」を参照してください。

csvlog ログを生成すること 1 つの利点は、log_fdw 拡張機能により、データが複数の列にきちんと分割された外部テーブルを構築できることです。これを行うには、インスタンスをカスタム DB パラメータグループに関連付けて、log_destination の設定を変更できるようにする必要があります。これを行う方法については、「RDS for PostgreSQL DB インスタンスでのパラメータの使用」を参照してください。

次の例では、log_destination パラメータに cvslog が含まれることを前提としています。

log_fdw 拡張を使用するには

1. log_fdw 拡張機能をインストールします。

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. 外部データラッパーとしてログサーバーを作成します。

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. ログファイルのリストからすべてを選択します。

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

   レスポンスの例を次に示します。

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
```

```
(4 rows)
```

4. 選択したファイルの、1 つの 'log_entry' 列でテーブルを作成します。

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

レスポンスでは、テーブルが存在しているということ以外の詳細を返しません。

```
-----------------------------------
(1 row)
```

5. ログファイルのサンプルを選択します。次のコードは、ログの時間とエラーメッセージの説明を取得します。

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

レスポンスの例を次に示します。

```
            log_time             |                                 message
---------------------------------
+------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

外部データへのアクセスのための postgres_fdw 拡張機能の使用

postgres_fdw 拡張を使用してリモートデータベースサーバーにあるテーブルのデータにアクセスできます。PostgreSQL DB インスタンスからリモート接続を設定すると、リードレプリカにもアクセスできます。

postgres_fdw を使用してリモートデータベースサーバーにアクセスするには

1. postgres_fdw 拡張をインストールします。

```
CREATE EXTENSION postgres_fdw;
```

2. CREATE SERVER を使用して外部データサーバーを作成します。

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. リモートサーバーで使用するロールを識別するためのユーザーマッピングを作成します。

> **Important**
>
> パスワードを編集してログに表示されないようにするには、セッションレベルで `log_statement=none` を設定します。パラメータレベルで設定しても、パスワードは編集されません。

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. リモートサーバーのテーブルにマッピングするテーブルを作成します。

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

mysql_fdw 拡張機能による MySQL データベースの操作

RDS for PostgreSQL DB インスタンスから MySQL 互換データベースにアクセスするには、`mysql_fdw` 拡張機能をインストールしそれを使用します。この外部データラッパーを使用すると、RDS for MySQL、Aurora MySQL、MariaDB、その他の MySQL 互換データベースを操作できます。RDS for PostgreSQL DB インスタンスから MySQL データベースへの接続は、クライアントと

サーバーの設定に応じて、ベストエフォートベースで暗号化されます。ただし、必要に応じて暗号化を強制できます。詳細については、「拡張機能で転送中の暗号化を使用する」を参照してください。

mysql_fdw 拡張機能は、Amazon RDS for PostgreSQL バージョン 14.2、13.6 以降のリリースでサポートされています。MySQL 互換データベースインスタンス上のテーブルに対する RDS for PostgreSQL DB での選択、挿入、更新、および削除をサポートします。

トピック

- mysql_fdw 拡張機能を使用するように RDS for PostgreSQL DB をセットアップする
- 例: RDS for PostgreSQL から RDS for MySQL データベースを操作する
- 拡張機能で転送中の暗号化を使用する

mysql_fdw 拡張機能を使用するように RDS for PostgreSQL DB をセットアップする

RDS for PostgreSQL DB インスタンスでの mysql_fdw 拡張機能のセットアップには、DB インスタンスでの拡張機能のロードと、MySQL DB インスタンスへの接続ポイントの作成が関係しています。このタスクでは、MySQL DB インスタンスに関する次の詳細が必要です。

- ホスト名またはエンドポイント。RDS for MySQL DB インスタンスの場合、コンソールを使用してエンドポイントを見つけることができます。[接続とセキュリティ] タブを選択し、[エンドポイントとポート] セクションを確認します。
- ポート番号。MySQL のデフォルトポート番号は 3306 です。
- データベースの名前 DB 識別子。

また、MySQL ポート 3306 のセキュリティグループまたはアクセスコントロールリスト (ACL) へのアクセスを提供する必要があります。RDS for PostgreSQL DB インスタンスと RDS for MySQL DB インスタンスの両方がポート 3306 にアクセスする必要があります。アクセスが正しく設定されていない場合、MySQL 互換テーブルに接続しようとすると、次のようなエラーメッセージが表示されます。

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-region.rds.amazonaws.com:3306' (110)
```

次の手順では、ユーザーが (rds_superuser アカウントとして) 外部サーバーを作成します。次に、外部サーバーへのアクセスを特定のユーザーに付与します。その後、これらのユーザーは、MySQL DB インスタンスを操作するための適切な MySQL ユーザーアカウントへの独自のマッピングを作成します。

mysql_fdw を使用して MySQL データベースサーバーにアクセスするには

1. rds_superuser ロールがあるアカウントを使用して PostgreSQL DB インスタンスを接続します。 RDS for PostgreSQL DB インスタンスの作成時にデフォルトを受け入れた場合、ユーザー名は postgres であり、psql コマンドラインツールを使用して次のように接続できます。

```
psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. 次のように mysql_fdw 拡張機能をインストールします。

```
postgres=> CREATE EXTENSION mysql_fdw;
CREATE EXTENSION
```

拡張機能が RDS for PostgreSQL DB インスタンスにインストールされたら、MySQL データベースへの接続を提供する外部サーバーをセットアップします。

外部サーバーを作成するには

RDS for PostgreSQL DB インスタンス でこれらのタスクを実行します。このステップは、rds_superuser特権 (postgres など) があるユーザーとして接続していることを前提としています。

1. RDS for PostgreSQL DB インスタンスで外部サーバーを作成します。

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-
name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. 適切なユーザーに外部サーバーへのアクセスを付与します。これらは、管理者以外のユーザー、つまり、rds_superuser ロールのないユーザーである必要があります。

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

PostgreSQL ユーザーは、外部サーバーを介して MySQL データベースへの独自の接続を作成し、管理します。

例: RDS for PostgreSQL から RDS for MySQL データベースを操作する

RDS for PostgreSQL DB インスタンスにシンプルなテーブルがあると仮定します。RDS for PostgreSQL ユーザーが、そのテーブルで (SELECT)、INSERT、UPDATE、DELETE の項目をクエリしたいと思っています。mysql_fdw 拡張機能は、前の手順で詳述されているように、RDS for PostgreSQL DB インスタンスで作成された、と仮定します。rds_superuser 権限のあるユーザーとして RDS for PostgreSQL DB インスタンスに接続した後、次の手順に進むことができます。

1. RDS for PostgreSQL DB インスタンスで外部サーバーを作成します。

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-
DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. rds_superuser の許可を持たないユーザーに、(例えば user1 として) 使用を許可します。

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. *user1* として接続し、MySQL ユーザーへのマッピングを作成します。

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. MySQL テーブルにリンクされた外部テーブルを作成します。

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. 外部テーブルに対して単純なクエリを実行します。

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. MySQL テーブルでのデータの追加、変更、削除を行うことができます。例:

```
test=> INSERT INTO mytab values (2, 'mango');
```

```
INSERT 0 1
```

SELECT クエリをもう一度実行して、結果を確認します。

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

拡張機能で転送中の暗号化を使用する

RDS for PostgreSQL から MySQL への接続は、デフォルトで転送中の暗号化 (TLS/SSL) を使用します。ただし、クライアントとサーバーの設定が異なる場合、接続は暗号化されていない状態に戻ります。RDS for MySQL ユーザアカウントの REQUIRE SSL オプションを指定して、すべての発信接続に対して暗号化を適用できます。この同じアプローチは MariaDB および Aurora MySQL ユーザーアカウントでも機能します。

REQUIRE SSL に構成された MySQL ユーザーアカウントの場合、安全な接続を確立できないと接続の試行は失敗します。

既存の MySQL データベースユーザーアカウントの暗号化を強制するには、ALTER USER コマンドを使用できます。次の表に示すとおり、構文は MySQL のバージョンによって異なります。詳細については、MySQL リファレンスマニュアルの ALTER USER を参照してください。

MySQL 5.7、MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

mysql_fdw 拡張機能の詳細については、mysql_fdw ドキュメントをご覧ください。

oracle_fdw 拡張機能による Oracle データベースの操作

RDS for PostgreSQL DB インスタンス から Oracle データベースにアクセスするには、oracle_fdw 拡張機能をインストールして、使用します。この拡張機能は、Oracle データベース用の外部データラッパーです。この拡張機能の詳細については、oracle_fdw のドキュメントを参照してください。

oracle_fdw 拡張機能は、RDS for PostgreSQL のバージョン 12.7、13.3 以上のバージョンでサポートされています。

トピック

- oracle_fdw 拡張機能の有効化
- 例: Amazon RDS for Oracle Database にリンクされた外部サーバーの使用
- 転送時の暗号化の使用
- pg_user_mappings のビューおよび許可を理解する

oracle_fdw 拡張機能の有効化

oracle_fdw 拡張機能を使用するには、以下の手順を実行します。

oracle_fdw 拡張機能を有効化するには

- rds_superuser のアクセス許可を持つアカウントを使用して、次のコマンドを実行します。

```
CREATE EXTENSION oracle_fdw;
```

例: Amazon RDS for Oracle Database にリンクされた外部サーバーの使用

以下は、Amazon RDS for Oracle のデータベースにリンクされた外部サーバーの使用例です。

RDS for Oracle データベースにリンクされた外部サーバーを作成するには

1. RDS for Oracle DB インスタンスの以下の点を書き留めます。
   - エンドポイント
   - ポート
   - データベース名
2. 外部サーバーを作成します。

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. rds_superuser の権限を持たないユーザーに、(例えば user1 として) 使用を許可します。

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. user1 として接続し、Oracle ユーザーへのマッピングを作成します。

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Oracle テーブルにリンクされた外部テーブルを作成します。

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. 外部テーブルに対しクエリを実行します。

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

クエリで次のエラーが報告された場合は、セキュリティグループとアクセスコントロールリストをチェックして、両方のインスタンス間で通信が可能なことを確認します。

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

転送時の暗号化の使用

PostgreSQL から Oracle への転送時における暗号化は、クライアントとサーバーの設定パラメータの組み合わせに基づき構成されます。Oracle 21c の使用例については、Oracle ドキュメントの「About the Values for Negotiating Encryption and Integrity」を参照してください。Amazon RDS で oracle_fdw 用に使用されるクライアントは、ACCEPTED に設定されています。つまり、暗号化は Oracle データベースサーバーの設定に依存しており、暗号化には Oracle Security Library (libnnz) が使用されます。

データベースが RDS for Oracle 上にある場合の暗号化の設定については、「Oracle ネイティブネットワーク暗号化」を参照してください。

pg_user_mappings のビューおよび許可を理解する

PostgreSQL カタログ pg_user_mapping は、RDS for PostgreSQL ユーザーからのマッピングを外部データ (リモート) サーバー上のユーザーに保存します。カタログへのアクセスは制限されていますが、pg_user_mappings ビューをクリックすると、マッピングが表示されます。以下に、Oracle データベースの例で許可がどのように適用されるかを示す例がありますが、この情報は一般的に外部データラッパーに適用されます。

次の出力では、ロールとアクセス許可が、3 つの異なるサンプルユーザーにマップされていることが示されています。ここで、ユーザー rdssu1 と rdssu2 は rds_superuser ロールのメンバーであり、user1 はメンバーではありません。この例では、psql メタコマンド \du を使用して、既存のロールを一覧表示します。

```
test=>  \du
                                                                 List of roles
    Role name    |                            Attributes                            |
                 Member of
-----------------+------------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                                  |
 {rds_superuser}
 rdssu2          |                                                                  |
 {rds_superuser}
 user1           |                                                                  | {}
```

すべてのユーザー (rds_superuser 権限を持っているユーザーを含む) は、pg_user_mappings テーブルで独自のユーザーマッピング (umoptions) を表示することが許可されています。次の例に示すように、rdssu1 がすべてのユーザーマッピングを取得しようとすると、rdssu1rds_superuser 権限があっても、次のエラーが発生します。

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

次に例をいくつか示します。

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
```

```
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

information_schema._pg_user_mappings と pg_catalog.pg_user_mappings の間に実装上の違いがあるため、手動で作成された rds_superuser が pg_catalog.pg_user_mappings 内のパスワードを表示する場合には、追加のアクセス許可が必要となります。

rds_superuser が information_schema._pg_user_mappings 内のパスワードを表示する際には、追加のアクセス許可は必要ありません。

rds_superuser ロールを持たないユーザーの場合、以下の条件の下でのみ、pg_user_mappings 内のパスワードを表示できます。

- 現在のユーザーはマップされているユーザーであり、サーバーの所有者であるか、そのサーバーに対する USAGE 権限を保持しています。
- 現在のユーザーはサーバーの所有者であり、マッピングは PUBLIC となっています。

tds_fdw 拡張機能による SQL Server データベースの操作

PostgreSQL `tds_fdw` 拡張機能を使用して、Sybase や Microsoft SQL Server データベースなど、表形式データストリーム (TDS) プロトコルをサポートするデータベースにアクセスできます。この外部データラッパーを使用すると、RDS for PostgreSQL DB インスタンス を、Amazon RDS for Microsoft SQL Server を含む、TDS プロトコルを使用するデータベースに接続できます。詳細については、GitHub にある tds-fdw/tds_fdw に関するドキュメントを参照してください。

`tds_fdw` 拡張機能は、Amazon RDS for PostgreSQL のバージョン 14.2、13.6、およびそれ以降のリリースでサポートされています。

tds_fdw 拡張機能を使用するように Aurora PostgreSQL DB をセットアップする

次の手順では、`tds_fdw` をセットアップして、RDS for PostgreSQL DB インスタンスと使用する例を示します。`tds_fdw` を使用して SQL Server データベースに接続する前に、インスタンスの次の詳細を取得する必要があります。

- ホスト名またはエンドポイント。RDS for SQL Server DB インスタンスの場合、コンソールを使用してエンドポイントを見つけることができます。[Connectivity & security] (接続とセキュリティ) タブを選択し、[Endpoint and port] (エンドポイントとポート) セクションを確認します。
- ポート番号。Microsoft SQL Server のデフォルトポート番号は 1433 です。
- データベースの名前 DB 識別子。

また、SQL Server ポート、1433 のセキュリティグループまたはアクセスコントロールリスト (ACL) でのアクセスを提供する必要があります。RDS for PostgreSQL DB インスタンスと RDS for SQL Server DB インスタンスの両方が、ポート 1433 にアクセスする必要があります。アクセスが正しく設定されていない場合、Microsoft SQL Server をクエリしようとすると、次のエラーメッセージが表示されます。

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

tds_fdw を使用して SQL Server データベースに接続するには

1. `rds_superuser` ロールがあるアカウントを使用して、PostgreSQL DB インスタンスに接続します。

```
psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
username=test --password
```

2. tds_fdw 拡張機能をインストールします。

```
test=> CREATE EXTENSION tds_fdw;
CREATE EXTENSION
```

RDS for PostgreSQL DB インスタンスに拡張機能をインストールした後、外部サーバーをセットアップします。

外部サーバーを作成するには

rds_superuser 権限があるアカウントを使用する RDS for PostgreSQL DB インスタンスで次のタスクを実行します。

1. RDS for PostgreSQL DB インスタンスで外部サーバーを作成します。

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

   SQLServer 側で非 ASCII データにアクセスするには、RDS for PostgreSQL DB インスタンスの character_set オプションを使用してサーバーリンクを作成します。

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. rds_superuser ロール権限を持たないユーザーに、(例えば user1 として) 許可を付与します。

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. user1 として接続し、SQL Server ユーザーへのマッピングを作成します。

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. SQL Server テーブルにリンクされた外部テーブルを作成します。

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. 外部テーブルに対しクエリを実行します。

```
test=> SELECT * FROM mytab;
 a
---
 1
(1 row)
```

接続に転送中の暗号化を使用する

RDS for PostgreSQL から SQL Server への接続には、SQL Server のデータベース設定に応じて、転送中の暗号化 (TLS/SSL) を使用します。SQL Server が暗号化用に設定されていない場合、SQL Server データベースへの要求を行う RDS for PostgreSQL クライアントは、暗号化されていない状態に戻ります。

`rds.force_ssl` パラメータを設定して、RDS for SQL Server DB インスタンスへの接続に暗号化を強制できます。この方法については、「DB インスタンスへの接続に SSL を使用させる」を参照してください。RDS for SQL Server での SSL/TLS 設定の詳細については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。

Trusted Language Extensions for PostgreSQL を使用した操作

Trusted Language Extensions for PostgreSQL は PostgreSQL 拡張機能を構築するためのオープンソース開発キットです。これにより、高性能の PostgreSQL 拡張機能を構築し、それらを RDS for PostgreSQL DB インスタンス。PostgreSQL の Trusted Language Extensions (TLE) を使用することで、PostgreSQL の機能を拡張する文書化されたアプローチに従った PostgreSQL 拡張機能を作成できます。詳細については、PostgreSQL ドキュメントの「エクステンションへの関連オブジェクトのパッケージ化」を参照してください。

TLE の主な利点の 1 つは、PostgreSQL インスタンスの基盤となるファイルシステムへのアクセスを提供しない環境で使用できることです。以前は、新しい拡張機能をインストールするにはファイルシステムへのアクセスが必要でした。TLE ではこの制約がありません。で実行されているものを含め、あらゆる PostgreSQL データベース用の新しい拡張機能を作成するための開発環境を提供します。RDS for PostgreSQL DB インスタンス

TLE は、TLE を使用して作成する拡張機能の危険なリソースへのアクセスを防ぐように設計されています。そのランタイム環境では、拡張機能の不具合による影響は 1 つのデータベース接続に限定されます。また、TLE では、データベース管理者が拡張機能をインストールできるユーザーをきめ細かく制御でき、拡張機能を実行するためのアクセス許可モデルも用意されています。

TLE は、以下の RDS for PostgreSQL バージョンでサポートされています。

- バージョン 18.1 以降のバージョン 18
- バージョン 17.1 以降のバージョン 17
- バージョン 16.1 以降のバージョン 16
- バージョン 15.2 以降のバージョン 15
- バージョン 14.5 以降のバージョン 14
- バージョン 13.12 以降のバージョン 13

Trusted Language Extensions の開発環境とランタイムは、`pg_tle` PostgreSQL 拡張機能のバージョン 1.0.1 としてパッケージ化されています。JavaScript、Perl、PL/pgSQL、および SQL での拡張機能の作成をサポートしています。他の PostgreSQL 拡張機能をインストールするのと同じ方法で、 RDS for PostgreSQL DB インスタンスに `pg_tle` 拡張機能をインストールします。`pg_tle` をセットアップすると、開発者はこれを使用して TLE 拡張機能と呼ばれる新しい PostgreSQL 拡張機能を作成できます。

次のトピックでは、Trusted Language Extensions をセットアップする方法と、独自の TLE 拡張機能の作成を開始する方法について説明します。

トピック

- 用語
- Trusted Language Extensions for PostgreSQL を使用するための要件
- RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する
- Trusted Language Extensions for PostgreSQL の概要
- RDS for PostgreSQL の TLE 拡張機能の作成
- TLE 拡張機能をデータベースから削除する
- Trusted Language Extensions for PostgreSQL のアンインストール
- TLE 拡張機能で PostgreSQL フックを使用する
- TLE でのカスタムデータ型の使用
- Trusted Language Extensions for PostgreSQL の関数リファレンス
- Trusted Language Extensions for PostgreSQL のフックリファレンス

用語

Trusted Language Extensions の理解を深めるために、このトピックで使用されている用語については、次の用語集を参照してください。

Trusted Language Extensions for PostgreSQL

Trusted Language Extensions for PostgreSQL は、`pg_tle` 拡張機能としてパッケージされているオープンソース開発キットの正式名称です。これは、どの PostgreSQL システムでも使用できます。詳細については、GitHub の「aws/pg_tle」を参照してください。

Trusted Language Extensions

Trusted Language Extensions は、Trusted Language Extensions for PostgreSQL の省略名です。このドキュメントでは、この短縮名とその略称 (TLE) も使用されています。

信頼できる言語

信頼できる言語とは、特定のセキュリティ属性を持つプログラミング言語またはスクリプト言語です。例えば、信頼できる言語は通常、ファイルシステムへのアクセスを制限し、指定されたネットワークプロパティの使用を制限します。TLE 開発キットは、信頼できる言語をサポートするように設計されています。PostgreSQL は、信頼できる、または信頼できない拡張機能を作

成するために使用される複数の異なる言語をサポートしています。例については、PostgreSQL ドキュメントの「信頼できる PL/Perl と信頼できない PL/Perl」を参照してください。Trusted Language Extensions を使用して拡張機能を作成すると、その拡張機能は本質的に信頼できる言語メカニズムを使用します。

TLE 拡張機能

TLE 拡張機能は、Trusted Language Extensions (TLE) 開発キットを使用して作成された PostgreSQL 拡張機能です。

Trusted Language Extensions for PostgreSQL を使用するための要件

TLE 開発キットをセットアップして使用するための要件は次のとおりです。

- RDS for PostgreSQL バージョン – Trusted Language Extensions は、 RDS for PostgreSQL バージョン 13.12 以降の 13 バージョン、14.5 以降の 14 バージョン、15.2 以降のバージョンでのみサポートされています。
  - RDS for PostgreSQL インスタンスをアップグレードする必要がある場合は、「」「RDS for PostgreSQL DB エンジンのアップグレード」を参照してください。
  - PostgreSQL を実行している Amazon RDS DB インスタンスをまだ持っていない場合は作成できます。詳細については、「」を参照してください。RDS for PostgreSQL DB インスタンスについては、「PostgreSQL DB インスタンスを作成して接続する」を参照してください。
- **rds_superuser** 権限が必要です - pg_tle 拡張機能をセットアップおよび設定するには、データベースユーザーロールに rds_superuser ロールのアクセス許可が必要です。デフォルトでは、このロールは を作成する postgres ユーザーに付与されます。RDS for PostgreSQL DB インスタンス。
- カスタム DB パラメータグループが必要です – RDS for PostgreSQL DB インスタンス には、カスタム DB パラメータグループを設定する必要があります。
  - RDS for PostgreSQL DB インスタンスがカスタム DB パラメータグループで構成されていない場合は、カスタム DB パラメータグループを作成して に関連付ける必要があります。RDS for PostgreSQL DB インスタンス。ステップの簡単な概要については、「カスタム DB パラメータグループの作成と適用」を参照してください。
  - RDS for PostgreSQL DB インスタンスが、カスタム DB パラメータグループを使用して既に設定されている場合は、Trusted Language Extensions をセットアップできます。詳細については、「 RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する」を参照してください。

カスタム DB パラメータグループの作成と適用

以下のステップを使用してカスタム DB パラメータグループを作成し、それを使用するように RDS for PostgreSQL DB インスタンスを設定します。

コンソール

カスタム DB パラメータグループを作成して、 RDS for PostgreSQL DB インスタンスで使用するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. Amazon RDS メニューから [Parameter groups] (パラメータグループ) を選択します。
3. [パラメータグループの作成] を選択します。
4. [Parameter group details] (パラメータグループの詳細) ページで、次の情報を入力します。
   - [Parameter group family] (パラメータグループファミリー) で、[postgres14.] を選択します
   - [Type] (タイプ) で、[DB Parameter Group] (DB パラメータグループ) を選択します。
   - [Group name] (グループ名) には、パラメータグループに操作の内容に合ったわかりやすい名前を付けます。
   - [Description] (説明) には、チームの他のメンバーが簡単に見つけられるように、わかりやすい説明を入力します。
5. [作成] を選択します。カスタム DB パラメータグループは AWS リージョン で作成されます。次のステップに従って、 RDS for PostgreSQL DB インスタンスを使用するように変更できるようになりました。
6. Amazon RDS メニューから [Databases] (データベース) を選択します。
7. 一覧から TLE で使用する RDS for PostgreSQL DB インスタンスを選択し、[Modify] (変更) を選択します。
8. DB インスタンス設定の変更ページで、追加設定セクションで [データベースオプション] (データベースオプション) を選択し、セレクターからカスタム DB パラメータグループを選択します。
9. [Continue] (続行) を選択して、変更を保存します。
10. [Apply immediately] (すぐに適用) を選択すると、引き続き RDS for PostgreSQL DB インスタンスを TLE を使用するようにセットアップできます。

Trusted Language Extensions のシステム設定を継続するには、「RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する」を参照してください。

DB パラメータグループについては、「Amazon RDS DB インスタンスの DB パラメータグループ」を参照してください。

AWS CLI

AWS CLI をデフォルト AWS リージョン に設定することで、CLI コマンドを使用するときに `--region` 引数を指定しなくても済みます。詳細については、AWS Command Line Interface ユーザーガイドの「設定の基本」を参照してください。

カスタム DB パラメータグループを作成して、 RDS for PostgreSQL DB インスタンスで使用するには

1. create-db-parameter-group AWS CLI コマンドを使用して、AWS リージョン の postgres14 をベースにしたカスタム DB パラメータグループを作成してください。

   Linux、macOS、Unix の場合:

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family postgres14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Windows の場合:

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family postgres14 ^
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   AWS リージョン でカスタム DB パラメータグループを使用できるため、 RDS for PostgreSQL DB インスタンスのライターインスタンスを変更してそれを使用できます。

2. modify-db-instance AWS CLI コマンドを使用して、カスタム DB パラメータグループを RDS for PostgreSQL DB インスタンス。このコマンドは、アクティブなインスタンスを直ちに再起動します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
  --region aws-region \
  --db-instance-identifier your-instance-name \
  --db-parameter-group-name custom-params-for-pg-tle \
  --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
  --region aws-region ^
  --db-instance-identifier your-instance-name ^
  --db-parameter-group-name custom-params-for-pg-tle ^
  --apply-immediately
```

Trusted Language Extensions のシステム設定を継続するには、「RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する」を参照してください。

詳細については、「Amazon RDS のパラメータグループ」を参照してください。

RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する

以下のステップでは、RDS for PostgreSQL DB インスタンスがカスタム DB パラメータグループに関連付けられていることを前提としています。これらの手順には、AWS マネジメントコンソール または AWS CLI を使用できます。

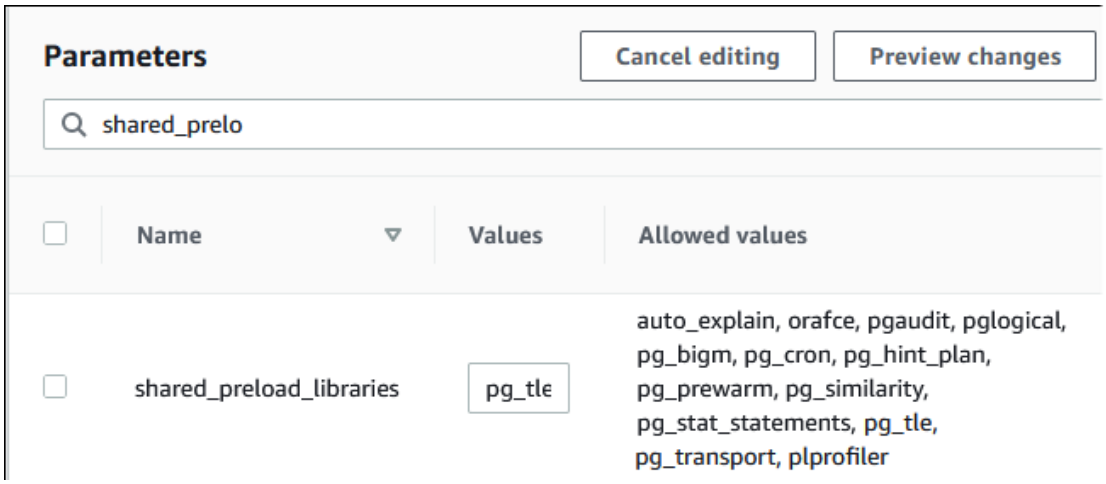
RDS for PostgreSQL DB インスタンスで信頼できる Trusted Language Extensions をセットアップする場合、そのデータベースに対するアクセス許可を持つデータベースユーザーが使用できるように、特定のデータベースにインストールします。

コンソール

Trusted Language Extensions をセットアップするには

`rds_superuser` グループ (ロール) のメンバーであるアカウントを使用して、次のステップを実行します。

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、 RDS for PostgreSQL DB インスタンスを選択します。
3. の [Configuration] (設定) タブを開きます。RDS for PostgreSQL DB インスタンス。インスタンスの詳細の中から、パラメータグループのリンクを見つけてください。
4. リンクを選択して、に関連するカスタムパラメータを開きます。RDS for PostgreSQL DB インスタンス。
5. パラメータ検索フィールドに、`shared_pre` を入力して `shared_preload_libraries` パラメータを検索します。
6. プロパティ値にアクセスするには、[Edit parameters] (パラメータの編集) を選択します。
7. [Values] (値) フィールドのリストに `pg_tle` を追加します。値のリスト内の項目を区切るにはカンマを使用します。



8. RDS for PostgreSQL DB instance を再起動して、`shared_preload_libraries` パラメータの変更を有効にします。
9. インスタンスが使用可能になったら、`pg_tle` が初期化されていることを確認します。`psql` を使用して RDS for PostgreSQL DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. `pg_tle` 拡張子を初期化すると、拡張機能を作成できるようになりました。

```
CREATE EXTENSION pg_tle;
```

以下の psql メタコマンドを使用して、拡張機能がインストールされていることを確認できます。

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+---------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. RDS for PostgreSQL DB インスタンスのセットアップ時に作成したプライマリユーザー名に pgtle_admin ロールを付与します。デフォルトを受け入れた場合は、postgres です。

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

次の例に示すように、psql メタコマンドを使用して、付与されたことを確認できます。出力には pgtle_admin と postgres ロールのみが表示されます。詳細については、「rds_superuser ロールを理解する」を参照してください。

```
labdb=> \du
                           List of roles
    Role name    |           Attributes            |               Member of
-----------------+---------------------------------
+-----------------------------------
pgtle_admin      | Cannot login                     | {}
postgres         | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                 | Password valid until infinity    |...
```

12. \q メタコマンドを使用して psql セッションを終了します。

```
\q
```

TLE 拡張機能の作成を開始するには、「例: SQL を使用した信頼できる言語拡張関数の作成」を参照してください。

AWS CLI

AWS CLI をデフォルト AWS リージョン に設定することで、CLI コマンドを使用するときに `--region` 引数を指定しなくても済みます。詳細については、AWS Command Line Interface ユーザーガイドの「設定の基本」を参照してください。

Trusted Language Extensions をセットアップするには

1. `shared_preload_libraries` パラメータに `pg_tle` を追加するには、modify-db-parameter-group AWS CLI コマンドを使用します。

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. reboot-db-instance AWS CLI コマンドを使用して、RDS for PostgreSQL DB インスタンスを再起動し、`pg_tle` ライブラリを初期化します。

```
aws rds reboot-db-instance \
    --db-instance-identifier your-instance \
    --region aws-region
```

3. インスタンスが使用可能になると、`pg_tle` が初期化されていることを確認できます。`psql` を使用して RDS for PostgreSQL DB インスタンスに接続し、次のコマンドを実行します。

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

   `pg_tle` を初期化すると、拡張機能を作成できるようになりました。

```
CREATE EXTENSION pg_tle;
```

4. RDS for PostgreSQL DB インスタンスのセットアップ時に作成したプライマリユーザー名に `pgtle_admin` ロールを付与します。デフォルトを受け入れた場合は、`postgres` です。

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

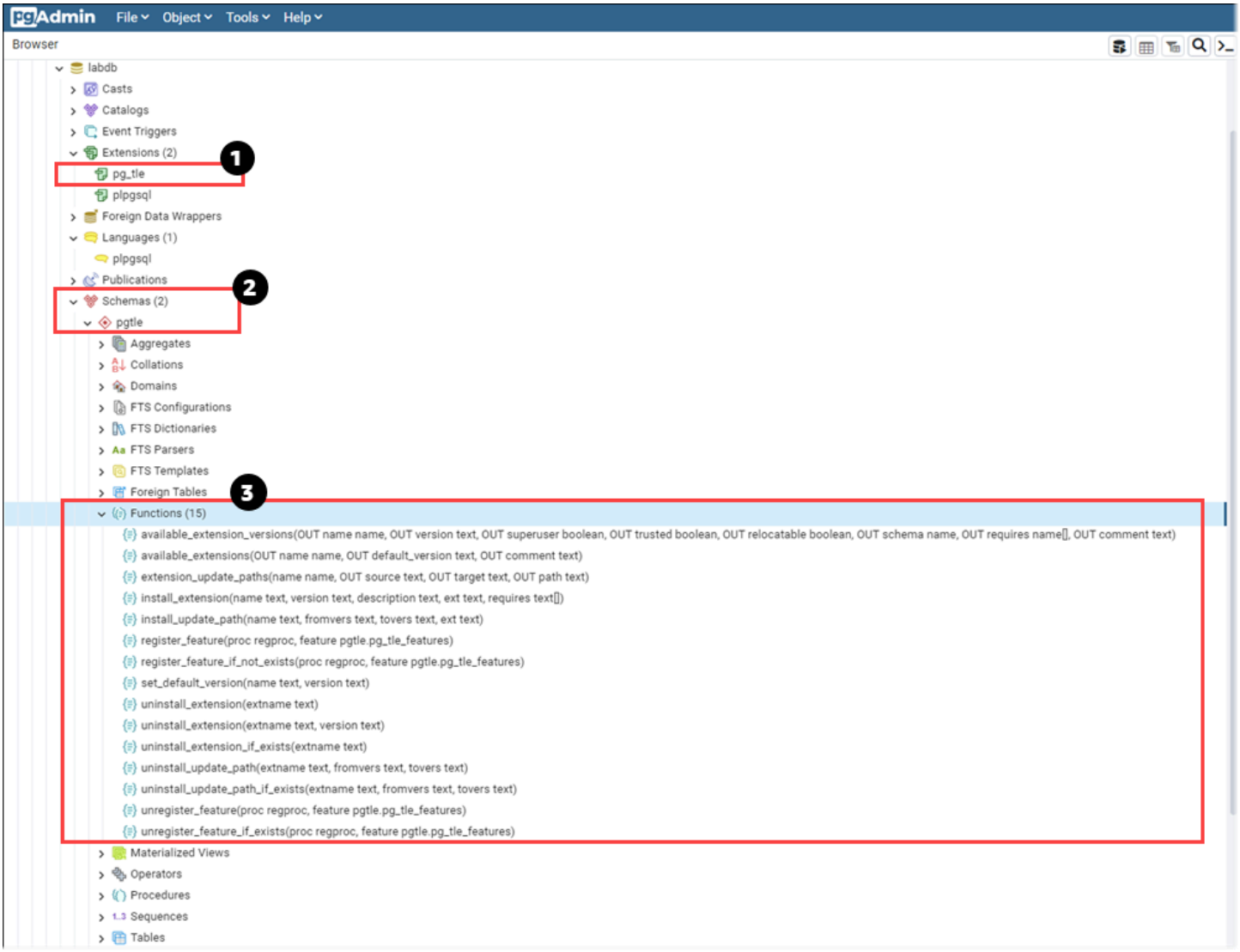
5. 以下のように psql セッションを終了します。

```
labdb=> \q
```

TLE 拡張機能の作成を開始するには、「例: SQL を使用した信頼できる言語拡張関数の作成」を参照してください。

Trusted Language Extensions for PostgreSQL の概要

Trusted Language Extensions for PostgreSQL は PostgreSQL 拡張機能で、他の PostgreSQL 拡張機能をセットアップするのと同じ方法で RDS for PostgreSQL DB インスタンスにインストールします。pgAdmin クライアントツールのサンプルデータベースの次の画像では、pg_tle 拡張機能を構成するコンポーネントの一部を確認できます。



以下の詳細を表示できます。

1. Trusted Language Extensions (TLE) for PostgreSQL 開発キットは `pg_tle` 拡張機能としてパッケージ化されています。そのため、`pg_tle` がインストールされているデータベースで使用可能な拡張機能にそれが追加されます。
2. TLE には独自のスキーマ、`pgtle` があります。このスキーマには、作成した拡張機能をインストールおよび管理するためのヘルパー関数 (3) が含まれています。
3. TLE には、拡張機能のインストール、登録、管理のためのヘルパー関数が十数種類用意されています。これらの関数の詳細については、「Trusted Language Extensions for PostgreSQL の関数リファレンス」を参照してください。

`pg_tle` 拡張機能のその他のコンポーネントには、以下のものが含まれています。

- **pgtle_admin** ロール – pgtle_admin ロールは、pg_tle 拡張機能のインストール時に作成されます。この役職には特権があり、そのように扱われる必要があります。データベースユーザーに pgtle_admin ロールを付与する場合は、最小特権の原則に従うことを強くお勧めします。つまり、postgres のような新しい TLE 拡張機能の作成、インストール、管理が許可されているデータベースユーザーにのみ pgtle_admin ロールを付与します。
- **pgtle.feature_info** テーブル – pgtle.feature_info テーブルは保護されたテーブルで、TLE、フック、およびそれらが使用するカスタムストアドプロシージャと関数に関する情報が含まれています。pgtle_admin権限がある場合は、以下の Trusted Language Extensions の関数を使用して、テーブル内の情報を追加および更新します。
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

RDS for PostgreSQL の TLE 拡張機能の作成

TLE を使用して作成した拡張機能は、pg_tle 拡張機能がインストールされている任意の RDS for PostgreSQL DB インスタンスにインストールできます。pg_tle 拡張機能のスコープは、インストールされている PostgreSQL データベースに限定されます。TLE を使用して作成した拡張機能は、同じデータベースを対象としています。

さまざまな pgtle 関数を使用して、TLE 拡張機能を構成するコードをインストールします。以下の Trusted Language Extensions 関数には pgtle_admin すべてロールが必要です。

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists

- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

例: SQL を使用した信頼できる言語拡張関数の作成

次の例は、さまざまな式を使用して距離を計算するためのいくつかの SQL 関数を含む pg_distance という名前の TLE 拡張機能を作成する方法を示しています。リストには、マンハッタン距離を計算する関数とユークリッド距離を計算する関数があります。これらの式の違いの詳細については、Wikipedia の「Taxicab geometry」と「Euclidean geometry」を参照してください。

RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する で説明されているように pg_tle 拡張機能をセットアップしていれば、この例を独自の RDS for PostgreSQL DB インスタンスで使用できます。

> **Note**
>
> この手順を実行するには、pgtle_admin ロールの権限が必要です。

サンプルの TLE 拡張機能を作成するには

以下の手順では、labdb という名前のサンプルデータベースを使用します。このデータベースは postgres プライマリユーザーが所有しています。postgres ロールには、pgtle_admin ロールのアクセス許可もあります。

1. psql を使用して、に接続します。RDS for PostgreSQL DB インスタンス。

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. 次のコードをコピーして psql セッションコンソールに貼り付けて、pg_distance という名前の TLE 拡張機能を作成します。

```
SELECT pgtle.install_extension
(
 'pg_distance',
 '0.1',
  'Distance functions for two points',
$_pg_tle_$
```

```
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

次のような出力が表示されます。

```
install_extension
---------------
 t
(1 row)
```

これで、pg_distance 拡張機能を構成するアーティファクトがデータベースにインストールされました。これらのアーティファクトには、コントロールファイルと拡張機能のコードが含まれます。これらは、CREATE EXTENSION コマンドを使用して拡張機能を作成するために必要となる項目です。つまり、データベースユーザーがその関数を利用できるようにするには、やはり拡張機能を作成する必要があります。

3. 拡張機能を作成するには、他の拡張機能と同じように CREATE EXTENSION コマンドを使用します。他の拡張機能と同様に、データベースユーザーにはデータベース内の CREATE アクセス許可が必要です。

```
CREATE EXTENSION pg_distance;
```

4. pg_distance TLE 拡張機能をテストするには、これを使用して 4 点間のマンハッタン距離を計算できます。

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

同じ点群間のユークリッド距離を計算するには、以下を使用できます。

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

pg_distance 拡張機能は関数をデータベースに読み込み、データベースに対するアクセス許可を持つすべてのユーザーがその関数を利用できるようにします。

TLE 拡張機能の変更

この TLE 拡張機能にパッケージされている関数のクエリパフォーマンスを向上させるには、次の 2 つの PostgreSQL 属性を仕様に追加してください。

- IMMUTABLE – IMMUTABLE 属性により、クエリオプティマイザが最適化を使用してクエリの応答時間を改善できるようになります。詳細については、PostgreSQL ドキュメントの「関数のボラティリティカテゴリ」を参照してください。
- PARALLEL SAFE – PARALLEL SAFE属性は、PostgreSQL が関数をパラレルモードで実行できるようにするもう 1 つの属性です。詳細については、PostgreSQL のドキュメントの「機能の作成」を参照してください。

次の例では、pgtle.install_update_path 関数を使用してこれらの属性を各関数に追加し、pg_distance TLE 拡張機能のバージョン 0.2 を作成する方法を確認できます。この関数の詳細については、「pgtle.install_update_path」を参照してください。このタスクを実行するには、pgtle_admin ロールが必要です。

既存の TLE 拡張機能を更新してデフォルトバージョンを指定するには

1. psql または pgAdmin などの別のクライアントツールを使用して、 RDS for PostgreSQL DB インスタンスのライターインスタンスに接続します。

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. 次のコードをコピーして psql セッションコンソールに貼り付けることで、既存の TLE 拡張機能を変更します。

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

次のようなレスポンスが表示されます。

```
install_update_path
---------------------
 t
(1 row)
```

このバージョンの拡張機能をデフォルトバージョンにすると、データベースユーザーがデータベースで拡張機能を作成または更新するときにバージョンを指定する必要がなくなります。

3. TLE 拡張機能の修正バージョン (バージョン 0.2) がデフォルトバージョンになるように指定するには、次の例に示す pgtle.set_default_version 関数を使用します。

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

この関数の詳細については、「pgtle.set_default_version」を参照してください。

4. コードを配置したら、次に示すように、ALTER EXTENSION ... UPDATE コマンドを使用して、インストールされている TLE 拡張機能を通常の方法で更新できます。

```
ALTER EXTENSION pg_distance UPDATE;
```

TLE 拡張機能をデータベースから削除する

TLE 拡張機能は、他の PostgreSQL 拡張機能の場合と同じように DROP EXTENSION コマンドを使用して削除できます。拡張機能を削除しても、拡張機能を構成するインストールファイルは削除されないため、ユーザーは拡張機能を再作成できます。拡張機能とそのインストールファイルを削除するには、次の 2 段階のプロセスを実行します。

TLE 拡張機能とそのインストールファイルを削除するには

1. psql または別のクライアントツールを使用して RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=dbname
```

2. PostgreSQL 拡張機能と同様に、この拡張機能を削除してください。

```
DROP EXTENSION your-TLE-extension
```

例えば、例: SQL を使用した信頼できる言語拡張関数の作成 で詳細を説明しているように pg_distance という拡張機能を作成する場合は、次のように拡張機能を削除できます。

```
DROP EXTENSION pg_distance;
```

次のように、拡張機能が削除されたことを確認する出力が表示されます。

```
DROP EXTENSION
```

この時点で、拡張機能はデータベースでアクティブではなくなります。ただし、インストールファイルとコントロールファイルはデータベースにまだ残っているため、データベースユーザーは必要に応じて拡張機能を再作成できます。

- 拡張ファイルをそのまま残して、データベースユーザーが TLE 拡張機能を作成できるようにする場合は、ここで終了してください。
- 拡張機能を占めるすべてのファイルを削除する場合は、次のステップに進みます。

3. 拡張機能のインストールファイルをすべて削除するには、pgtle.uninstall_extension 関数を使用してください。この関数は、拡張機能のコードとコントロールファイルをすべて削除します。

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

例えば、すべての pg_distance インストールファイルを削除するには、次のコマンドを使用します。

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Trusted Language Extensions for PostgreSQL のアンインストール

TLE を使用して独自の TLE 拡張機能を作成する必要がなくなった場合は、pg_tle 拡張機能を削除してすべてのアーティファクトを削除できます。このアクションには、データベース内のすべての TLE 拡張機能の削除と pgtle スキーマの削除が含まれます。

pg_tle 拡張機能とそのスキーマをデータベースから削除するには

1. psql または別のクライアントツールを使用して RDS for PostgreSQL DB インスタンスに接続します。

```
psql --host=.111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=dbname
```

2. pg_tle 拡張機能をデータベースから削除します。データベースに独自の TLE 拡張機能がまだデータベースで実行されている場合は、それらの拡張機能も削除する必要があります。そのためには、次に示すように、CASCADE キーワードを使用します。

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   pg_tle 拡張機能がデータベースでまだ有効になっていない場合は、CASCADE キーワードを使用する必要はありません。

3. pgtle スキーマを削除します。このアクションにより、データベースからすべての管理関数が削除されます。

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   このコマンドは、プロセスが完了すると、以下を返します。

   ```
   DROP SCHEMA
   ```

   pg_tle 拡張機能、そのスキーマ、関数、およびすべてのアーティファクトが削除されます。TLE を使用して新しい拡張機能を作成するには、セットアッププロセスをもう一度実行してください。詳細については、「RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する」を参照してください。

TLE 拡張機能で PostgreSQL フックを使用する

フックは PostgreSQL で利用できるコールバックメカニズムで、開発者は通常のデータベースオペレーション中にカスタム関数やその他のルーチンを呼び出すことができます。TLE 開発キットは PostgreSQL フックをサポートしているため、実行時にカスタム関数を PostgreSQL の動作と統合できます。例えば、フックを使用して認証プロセスを独自のカスタムコードに関連付けたり、特定のニーズに合わせてクエリの計画と実行プロセスを変更したりできます。

TLE 拡張機能にはフックを使用できます。フックの適用範囲がグローバルな場合、すべてのデータベースに適用されます。そのため、TLE 拡張機能がグローバルフックを使用している場合は、ユーザーがアクセスできるすべてのデータベースに TLE 拡張機能を作成する必要があります。

pg_tle 拡張機能を使用して独自の Trusted Language Extensions を構築する場合、SQL API の利用可能なフックを使用して拡張機能の関数を構築できます。すべてのフックを pg_tle に登録する必要があります。一部のフックでは、さまざまな設定パラメータを設定する必要がある場合もあります。例えば、passcode チェックフックをオン、オフ、または必須に設定できます。使用可能な pg_tle フックの特定の要件の詳細については、「Trusted Language Extensions for PostgreSQL のフックリファレンス」を参照してください。

例: PostgreSQL フックを使用する拡張機能の作成

このセクションで説明する例では、PostgreSQL フックを使用して特定の SQL のオペレーション中に入力されたパスワードをチェックし、データベースユーザーが自分のパスワードを password_check.bad_passwords テーブルに含まれるパスワードに設定できないようにします。この表には、一般的によく使用されているものの、簡単に破られてしまうパスワードの選択肢の上位 10 件が掲載されています。

この例を 、RDS for PostgreSQL DB インスタンスに設定するには、Trusted Language Extensions が既にインストールされている必要があります。詳細については、「 RDS for PostgreSQL DB インスタンスに Trusted Language Extensions を設定する」を参照してください。

パスワードチェックフックの例を設定するには

1. psql を使用して、に接続します。RDS for PostgreSQL DB インスタンス。

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. パスワードチェックフックコードリスト のコードをコピーし、データベースに貼り付けます。

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
```

```
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
```

```
);
```

拡張機能がデータベースに読み込まれると、次のような出力が表示されます。

```
 install_extension
-------------------
 t
(1 row)
```

3. データベースに接続したままで、拡張機能を作成できるようになりました。

```
CREATE EXTENSION my_password_check_rules;
```

4. 次の psql メタコマンドを使用して、拡張機能がデータベースに作成されたことを確認できます。

```
\dx
                        List of installed extensions
          Name           | Version |   Schema   |
 Description
-------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. 別のターミナルセッションを開いて、AWS CLI を操作します。パスワードチェックフックを有効にするには、カスタム DB パラメータグループを変更する必要があります。そのためには、以下の例に示すように modify-db-parameter-group CLI コマンドを使用します。

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

パラメータが正常に有効になると、次のような出力が表示されます。

```
(
    "DBParameterGroupName": "docs-lab-parameters-for-tle"
}
```

パラメータグループ設定の変更が適用されるまでには数分かかる場合があります。ただし、このパラメータは動的であるため、設定を有効にするために を再起動する必要はありません。

6. psql セッションを開き、データベースに問い合わせて password_check フックが有効になっていることを確認します。

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

パスワードチェックフックがアクティブになりました。次の例に示すように、新しいロールを作成し、不適切なパスワードを使用してテストできます。

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

出力は、読みやすい形式にしてあります。

次の例では、pgsql インタラクティブメタコマンドの \password 動作が password_check フックの影響を受けることを示しています。

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
```

```
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

この TLE 拡張機能モジュールを削除して、必要に応じてソースファイルをアンインストールできます。詳細については、「TLE 拡張機能をデータベースから削除する」を参照してください。

パスワードチェックフックコードリスト

ここに示すサンプルコードは、my_password_check_rules TLE 拡張機能の仕様を定義しています。このコードをコピーしてデータベースに貼り付けると、my_password_check_rules 拡張機能のコードがデータベースにロードされ、password_check フックが拡張機能で使用できるように登録されます。

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);
```

```
  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

TLE でのカスタムデータ型の使用

PostgreSQLは、データベース内の複雑なデータ構造を効率的に処理するための新しい基本型（スカラー型とも呼ばれます）を登録するコマンドをサポートしています。ベースタイプを使用すると、データを内部に保存する方法や、データを外部のテキスト表現に変換したり、外部のテキスト表現から変換したりする方法をカスタマイズできます。これらのカスタムデータ型は、数値やテキストなどの組み込み型では十分な検索セマンティクスを提供できない機能ドメインをサポートするためにPostgreSQL を拡張する場合に役立ちます。

RDS for PostgreSQL では、信頼できる言語拡張機能でカスタムデータ型を作成し、これらの新しいデータ型の SQL 操作とインデックス操作をサポートする関数を定義できます。カスタムデータ型は、以下のバージョンで利用できます。

- RDS for PostgreSQL 15.4 またはそれ以降の 15 バージョン
- RDS for PostgreSQL 14.9 またはそれ以降の 14 バージョン
- RDS for PostgreSQL 13.12 またはそれ以降の 13 バージョン

詳細については、「信頼言語ベースタイプ」を参照してください。

Trusted Language Extensions for PostgreSQL の関数リファレンス

Trusted Language Extensions for PostgreSQL で利用できる関数については、以下のリファレンスドキュメントを参照してください。これらの関数を使用して、TLE 拡張機能、つまり Trusted Language Extensions 開発キットを使用して開発した PostgreSQL 拡張機能のインストール、登録、更新、管理を行います。

関数

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

pgtle.available_extensions 関数は、集合を返す関数です。データベース内の使用可能なすべての TLE 拡張機能を返します。返される各行には、1 つの TLE 拡張機能に関する情報が含まれています。

関数プロトタイプ

```
pgtle.available_extensions()
```

ロール

なし。

引数

なし。

出力

- name – TLE 拡張機能の名前。
- default_version – バージョンを指定せずに CREATE EXTENSION が呼び出されたときに使用する TLE 拡張機能のバージョン。
- description – TLE 拡張機能に関するさらに詳細な説明。

使用例

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

available_extension_versions 関数は、集合を返す関数です。使用可能なすべての TLE 拡張機能とそのバージョンの一覧を返します。各行には、特定のロールが必要かどうかなど、特定の TLE 拡張機能のバージョンに関する情報が含まれています。

関数プロトタイプ

```
pgtle.available_extension_versions()
```

ロール

なし。

引数

なし。

出力

- name – TLE 拡張機能の名前。
- version – TLE 拡張機能のバージョン。
- superuser – この値は、TLE 拡張機能では常に false です。TLE 拡張機能の作成または更新に必要なアクセス許可は、特定のデータベースに他のオブジェクトを作成する場合と同じです。
- trusted – この値は、TLE 拡張機能では常に false です。
- relocatable – この値は、TLE 拡張機能では常に false です。
- schema – TLE 拡張機能がインストールされているスキーマの名前を指定します。
- requires – この TLE 拡張機能に必要な他の拡張機能の名前を含む配列。
- description – TLE 拡張機能の詳細な説明。

出力値の詳細については、PostgreSQL ドキュメントの [Extension > Extension Files] (拡張機能 > 拡張機能ファイル) にある「関連オブジェクトのパッケージ化」を参照してください。

使用例

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

extension_update_paths 関数は、集合を返す関数です。TLE 拡張機能で使用可能なすべての更新パスのリストを返します。各行には、その TLE 拡張機能で使用可能なアップグレードまたはダウングレードが含まれています。

関数プロトタイプ

```
pgtle.extension_update_paths(name)
```

ロール

なし。

引数

name – アップグレードパスを取得する TLE 拡張機能の名前。

出力

- source – 更新のソースバージョン。
- target – 更新のターゲットバージョン。
- path – TLE 拡張機能を source バージョンから target にアップグレードするために使用されるアップグレードパス (例えば 0.1--0.2)。

使用例

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

この install_extension 関数を使用すると、TLE 拡張機能を構成するアーティファクトをデータベースにインストールし、その後、CREATE EXTENSION コマンドを使用して作成できます。

関数プロトタイプ

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

ロール

なし。

引数

- name – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。
- version – TLE 拡張機能のバージョン。
- description – TLE 拡張機能に関する詳細な説明。この説明は、pgtle.available_extensions() の comment フィールドに表示されます。
- ext – TLE 拡張機能の内容。この値には、関数などのオブジェクトが含まれます。

- `requires` – この TLE 拡張機能の依存関係を指定するオプションパラメータ。`pg_tle` 拡張機能は、依存関係として自動的に追加されます。

これらの引数の多くは、PostgreSQL インスタンスのファイルシステムに PostgreSQL 拡張機能をインストールするための拡張制御ファイルに含まれている引数と同じです。詳細については、PostgreSQL ドキュメントの「拡張機能への関連オブジェクトのパッケージ化」にある「拡張ファイル」を参照してください。

出力

この関数は、正常時には `OK` を、エラー時には `NULL` を返します。

- `OK` – TLE 拡張機能がデータベースに正常にインストールされました。
- `NULL` – TLE 拡張機能がデータベースに正常にインストールされませんでした。

使用例

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

この `install_update_path` 関数は、TLE 拡張機能の 2 つの異なるバージョン間の更新パスを提供します。この機能によって、TLE 拡張機能のユーザーは `ALTER EXTENSION ... UPDATE` 構文を使用してバージョンを更新できます。

関数プロトタイプ

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

ロール

pgtle_admin

引数

- name – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。
- fromvers – アップグレードの TLE 拡張機能のソースバージョン。
- tovers – アップグレードする TLE 拡張機能のデスティネーションバージョン。
- ext – 更新の内容。この値には、関数などのオブジェクトが含まれます。

出力

なし。

使用例

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

この register_feature 関数は、指定された内部 PostgreSQL 機能を pgtle.feature_info テーブルに追加します。PostgreSQL フックは、内部 PostgreSQL 機能の一例です。Trusted Language Extensions 開発キットは、PostgreSQL フックの使用をサポートしています。現在、この関数は次の機能をサポートしています。

- passcheck – PostgreSQL のパスワードチェック動作をカスタマイズするプロシージャまたは関数にパスワードチェックフックを登録します。

関数プロトタイプ

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

ロール

pgtle_admin

引数

- proc – その機能に使用するストアドプロシージャまた関数の名前。
- feature – 関数に登録する pg_tle 機能 (passcheck など) の名前。

出力

なし。

使用例

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

この pgtle.register_feature_if_not_exists 関数は、指定した PostgreSQL 機能を pgtle.feature_info テーブルに追加し、その機能を使用する TLE 拡張機能またはその他のプロシージャまたは関数を識別します。フックと Trusted Language Extensions の詳細については、「TLE 拡張機能で PostgreSQL フックを使用する」を参照してください。

関数プロトタイプ

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

ロール

pgtle_admin

引数

- proc – TLE 拡張機能として使用するロジック (コード) を含むストアドプロシージャまたは関数の名前。例えば、pw_hook コードです。

- feature – TLE 関数に登録する PostgreSQL 機能の名前。現在、使用できる機能は passcheck フックだけです。詳細については、「パスワードチェックフック (passcheck)」を参照してください。

出力

指定された拡張機能を登録した後、true を返します。機能が既に登録されていた場合は false を返します。

使用例

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

set_default_version 関数では、TLE 拡張機能に default_version を指定できます。この関数を使用してアップグレードパスを定義し、そのバージョンを TLE 拡張機能のデフォルトとして指定できます。データベースユーザーが CREATE EXTENSION および ALTER EXTENSION ... UPDATE コマンドで TLE 拡張機能を指定すると、そのユーザー用にそのバージョンの TLE 拡張機能がデータベースに作成されます。

この関数は成功すると true を返します。name 引数で指定された TLE 拡張機能が存在しない場合、関数はエラーを返します。同様に、TLE 拡張機能の version が存在しない場合、関数はエラーを返します。

関数プロトタイプ

```
pgtle.set_default_version(name text, version text)
```

ロール

pgtle_admin

引数

- name – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。
- version – デフォルトに設定する TLE 拡張機能のバージョン。

出力

- true — デフォルトバージョンの設定が成功すると、関数は true を返します。
- ERROR – 指定された名前またはバージョンの TLE 拡張機能が存在しない場合は、エラーメッセージを返します。

使用例

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

uninstall_extension 関数は、TLE 拡張機能のすべてのバージョンをデータベースから削除します。この関数により、今後の CREATE EXTENSION の呼び出しで TLE 拡張機能がインストールされないようにします。TLE 拡張機能がデータベースに存在しない場合、エラーが発生します。

uninstall_extension 関数は、データベースで現在アクティブな TLE 拡張機能を削除しません。現在アクティブな TLE 拡張機能を削除するには、DROP EXTENSION を明示的に呼び出して削除する必要があります。

関数プロトタイプ

```
pgtle.uninstall_extension(extname text)
```

ロール

pgtle_admin

引数

- extname – アンインストールする TLE 拡張機能の名前。この名前は、特定のデータベースで使用する TLE 拡張機能をロードするために CREATE EXTENSION で使用される名前と同じです。

出力

なし。

使用例

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(name, version)

`uninstall_extension(name, version)` 関数は、指定されたバージョンの TLE 拡張機能をデータベースから削除します。この関数は、`CREATE EXTENSION` および `ALTER EXTENSION` が TLE 拡張機能を指定されたバージョンにインストールまたは更新するのを防ぎます。この関数は、TLE 拡張機能の指定されたバージョンのすべての更新パスも削除します。この関数は、データベースで現在アクティブになっている TLE 拡張機能をアンインストールしません。TLE 拡張機能を削除するには、明示的に `DROP EXTENSION` を呼び出す必要があります。TLE 拡張機能のすべてのバージョンをアンインストールするには、「pgtle.uninstall_extension(name)」を参照してください。

関数プロトタイプ

```
pgtle.uninstall_extension(extname text, version text)
```

ロール

`pgtle_admin`

引数

- `extname` – TLE 拡張機能の名前。この値は `CREATE EXTENSION` を呼び出すときに使用されます。
- `version` - データベースからアンインストールする TLE 拡張機能のバージョン。

出力

なし。

使用例

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

`uninstall_extension_if_exists` 関数は、特定のデータベースから TLE 拡張機能のすべてのバージョンを削除します。TLE 拡張機能が存在しない場合、関数は何も返しません (エラーメッセージは発生しません)。指定された拡張機能がデータベース内で現在アクティブになっている場合、この関数はその拡張機能を削除しません。この関数を使用してアーティファクトをアンインストールする前に、`DROP EXTENSION` を明示的に呼び出して TLE 拡張機能を削除する必要があります。

関数プロトタイプ

```
pgtle.uninstall_extension_if_exists(extname text)
```

ロール

pgtle_admin

引数

- extname – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。

出力

uninstall_extension_if_exists 関数は、指定された拡張機能をアンインストールした後に、true を返します。指定した拡張機能が存在しない場合、この関数は false を返します。

- true – TLE 拡張機能をアンインストールした後に true を返します。
- false – TLE 拡張機能がデータベースに存在しない場合に false を返します。

使用例

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

uninstall_update_path 関数は TLE 拡張機能から指定された更新パスを削除します。これにより、ALTER EXTENSION ... UPDATE TO では、これを更新パスとして使用できなくなります。

TLE 拡張機能が、この更新パス上のいずれかのバージョンで現在使用されている場合、その拡張機能はデータベースに残ります。

指定された更新パスが存在しない場合、この関数はエラーを発生させます。

関数プロトタイプ

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

ロール

pgtle_admin

引数

- extname – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。
- fromvers – 更新パスで使用されている TLE 拡張機能のソースバージョン。
- tovers – 更新パスで使用されている TLE 拡張機能の送信先バージョン。

出力

なし。

使用例

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

uninstall_update_path_if_exists 関数は、指定された更新パスを TLE 拡張機能から削除するという点で uninstall_update_path に似ています。ただし、更新パスが存在しない場合、この関数はエラーメッセージを表示しません。代わりに、関数は false を返します。

関数プロトタイプ

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

ロール

pgtle_admin

引数

- extname – TLE 拡張機能の名前。この値は CREATE EXTENSION を呼び出すときに使用されます。
- fromvers – 更新パスで使用されている TLE 拡張機能のソースバージョン。

- tovers – 更新パスで使用されている TLE 拡張機能の送信先バージョン。

出力

- true – 関数は TLE 拡張機能のパスを正常に更新しました。
- false – この関数は TLE 拡張機能のパスを更新できませんでした。

使用例

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

unregister_feature 関数は、フックなどの pg_tle 機能を使用するために登録された関数を削除する方法を提供します。機能の登録については、「pgtle.register_feature」を参照してください。

関数プロトタイプ

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

ロール

pgtle_admin

引数

- proc – pg_tle 機能に登録するストアド関数の名前。
- feature – 関数に登録する pg_tle 機能の名前。例えば、passcheck は、お客様が開発した、信頼できる言語拡張機能で使用するために登録できる機能です。詳細については、「パスワードチェックフック (passcheck)」を参照してください。

出力

なし。

使用例

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

unregister_feature 関数は、フックなどの pg_tle 機能を使用するために登録された関数を削除する方法を提供します。詳細については、「TLE 拡張機能で PostgreSQL フックを使用する」を参照してください。機能を正常に登録解除すると true を返します。機能が登録されていない場合は false を返します。

TLE 拡張機能の pg_tle 機能の登録の詳細については、「pgtle.register_feature」を参照してください。

関数プロトタイプ

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

ロール

pgtle_admin

引数

- proc – pg_tle 機能を含めるように登録されたストアド関数の名前。
- feature - 信頼できる言語拡張機能に登録された pg_tle 機能の名前。

出力

次のように false または true を返します。

- true - 関数は拡張機能から機能を正常に登録解除しました。
- false – 関数は TLE 拡張機能から機能を登録解除できませんでした。

使用例

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Trusted Language Extensions for PostgreSQL のフックリファレンス

Trusted Language Extensions for PostgreSQL は PostgreSQL フックをサポートしています。フックは、PostgreSQL のコア機能を拡張するために開発者が利用できる内部コールバックメカニズムで

す。フックを使用することで、開発者はさまざまなデータベースオペレーションで使用する独自の関数やプロシージャを実装できるため、PostgreSQL の動作を何らかの方法で変更できます。例えば、passcheck フックを使用して、ユーザー (ロール) のパスワードを作成または変更する際に提供されたパスワードを PostgreSQL がどのように処理するかをカスタマイズできます。

TLE 拡張機能で利用できる passcheck フックについては、次のドキュメントを参照してください。クライアント認証フックを含む使用可能なフックの詳細については、「Trusted Language Extensions hooks」を参照してください。

パスワードチェックフック (passcheck)

passcheck フックは、以下の SQL コマンドと psql メタコマンドのパスワードチェックプロセス時の PostgreSQL の動作をカスタマイズするために使用されます。

- `CREATE ROLE username ...PASSWORD` – 詳細については、PostgreSQL のドキュメントの「CREATE ROLE」を参照してください。
- `ALTER ROLE username...PASSWORD` – 詳細については、PostgreSQL のドキュメントの「ALTER ROLE」を参照してください。
- `\password username` – このインタラクティブな psql メタコマンドは、`ALTER ROLE ... PASSWORD` 構文を透過的に使用する前にパスワードをハッシュすることで、指定されたユーザーのパスワードを安全に変更します。このメタコマンドは `ALTER ROLE ... PASSWORD` コマンドの安全なラッパーであるため、フックは psql メタコマンドの動作に適用されます。

例については、「パスワードチェックフックコードリスト」を参照してください。

目次

- 関数プロトタイプ
- 引数
- 設定
- 使用に関する注意事項

関数プロトタイプ

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

引数

passcheck フック関数は、次の引数を取ります。

- username – パスワードを設定するロール (ユーザー名) の名前 (テキスト)。
- password – プレーンテキストまたはハッシュ化されたパスワード。入力するパスワードは、password_type で指定されたタイプと一致する必要があります。
- password_type – パスワードの pgtle.password_type 形式を指定します。この形式は、以下のいずれかのオプションになります。
  - PASSWORD_TYPE_PLAINTEXT – プレーンテキストのパスワード。
  - PASSWORD_TYPE_MD5 – MD5 (メッセージダイジェスト 5) アルゴリズムを使用してハッシュ化されたパスワード。
  - PASSWORD_TYPE_SCRAM_SHA_256 – SCRAM-SHA-256 アルゴリズムを使用してハッシュ化されたパスワード。
- valid_until – パスワードが無効になる時間を指定します。この引数はオプションです。この引数を使用する場合は、時間を timestamptz 値で指定します。
- valid_null – このブール値が true に設定されている場合、valid_until オプションは NULL に設定されます。

設定

この pgtle.enable_password_check 関数は、パスチェックフックが有効かどうかを制御します。パスチェックフックには 3 種類の設定があります。

- off – passcheck パスワードチェックフックをオフにします。これは、デフォルト値です。
- on – passcode パスワードチェックフックをオンにして、パスワードとテーブルを照合します。
- require – パスワードチェックフックを定義する必要があります。

使用に関する注意事項

passcheck フックをオンまたはオフにするには、、RDS for PostgreSQL DB インスタンスのカスタム DB パラメータグループを変更する必要があります。

Linux、macOS、Unix の場合:

```
aws rds modify-db-parameter-group \
```

```
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Windows の場合:

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

AWS SDK を使用した Amazon RDS のコード例

次のコード例は、AWS ソフトウェア開発キット (SDK) による Amazon RDS の使用方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDK を使用した Amazon RDS の基本的な例
  - Hello Amazon RDS
  - AWS SDK を使用した Amazon RDS の基本の詳細
  - AWS SDK を使用した Amazon RDS のアクション
    - AWS SDK または CLI で CreateDBInstance を使用する
    - AWS SDK または CLI で CreateDBParameterGroup を使用する
    - AWS SDK または CLI で CreateDBSnapshot を使用する
    - AWS SDK または CLI で DeleteDBInstance を使用する
    - AWS SDK または CLI で DeleteDBParameterGroup を使用する
    - AWS SDK または CLI で DescribeAccountAttributes を使用する
    - AWS SDK または CLI で DescribeDBEngineVersions を使用する
    - AWS SDK または CLI で DescribeDBInstances を使用する
    - AWS SDK または CLI で DescribeDBParameterGroups を使用する
    - AWS SDK または CLI で DescribeDBParameters を使用する
    - AWS SDK または CLI で DescribeDBSnapshots を使用する
    - AWS SDK または CLI で DescribeOrderableDBInstanceOptions を使用する

    - AWS SDK で GenerateRDSAuthToken を使用する
    - AWS SDK または CLI で ModifyDBInstance を使用する
    - AWS SDK または CLI で ModifyDBParameterGroup を使用する
    - AWS SDK または CLI で RebootDBInstance を使用する
- AWS SDK を使用した Amazon RDS のシナリオ
    - Aurora Serverless 作業項目トラッカーの作成
    - リレーショナルデータベースインスタンスの作成
- Amazon RDS のサーバーレス例
    - Lambda 関数での Amazon RDS データベースへの接続

AWS SDK を使用した Amazon RDS の基本的な例

次のコード例は、AWS SDK を使用した Amazon Relational Database Service サービスの基本的な使用方法を説明しています。

例

- Hello Amazon RDS
- AWS SDK を使用した Amazon RDS の基本の詳細
- AWS SDK を使用した Amazon RDS のアクション
    - AWS SDK または CLI で CreateDBInstance を使用する
    - AWS SDK または CLI で CreateDBParameterGroup を使用する
    - AWS SDK または CLI で CreateDBSnapshot を使用する
    - AWS SDK または CLI で DeleteDBInstance を使用する
    - AWS SDK または CLI で DeleteDBParameterGroup を使用する
    - AWS SDK または CLI で DescribeAccountAttributes を使用する
    - AWS SDK または CLI で DescribeDBEngineVersions を使用する
    - AWS SDK または CLI で DescribeDBInstances を使用する
    - AWS SDK または CLI で DescribeDBParameterGroups を使用する
    - AWS SDK または CLI で DescribeDBParameters を使用する
    - AWS SDK または CLI で DescribeDBSnapshots を使用する
    - AWS SDK または CLI で DescribeOrderableDBInstanceOptions を使用する

- AWS SDK で GenerateRDSAuthToken を使用する
- AWS SDK または CLI で ModifyDBInstance を使用する
- AWS SDK または CLI で ModifyDBParameterGroup を使用する
- AWS SDK または CLI で RebootDBInstance を使用する

Hello Amazon RDS

次のコード例は、Amazon RDS の使用を開始する方法を示しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;

public static class HelloRds
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
```

```
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });

        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBInstances」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_rds.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
```

```
                            << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << instanceDBIDs.size() << " RDS instances found." <<
 std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBInstances」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)
```

```
// main uses the AWS SDK for Go V2 to create an Amazon Relational Database
 Service (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n",
 *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBInstances」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
```

```
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeDBInstances」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational
 Database Service
(Amazon RDS) to list the databases in your account.
"""

import boto3
from botocore.exceptions import ClientError
```

```
# Create an RDS client
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBInstances」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
```

```
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「DescribeDBInstances」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon RDS の基本の詳細

次のコード例は、以下を実行する方法を示しています。

- カスタム DB パラメータグループを作成し、パラメータ値を設定します。
- パラメータグループを使用するように設定した DB インスタンスを作成します。DB インスタンスにはデータベースも含まれています。
- インスタンスのスナップショットを取得します。
- インスタンスとパラメータグループを削除します。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
    2.  Selects an engine family and creates a custom DB parameter group using
 the CreateDBParameterGroupAsync method.
    3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync
 method.
    4.  Gets parameters in the group using the DescribeDBParameters method.
    5.  Parses and displays parameters in the group.
    6.  Modifies both the auto_increment_offset and auto_increment_increment
 parameters
        using the ModifyDBParameterGroupAsync method.
    7.  Gets and displays the updated parameters using the DescribeDBParameters
 method with a source of "user".
    8.  Gets a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Displays and selects from a list of micro instance classes available for
 the selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
 parameter group
        using the CreateDBInstanceAsync method.
    11. Waits for DB instance to be ready using the DescribeDBInstancesAsync
 method.
    12. Prints out the connection endpoint string for the new DB instance.
    13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
 method.
    14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
    15. Deletes the DB instance using the DeleteDBInstanceAsync method.
    16. Waits for DB instance to be deleted using the DescribeDbInstances method.
    17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
    */

    private static readonly string sepBar = new('-', 80);
```

```
    private static RDSWrapper rdsWrapper = null!;
    private static ILogger logger = null!;
    private static readonly string engine = "mysql";
    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon RDS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<RDSWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<RDSInstanceScenario>();

        rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
        Console.WriteLine(sepBar);

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await
CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });
```

```
            await ModifyParameters(parameterGroup.DBParameterGroupName,
parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
```

```
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");
```

```
        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");

        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));
```

```
        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
```

```
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
```

```
                $"\t{i}. Engine: {version.Engine} Version
 {version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDbInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
```

```
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB
 instance.</param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"10. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
```

```
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }
```

```
        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection
 string.</param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P
 {instance.Endpoint.Port} "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</
param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;

        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
```

```
        {
            var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
        Console.WriteLine(sepBar);
    }
```

```
    /// <summary>
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup
 parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
 {parameterGroup.DBParameterGroupName}.");
        await
 rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

        Console.WriteLine(sepBar);
    }
```

DB インスタンスアクションのシナリオで使用されるラッパーメソッド。

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 DB instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>List of DBEngineVersions.</returns>
```

```
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }



    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }



   /// <summary>
   /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
   /// to determine when the DB instance is ready to use.
   /// </summary>
   /// <param name="dbName">Name for the DB instance.</param>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
   /// <param name="dbEngine">The engine for the DB instance.</param>
   /// <param name="dbEngineVersion">Version for the DB instance.</param>
   /// <param name="instanceClass">Class for the DB instance.</param>
   /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
   /// <param name="adminName">Admin user name.</param>
   /// <param name="adminPassword">Admin user password.</param>
   /// <returns>DB instance object.</returns>
```

```
   public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
       string parameterGroupName, string dbEngine, string dbEngineVersion,
       string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
   {
       var response = await _amazonRDS.CreateDBInstanceAsync(
           new CreateDBInstanceRequest()
           {
               DBName = dbName,
               DBInstanceIdentifier = dbInstanceIdentifier,
               DBParameterGroupName = parameterGroupName,
               Engine = dbEngine,
               EngineVersion = dbEngineVersion,
               DBInstanceClass = instanceClass,
               AllocatedStorage = allocatedStorage,
               MasterUsername = adminName,
               MasterUserPassword = adminPassword
           });

       return response.DBInstance;
   }



   /// <summary>
   /// Delete a particular DB instance.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
   {
       var response = await _amazonRDS.DeleteDBInstanceAsync(
           new DeleteDBInstanceRequest()
           {
               DBInstanceIdentifier = dbInstanceIdentifier,
               SkipFinalSnapshot = true,
               DeleteAutomatedBackups = true
           });

       return response.DBInstance;
   }
```

DB パラメータグループのシナリオで使用されるラッパーメソッド。

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
```

```
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
```

```
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
 group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

DB スナップショットアクションのシナリオで使用されるラッパーメソッド。

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
```

```
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
```

```
        }
        return results;
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
//! Routine which creates an Amazon RDS instance and demonstrates several
 operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available engine versions for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available database engine versions for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB parameter group.
        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "Let's set some parameter values in your parameter group."
          << std::endl;

Aws::String marker;
Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
// 4.  Get the parameters in the DB parameter group.
if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                     autoIncrementParameters,
                     client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
    if (autoIncParameter.GetIsModifiable() &&
        (autoIncParameter.GetDataType() == "integer")) {
        std::cout << "The " << autoIncParameter.GetParameterName()
                  << " is described as: " <<
                  autoIncParameter.GetDescription() << "." << std::endl;
        if (autoIncParameter.ParameterValueHasBeenSet()) {
            std::cout << "The current value is "
                      << autoIncParameter.GetParameterValue()
                      << "." << std::endl;
        }
        std::vector<int> splitValues = splitToInts(
                autoIncParameter.GetAllowedValues(), '-');
        if (splitValues.size() == 2) {
            int newValue = askQuestionForIntRange(
```

```
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
```

```
                        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

    Aws::RDS::Model::DBInstance dbInstance;
    // 7.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;
```

```
        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
```

```
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
```

```
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();

    // 12. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbInstance);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB instance (y/n)? ")) {
        Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 13. Create a snapshot of the DB instance.
            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
```

```
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for snapshot to become available." << std::endl;

        Aws::RDS::Model::DBSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

```
            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB instance and parameter group (y/n)? "))
 {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
```

```
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
```

```
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    } while (!marker.empty());


    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

```
//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
```

```
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
```

```
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the
parameter group."
                << std::endl;
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) ==
0) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
```

```
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots

- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
import (
 "context"
 "fmt"
 "log"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
 "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the
 AWS SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
```

```
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
```

```
 instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
 parameterGroup)
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(ctx, instance)
 scenario.Cleanup(ctx, instance, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
```

```
  log.Println("Creating a DB parameter group.")
  _, err = scenario.instances.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
```

```
    dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
    updateParams = append(updateParams, dbParam)
   }
  }
  err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
  if err != nil {
   panic(err)
  }
  log.Println("To get a list of parameters that you set previously, specify a
  source of 'user'.")
  userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
  "user")
  if err != nil {
   panic(err)
  }
  log.Println("Here are the parameters you set:")
  for _, param := range userParameters {
   log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
  }
  log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context,
 instanceName string, dbEngine string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(ctx, instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
   *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
```

```
 var engineChoices []string
 for _, engine := range engineVersions {
  engineChoices = append(engineChoices, *engine.EngineVersion)
 }
 engineIndex := scenario.questioner.AskChoice(
  "The available engines for your parameter group are:\n", engineChoices)
 engineSelection := engineVersions[engineIndex]
 instOpts, err := scenario.instances.GetOrderableInstances(ctx,
*engineSelection.Engine,
  *engineSelection.EngineVersion)
 if err != nil {
  panic(err)
 }
 optSet := map[string]struct{}{}
 for _, opt := range instOpts {
  if strings.Contains(*opt.DBInstanceClass, "micro") {
   optSet[*opt.DBInstanceClass] = struct{}{}
  }
 }
 var optChoices []string
 for opt := range optSet {
  optChoices = append(optChoices, opt)
 }
 sort.Strings(optChoices)
 optIndex := scenario.questioner.AskChoice(
  "The available micro DB instance classes for your database engine are:\n",
optChoices)
 storageType := "standard"
 allocatedStorage := int32(5)
 log.Printf("Creating a DB instance named %v and database %v.\n"+
  "The DB instance is configured to use your custom parameter group %v,\n"+
  "selected engine %v,\n"+
  "selected DB instance class %v,"+
  "and %v GiB of %v storage.\n"+
  "This typically takes several minutes.",
  instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
  optChoices[optIndex], allocatedStorage, storageType)
 instance, err = scenario.instances.CreateInstance(
  ctx, instanceName, dbName, *engineSelection.Engine,
*engineSelection.EngineVersion,
  *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
  allocatedStorage, adminUsername, adminPassword)
 if err != nil {
```

```
   panic(err)
  }
  for *instance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx, instanceName)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Instance created and available.")
 }
 log.Println("Instance data:")
 log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *instance.Engine)
 log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance)
 {
 log.Println(
  "You can now connect to your database by using your favorite MySQL client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
   "port, and administrator username to 'mysql'. Then, enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
 log.Println("For more information, see the User Guide for RDS:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
 *types.DBInstance) {
```

```
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.instances.CreateSnapshot(ctx,
 *instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 ctx context.Context, instance *types.DBInstance, parameterGroup
 *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ",
 "y") {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
```

```
  }
  log.Println(
   "Waiting for the DB instance to delete. This typically takes several
 minutes.")
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx,
 *instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBParameterGroupName)
  err = scenario.instances.DeleteParameterGroup(ctx,
 *parameterGroup.DBParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Amazon RDS アクションを管理するためにシナリオによって呼び出される関数を定義します。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}


// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

```
// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
```

```
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
```

```
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
```

```
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

```
// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
```

```
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups

- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

複数のオペレーションを実行します。

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
```

```
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
```

```
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name
 (for example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql
database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
```

```
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
```

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   // Waits until the snapshot instance is available.
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
           String dbSnapshotIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                   .dbSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
               List<DBSnapshot> snapshotList = response.dbSnapshots();
               for (DBSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
```

```
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
```

```
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient

.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption :
orderableDBInstances) {
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
```

```
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
```

```
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .defaultOnly(true)
                   .engine("mysql")
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking
 the createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment
 parameters by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the
 describeDbEngineVersions method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance
 that contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint
 value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
```

```
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
```

```
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database
 and uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
```

```
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
```

```
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

```
// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
```

```
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
```

```
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is
 ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is
 ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
```

```
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset
 or auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
```

```
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
```

```
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the
 database engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブなシナリオを実行します。

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
```

```
        self.instance_wrapper = instance_wrapper

    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB parameter group that is compatible with a selected engine
family.

        :param parameter_group_name: The name given to the newly created
parameter group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
{parameter_group_name}."
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions =
self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
```

```
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine,
parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.
```

```
        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the
DB instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
```

```
                "Which micro DB instance class do you want to use? ",
inst_choices
            )
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
                f"The DB instance is configured to use your custom parameter
group {group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
```

```
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P
 {db_inst['Endpoint']['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
 instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
```

```
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances
must first
        be deleted.

        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)?
",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes
several minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name,
db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)
```

```
        parameter_group = self.create_parameter_group(parameter_group_name,
 db_engine)
        self.update_parameters(parameter_group_name)
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Amazon RDS アクションを管理するためにシナリオによって呼び出される関数を定義します。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
```

```
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
```

```
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
```

```
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

```
    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups

- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

`Package.swift` ファイル。

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "rds-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
```

```
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "rds-scenario",
            dependencies: [
                .product(name: "AWSRDS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Swift コードファイル、`entry.swift`。

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Relational Database Service (RDS).
//

import ArgumentParser
import Foundation
import AWSRDS

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"
    @Option(help: "The username to use for the database administrator.")
    var dbUsername = "admin"
    @Option(help: "The password to use for the database administrator.")
    var dbPassword: String

    static var configuration = CommandConfiguration(
        commandName: "rds-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon RDS
 Instances
        using the AWS SDK for Swift.
```

```
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let example = try await Example(region: awsRegion, username: dbUsername,
 password: dbPassword)

        await example.run()
    }
}

class Example {
    let rdsClient: RDSClient

    // Storage for AWS RDS properties

    let dbUsername: String
    let dbPassword: String
    var dbInstanceIdentifier: String
    var dbSnapshotIdentifier: String
    var dbParameterGroupName: String
    var dbParameterGroup: RDSClientTypes.DBParameterGroup?
    var selectedEngineVersion: String?

    init(region: String, username: String, password: String) async throws{
        let rdsConfig = try await RDSClient.RDSClientConfiguration(region:
 region)
        rdsClient = RDSClient(config: rdsConfig)

        dbUsername = username
        dbPassword = password
        dbParameterGroupName = ""
        dbInstanceIdentifier = ""
        dbSnapshotIdentifier = ""
    }

    /// The example's main body.
    func run() async {
        var parameterGroupFamilies: Set<String> = []

        //====================================================================
```

```
        // 1. Get available database engine families for MySQL.
        //=========================================================================

        let engineVersions = await getDBEngineVersions(engineName: "mysql")

        for version in engineVersions {
            if version.dbParameterGroupFamily != nil {
                parameterGroupFamilies.insert(version.dbParameterGroupFamily!)
            }
        }

        if engineVersions.count > 0 {
            selectedEngineVersion = engineVersions.last!.engineVersion
        } else {
            print("*** Unable to find a valid database engine version. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Found \(parameterGroupFamilies.count) parameter group families:")
        for family in parameterGroupFamilies {
            print("    \(family)")
        }

        //=========================================================================
        // 2. Select an engine family and create a custom DB parameter group.
        //    We select a family by sorting the set of family names, then
        //    choosing the last one.
        //=========================================================================

        let sortedFamilies = parameterGroupFamilies.sorted()

        guard let selectedFamily = sortedFamilies.last else {
            print("*** Unable to find a database engine family. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Selected database engine family \(selectedFamily)")

        dbParameterGroupName = tempName(prefix: "rds-example")
```

```
        print("Creating a database parameter group named \(dbParameterGroupName)
using \(selectedFamily)")
        dbParameterGroup = await createDBParameterGroup(groupName:
dbParameterGroupName,
                                                        familyName:
selectedFamily)

        //===========================================================================
        // 3. Get the parameter group's details.
        //===========================================================================

        print("Getting the database parameter group list...")
        let dbParameterGroupList = await describeDBParameterGroups(groupName:
dbParameterGroupName)
        guard let dbParameterGroupList else {
            await cleanUp()
            return
        }

        print("Found \(dbParameterGroupList.count) parameter groups...")
        for group in dbParameterGroupList {
            print("    \(group.dbParameterGroupName ?? "<unknown>")")
        }
        print()

        //===========================================================================
        // 4. Get a list of the parameter group's parameters. This list is
        //    likely to be long, so use pagination. Find the
        //    auto_increment_offset and auto_increment_increment parameters.
        //===========================================================================

        let parameters = await describeDBParameters(groupName:
dbParameterGroupName)

        //===========================================================================
        // 5. Parse and display each parameter's name, description, and
        //    allowed values.
        //===========================================================================

        for parameter in parameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
```

```
            }

            if name == "auto_increment_offset" || name ==
"auto_increment_increment" {
                print("Parameter \(name):")
                print("          Value: \(parameter.parameterValue ??
"<undefined>")")
                print("      Data type: \(parameter.dataType ?? "<unknown>")")
                print("    Description: \(parameter.description ?? "")")
                print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
                print(String(repeating: "=", count: 78))
            }
        }

        //============================================================================
        // 6. Modify both the auto_increment_offset and
        //    auto_increment_increment parameters in one call in the custom
        //    parameter group. Set their parameterValue fields to a new
        //    permitted value.
        //============================================================================

        print("Setting auto_increment_offset and auto_increment_increment both to
5...")
        await modifyDBParameters(groupName: dbParameterGroupName)

        //============================================================================
        // 7. Get and display the updated parameters, specifying a source of
        //    "user" to get only the modified parameters.
        //============================================================================

        let updatedParameters = await describeDBParameters(groupName:
dbParameterGroupName, source: "user")

        for parameter in updatedParameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            print("Parameter \(name):")
            print("          Value: \(parameter.parameterValue ??
"<undefined>")")
```

```
            print("      Data type: \(parameter.dataType ?? "<unknown>")")
            print("    Description: \(parameter.description ?? "")")
            print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
            print(String(repeating: "=", count: 78))
        }

        //========================================================================
        // 8. Get a list of allowed engine versions using
        //    DescribeRDSEngineVersions.
        //========================================================================

        await listAllowedEngines(family: selectedFamily)

        //========================================================================
        // 9. Get a list of micro instance classes available for the selected
        //    engine and engine version.
        //========================================================================

        let dbInstanceClass = await chooseMicroInstance(engine: "mysql",
engineVersion: selectedEngineVersion)
        guard let dbInstanceClass else {
            print("Did not get a valid instance class. Canceling operations.")
            await cleanUp()
            return
        }

        //========================================================================
        // 10. Create an RDS database that contains a MySQL database and uses
        //     the parameter group we created.
        //========================================================================

        print("Creating the database instance...")

        guard let instanceClass = dbInstanceClass.dbInstanceClass else {
            print("Instance class name is unknown. Canceling operations.")
            await cleanUp()
            return
        }

        dbInstanceIdentifier = tempName(prefix: "sample-identifier")
        let dbInstanceArn = await createDBInstance(
            name: "SampleDatabase\(Int.random(in: 1000000..<1000000000))",
            instanceIdentifier: dbInstanceIdentifier,
```

```
            parameterGroupName: dbParameterGroupName,
            engine: "mysql",
            engineVersion: selectedEngineVersion!,
            instanceClass: instanceClass,
            username: dbUsername,
            password: dbPassword
        )

        if dbInstanceArn == nil {
            await cleanUp()
            return
        }

        //=========================================================================
        // 11. Wait for the database instance to be ready by calling
        //     DescribeDBInstances repeatedly until it reports
        //     dbInstanceStatus as "available". This can take upwards of 10
        //     minutes, let the user know that.
        //=========================================================================

        guard let endpoint = await waitUntilDBInstanceReady(instanceIdentifier:
dbInstanceIdentifier) else {
            print("\nDid not get a valid endpoint from AWS RDS.")
            await cleanUp()
            return
        }

        guard let endpointAddress = endpoint.address else {
            print("\nNo endpoint address returned.")
            await cleanUp()
            return
        }
        guard let endpointPort = endpoint.port else {
            print("\nNo endpoint port returned.")
            await cleanUp()
            return
        }

        //=========================================================================
        // 12. Display connection information for the database instance.
        //=========================================================================

        print("\nTo connect to the new database instance using 'mysql' from the
shell:")
```

```
        print("    mysql -h \(endpointAddress) -P \(endpointPort) -u
\(self.dbUsername)")

        //========================================================================
        // 13. Create a snapshot of the database instance.
        //========================================================================

        dbSnapshotIdentifier = tempName(prefix: "sample-snapshot")
        await createDBSnapshot(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        //========================================================================
        // 14. Wait for the snapshot to be ready.
        //========================================================================

        await waitUntilDBSnapshotReady(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        // That's it! Clean up and exit!

        print("Example complete! Cleaning up...")
        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items.
    func cleanUp() async {
        print("Deleting the database instance \(dbInstanceIdentifier)...")
        await deleteDBInstance(instanceIdentifier: dbInstanceIdentifier)
        await waitUntilDBInstanceDeleted(instanceIdentifier:
dbInstanceIdentifier)

        print("Deleting the database parameter group \(dbParameterGroupName)...")
        await deleteDBParameterGroup(groupName: dbParameterGroupName)
    }

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
```

```
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
```

```
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
[RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
```

```
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

```
    /// Output a list of the database engine versions supported by the
    /// specified family.
    ///
    /// - Parameter family: The family for which to list allowed database
    ///   engines.
    func listAllowedEngines(family: String?) async {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    dbParameterGroupFamily: family,
                    engine: "mysql"
                )
            )

            guard let engineVersions = output.dbEngineVersions else {
                print("No engine versions returned.")
                return
            }

            print("Found \(engineVersions.count) database engine versions:")
            for version in engineVersions {
                print("    \(version.engineVersion ?? "<unknown>"):
\(version.dbEngineDescription ?? "")")
            }
        } catch {
            print("*** Error getting database engine version list:
\(error.localizedDescription)")
            return
        }
    }

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
```

```
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }

            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a new database instance.
    ///
    /// - Parameters:
```

```
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
    func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                          engine: String, engineVersion: String, instanceClass:
String,
                          username: String, password: String) async -> String? {
        do {
            let output = try await rdsClient.createDBInstance(
                input: CreateDBInstanceInput(
                    allocatedStorage: 100,
                    dbInstanceClass: instanceClass,
                    dbInstanceIdentifier: instanceIdentifier,
                    dbName: name,
                    dbParameterGroupName: parameterGroupName,
                    engine: engine,
                    engineVersion: engineVersion,
                    masterUserPassword: password,
                    masterUsername: username,
                    storageType: "gp2"
                )
            )

            guard let dbInstance = output.dbInstance else {
                print("*** Unable to get the database instance.")
                return nil
            }

            return dbInstance.dbInstanceArn
        } catch {
```

```
            print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
```

```
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID
\(snapshot.dbiResourceId ?? "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
        }
    }

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
```

```
    ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
    func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
        var snapshotReady = false

        putString("Waiting for the snapshot to be ready...")

        do {
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }

                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }
```

```
    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }

    /// Wait until the specified database instance has been deleted.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to wait for.
    func waitUntilDBInstanceDeleted(instanceIdentifier: String) async {
        putString("Waiting for the database instance to be deleted. This may take
a few minutes...")
        do {
            var isDatabaseDeleted = false
            var foundInstance = false

            while !isDatabaseDeleted {
                let output = try await rdsClient.describeDBInstances(input:
DescribeDBInstancesInput())
                guard let instanceList = output.dbInstances else {
                    return
                }

                foundInstance = false

                for instance in instanceList {
                    guard let foundInstanceIdentifier =
instance.dbInstanceIdentifier else {
                        continue
                    }
```

```
                if instanceIdentifier == foundInstanceIdentifier {
                    foundInstance = true
                    break
                } else {
                    putString(".")
                    do {
                        try await Task.sleep(for: .seconds(15))
                    } catch {
                        print("\n*** Error pausing the task!")
                    }
                }
            }
            if !foundInstance {
                isDatabaseDeleted = true
                print()
            }
        }
    } catch {
        print("\n*** Error waiting for the database instance to be deleted:
\(error.localizedDescription)")
    }
}

/// Delete the specified database parameter group.
///
/// - Parameter groupName: The name of the parameter group to delete.
func deleteDBParameterGroup(groupName: String) async {
    do {
        _ = try await rdsClient.deleteDBParameterGroup(
            input: DeleteDBParameterGroupInput(
                dbParameterGroupName: groupName
            )
        )
    } catch {
        print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
    }
}

/// Generate and return a unique file name that begins with the specified
/// string.
///
/// - Parameters:
```

```
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Print a string to standard output without a trailing newline, and
    /// without buffering.
    ///
    /// - Parameter str: The string to output.
    func putString(_ str: String = "") {
        if str.length >= 1 {
            let data = str.data(using: .utf8)
            guard let data else {
                return
            }
            FileHandle.standardOutput.write(data)
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API の詳細については、AWS SDK for Swift API リファレンスの以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon RDS のアクション

次のコード例では、AWS SDK を使用して個々の Amazon RDS アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらは Amazon RDS API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは AWS SDK を使用した Amazon RDS のシナリオ のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「Amazon Relational Database Service API Reference」を参照してください。

例

- AWS SDK または CLI で CreateDBInstance を使用する
- AWS SDK または CLI で CreateDBParameterGroup を使用する

- AWS SDK または CLI で CreateDBSnapshot を使用する
- AWS SDK または CLI で DeleteDBInstance を使用する
- AWS SDK または CLI で DeleteDBParameterGroup を使用する
- AWS SDK または CLI で DescribeAccountAttributes を使用する
- AWS SDK または CLI で DescribeDBEngineVersions を使用する
- AWS SDK または CLI で DescribeDBInstances を使用する
- AWS SDK または CLI で DescribeDBParameterGroups を使用する
- AWS SDK または CLI で DescribeDBParameters を使用する
- AWS SDK または CLI で DescribeDBSnapshots を使用する
- AWS SDK または CLI で DescribeOrderableDBInstanceOptions を使用する
- AWS SDK で GenerateRDSAuthToken を使用する
- AWS SDK または CLI で ModifyDBInstance を使用する
- AWS SDK または CLI で ModifyDBParameterGroup を使用する
- AWS SDK または CLI で RebootDBInstance を使用する

AWS SDK または CLI で **CreateDBInstance** を使用する

次のサンプルコードは、CreateDBInstance を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「CreateDBInstance」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBName(DB_NAME);
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetEngine(engineVersion.GetEngine());
    request.SetEngineVersion(engineVersion.GetEngineVersion());
    request.SetDBInstanceClass(dbInstanceClass);
    request.SetStorageType(DB_STORAGE_TYPE);
    request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
            client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
                  << outcome.GetError().GetMessage()
```

```
                    << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「CreateDBInstance」を参照してください。

CLI

AWS CLI

DB インスタンスを作成するには

次の create-db-instance の例は、必須のオプションを使用して新しい DB インスタンスを起動します。

```
aws rds create-db-instance \
    --db-instance-identifier test-mysql-instance \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username admin \
    --master-user-password secret99 \
    --allocated-storage 20
```

出力:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "12:55-13:25",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-12345abc",
```

```
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql5.7",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2ff2ff2f",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2c"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2d"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
```

```
        },
        "PreferredMaintenanceWindow": "sun:08:07-sun:08:37",
        "PendingModifiedValues": {
            "MasterUserPassword": "****"
        },
        "MultiAZ": false,
        "EngineVersion": "5.7.22",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:mysql-5-7",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": true,
        "StorageType": "gp2",
        "DbInstancePort": 0,
        "StorageEncrypted": false,
        "DbiResourceId": "db-5555EXAMPLE44444444EXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:test-mysql-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": []
    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「Amazon RDS DB インスタンスの作成」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「CreateDBInstance」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
```

```
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「CreateDBInstance」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
```

```
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
```

```
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
```

```
                DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「CreateDBInstance」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
```

```
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
```

```
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「CreateDBInstance」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
```

```
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「CreateDBInstance」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
```

```
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
```

```
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「CreateDBInstance」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
```

```
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
    func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                          engine: String, engineVersion: String, instanceClass:
String,
                          username: String, password: String) async -> String? {
        do {
            let output = try await rdsClient.createDBInstance(
                input: CreateDBInstanceInput(
                    allocatedStorage: 100,
                    dbInstanceClass: instanceClass,
                    dbInstanceIdentifier: instanceIdentifier,
                    dbName: name,
                    dbParameterGroupName: parameterGroupName,
                    engine: engine,
                    engineVersion: engineVersion,
                    masterUserPassword: password,
                    masterUsername: username,
                    storageType: "gp2"
                )
            )

            guard let dbInstance = output.dbInstance else {
                print("*** Unable to get the database instance.")
                return nil
            }

            return dbInstance.dbInstanceArn
        } catch {
```

```
            print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「CreateDBInstance」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateDBParameterGroup** を使用する

次のサンプルコードは、CreateDBParameterGroup を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
```

```
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }
```

- 詳細については、AWS SDK for .NET API リファレンスの「CreateDBParameterGroup」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
```

```
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 詳細については、「AWS SDK for C++ API リファレンス」の「CreateDBParameterGroup」を参照してください。

CLI

AWS CLI

DB パラメータグループを作成するには

次の `create-db-parameter-group` の例は、DB パラメータグループを作成します。

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL5.6 \
    --description "My new parameter group"
```

出力:

```
{
    "DBParameterGroup": {
        "DBParameterGroupName": "mydbparametergroup",
        "DBParameterGroupFamily": "mysql5.6",
        "Description": "My new parameter group",
        "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:mydbparametergroup"
```

```
    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB パラメータグループを作成する」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「CreateDBParameterGroup」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
```

```
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}
```

- 詳細については、AWS SDK for Go API リファレンスの「CreateDBParameterGroup」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();
```

```
            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- 詳細については、AWS SDK for Java 2.x API リファレンスの「CreateDBParameterGroup」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「CreateDBParameterGroup」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
" iv_dbparametergroupname   = 'mydbparametergroup'
" iv_dbparametergroupfamily = 'mysql8.0'
" iv_description            = 'My custom DB parameter group for MySQL 8.0'
TRY.
    oo_result = lo_rds->createdbparametergroup(
      iv_dbparametergroupname   = iv_dbparametergroupname
      iv_dbparametergroupfamily = iv_dbparametergroupfamily
      iv_description            = iv_description ).
    MESSAGE 'DB parameter group created.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbparmgralrexfault.
    MESSAGE 'DB parameter group already exists.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
    MESSAGE 'DB parameter group quota exceeded.' TYPE 'I'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「CreateDBParameterGroup」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS
```

```
    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「CreateDBParameterGroup」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateDBSnapshot** を使用する

次のサンプルコードは、CreateDBSnapshot を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「CreateDBSnapshot」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「CreateDBSnapshot」を参照してください。

CLI

AWS CLI

DB スナップショットを作成するには

次の `create-db-snapshot` の例は、DB スナップショットを作成します。

```
aws rds create-db-snapshot \
    --db-instance-identifier database-mysql \
    --db-snapshot-identifier mydbsnapshot
```

出力:

```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "database-mysql",
        "Engine": "mysql",
        "AllocatedStorage": 100,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1b",
        "VpcId": "vpc-6594f31c",
        "InstanceCreateTime": "2019-04-30T15:45:53.663Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
        "Iops": 1000,
        "OptionGroupName": "default:mysql-5-6",
        "PercentProgress": 0,
        "StorageType": "io1",
        "Encrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
AKIAIOSFODNN7EXAMPLE",
        "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
        "IAMDatabaseAuthenticationEnabled": false,
        "ProcessorFeatures": [],
        "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB スナップショットの作成」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「CreateDBSnapshot」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「CreateDBSnapshot」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
```

```
        }
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「CreateDBSnapshot」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「CreateDBSnapshot」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
```

```
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「CreateDBSnapshot」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
```

```
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「CreateDBSnapshot」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
 async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
```

```
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID
\(snapshot.dbiResourceId ?? "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「CreateDBSnapshot」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteDBInstance** を使用する

次のサンプルコードは、DeleteDBInstance を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DeleteDBInstance」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
```

```
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DeleteDBInstance」を参照してください。

CLI

AWS CLI

DB インスタンスを削除するには

次の delete-db-instance の例は、test-instance-final-snap という名前の最終 DB スナップショットを作成した後に、指定された DB インスタンスを削除します。

```
aws rds delete-db-instance \
    --db-instance-identifier test-instance \
    --final-db-snapshot-identifier test-instance-final-snap
```

出力:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-instance",
        "DBInstanceStatus": "deleting",
        ...some output truncated...
```

```
    }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteDBInstance」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DeleteDBInstance」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DeleteDBInstance」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DeleteDBInstance」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「DeleteDBInstance」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DeleteDBInstance」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
```

```
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DeleteDBInstance」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteDBParameterGroup** を使用する

次のサンプルコードは、DeleteDBParameterGroup を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
```

```
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 詳細については、AWS SDK for .NET API リファレンスの「DeleteDBParameterGroup」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- 詳細については、「AWS SDK for C++ API リファレンス」の「DeleteDBParameterGroup」を参照してください。

CLI

AWS CLI

DB パラメータグループを削除するには

次の command の例は、DB パラメータグループを削除します。

```
aws rds delete-db-parameter-group \
    --db-parameter-group-name mydbparametergroup
```

このコマンドでは何も出力されません。

詳細については、「Amazon RDS ユーザーガイド」の「DB パラメータグループを使用する」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteDBParameterGroup」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- 詳細については、AWS SDK for Go API リファレンスの「DeleteDBParameterGroup」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while
database
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
```

```
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
 DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- 詳細については、AWS SDK for Java 2.x API リファレンスの「DeleteDBParameterGroup」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DeleteDBParameterGroup」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
" iv_dbparametergroupname = 'mydbparametergroup'
TRY.
    lo_rds->deletedbparametergroup(
      iv_dbparametergroupname = iv_dbparametergroupname ).
    MESSAGE 'DB parameter group deleted.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DeleteDBParameterGroup」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
```

```
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DeleteDBParameterGroup」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeAccountAttributes** を使用する

次のサンプルコードは、DescribeAccountAttributes を使用する方法を説明しています。

CLI

AWS CLI

アカウントの属性を記述するには

次の describe-account-attributes の例は、現在の AWS アカウントの属性を取得します。

```
aws rds describe-account-attributes
```

出力:

```
{
    "AccountQuotas": [
        {
            "Max": 40,
```

```
            "Used": 4,
            "AccountQuotaName": "DBInstances"
        },
        {
            "Max": 40,
            "Used": 0,
            "AccountQuotaName": "ReservedDBInstances"
        },
        {
            "Max": 100000,
            "Used": 40,
            "AccountQuotaName": "AllocatedStorage"
        },
        {
            "Max": 25,
            "Used": 0,
            "AccountQuotaName": "DBSecurityGroups"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "AuthorizationsPerDBSecurityGroup"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBParameterGroups"
        },
        {
            "Max": 100,
            "Used": 3,
            "AccountQuotaName": "ManualSnapshots"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "EventSubscriptions"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBSubnetGroups"
        },
        {
```

```
            "Max": 20,
            "Used": 1,
            "AccountQuotaName": "OptionGroups"
        },
        {
            "Max": 20,
            "Used": 6,
            "AccountQuotaName": "SubnetsPerDBSubnetGroup"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "ReadReplicasPerMaster"
        },
        {
            "Max": 40,
            "Used": 1,
            "AccountQuotaName": "DBClusters"
        },
        {
            "Max": 50,
            "Used": 0,
            "AccountQuotaName": "DBClusterParameterGroups"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "DBClusterRoles"
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeAccountAttributes」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import
 software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
```

```
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeAccountAttributes」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeAccountAttributes」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeDBEngineVersions** を使用する

次のサンプルコードは、DescribeDBEngineVersions を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
```

```
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBEngineVersions」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
```

```
                                        const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBEngineVersions」を参照してください。

CLI

AWS CLI

MySQL DB エンジンの DB エンジンバージョンを記述するには

次の describe-db-engine-versions の例は、指定された DB エンジンの各 DB エンジンバージョンに関する詳細が表示されます。

```
aws rds describe-db-engine-versions \
    --engine mysql
```

出力:

```
{
    "DBEngineVersions": [
        {
            "Engine": "mysql",
            "EngineVersion": "5.5.46",
            "DBParameterGroupFamily": "mysql5.5",
            "DBEngineDescription": "MySQL Community Edition",
            "DBEngineVersionDescription": "MySQL 5.5.46",
            "ValidUpgradeTarget": [
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.53",
                    "Description": "MySQL 5.5.53",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.54",
                    "Description": "MySQL 5.5.54",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
```

```
                    "EngineVersion": "5.5.57",
                    "Description": "MySQL 5.5.57",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                ...some output truncated...
            ]
        }
```

詳細については、「Amazon RDS ユーザーガイド」の「Amazon Relational Database Service (Amazon RDS) とは」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeDBEngineVersions」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBEngineVersions」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
```

```
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeDBEngineVersions」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBEngineVersions」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    " iv_engine                = 'mysql'
    " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
    TRY.
        oo_result = lo_rds->describedbengineversions(
          iv_engine                = iv_engine
          iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
        DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
        MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DescribeDBEngineVersions」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
 [RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeDBEngineVersions」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeDBInstances** を使用する

次のサンプルコードは、DescribeDBInstances を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBInstances」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBInstances」を参照してください。

CLI

AWS CLI

DB インスタンスを記述するには

次の describe-db-instances の例は、指定された DB インスタンスの詳細情報を取得します。

```
aws rds describe-db-instances \
    --db-instance-identifier mydbinstancecf
```

出力:

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "mydbinstancecf",
            "DBInstanceClass": "db.t3.small",
            "Engine": "mysql",
            "DBInstanceStatus": "available",
            "MasterUsername": "masterawsuser",
            "Endpoint": {
                "Address": "mydbinstancecf.abcexample.us-
east-1.rds.amazonaws.com",
                "Port": 3306,
```

```
                "HostedZoneId": "Z2R2ITUGPM61AM"
            },
            ...some output truncated...
        }
    ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeDBInstances」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
```

```
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBInstances」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeDBInstances」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest
 {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「DescribeDBInstances」を参照してください。

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「DescribeDBInstances」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBInstances」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「DescribeDBInstances」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
 RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
 may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
```

```
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeDBInstances」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeDBParameterGroups** を使用する

次のサンプルコードは、DescribeDBParameterGroups を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBParameterGroups」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBParameterGroups」を参照してください。

CLI

AWS CLI

DB パラメータグループを記述するには

次の describe-db-parameter-groups の例では、DB パラメータグループに関する詳細を取得します。

```
aws rds describe-db-parameter-groups
```

出力:

```
{
    "DBParameterGroups": [
        {
            "DBParameterGroupName": "default.aurora-mysql5.7",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "Default parameter group for aurora-mysql5.7",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-mysql5.7"
        },
        {
            "DBParameterGroupName": "default.aurora-postgresql9.6",
            "DBParameterGroupFamily": "aurora-postgresql9.6",
            "Description": "Default parameter group for aurora-postgresql9.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-postgresql9.6"
        },
        {
            "DBParameterGroupName": "default.aurora5.6",
            "DBParameterGroupFamily": "aurora5.6",
            "Description": "Default parameter group for aurora5.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora5.6"
        },
        {
            "DBParameterGroupName": "default.mariadb10.1",
            "DBParameterGroupFamily": "mariadb10.1",
            "Description": "Default parameter group for mariadb10.1",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.mariadb10.1"
        },
```

```
        ...some output truncated...
    ]
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB パラメータグループを使用する」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeDBParameterGroups」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
```

```
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBParameterGroups」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .maxRecords(20)
                   .build();
```

```
            DescribeDbParameterGroupsResponse response =
 rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbParameterGroupName());
                System.out.println("The group description is " +
 group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeDBParameterGroups」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
```

```
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBParameterGroups」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「DescribeDBParameterGroups」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
" iv_dbparametergroupname = 'mydbparametergroup'
TRY.
    oo_result = lo_rds->describedbparametergroups(
      iv_dbparametergroupname = iv_dbparametergroupname ).
    MESSAGE 'DB parameter group retrieved.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DescribeDBParameterGroups」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
```

```
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeDBParameterGroups」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeDBParameters** を使用する

次のサンプルコードは、DescribeDBParameters を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBParameters」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
```

```
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBParameters」を参照してください。

CLI

AWS CLI

DB パラメータグループのパラメータを記述するには

次の describe-db-parameters の例では、指定された DB パラメータグループに関する詳細を取得します。

```
aws rds describe-db-parameters \
    --db-parameter-group-name mydbpg
```

出力:

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have
 only an xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "auto_generate_certs",
            "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        ...some output truncated...
    ]
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB パラメータグループを使用する」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeDBParameters」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
```

```
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBParameters」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   // Retrieve parameters in the group.
   public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
       try {
           DescribeDbParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                       .dbParameterGroupName(dbGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                       .dbParameterGroupName(dbGroupName)
                       .source("user")
                       .build();
           }
```

```
            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「DescribeDBParameters」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this
 prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
```

```
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBParameters」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'
```

```
# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「DescribeDBParameters」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " iv_source               = 'user' (optional - filters by parameter source)
    TRY.
        oo_result = lo_rds->describedbparameters(
          iv_dbparametergroupname = iv_dbparametergroupname
          iv_source               = iv_source ).
        DATA(lv_param_count) = lines( oo_result->get_parameters( ) ).
        MESSAGE |Retrieved { lv_param_count } parameters.| TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
```

```
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DescribeDBParameters」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
 [RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )
```

```
            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeDBParameters」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeDBSnapshots** を使用する

次のサンプルコードは、DescribeDBSnapshots を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeDBSnapshots」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBSnapshots」を参照してください。

CLI

AWS CLI

例 1: DB インスタンスの DB スナップショットを記述するには

次の describe-db-snapshots の例は、DB インスタンスの DB スナップショットの詳細を取得します。

```
aws rds describe-db-snapshots \
    --db-snapshot-identifier mydbsnapshot
```

出力:

```
{
    "DBSnapshots": [
        {
            "DBSnapshotIdentifier": "mydbsnapshot",
            "DBInstanceIdentifier": "mysqldb",
            "SnapshotCreateTime": "2018-02-08T22:28:08.598Z",
            "Engine": "mysql",
            "AllocatedStorage": 20,
            "Status": "available",
            "Port": 3306,
            "AvailabilityZone": "us-east-1f",
            "VpcId": "vpc-6594f31c",
            "InstanceCreateTime": "2018-02-08T22:24:55.973Z",
            "MasterUsername": "mysqladmin",
            "EngineVersion": "5.6.37",
            "LicenseModel": "general-public-license",
            "SnapshotType": "manual",
            "OptionGroupName": "default:mysql-5-6",
            "PercentProgress": 100,
            "StorageType": "gp2",
            "Encrypted": false,
            "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
            "IAMDatabaseAuthenticationEnabled": false,
            "ProcessorFeatures": [],
            "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
        }
    ]
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB スナップショットの作成」を参照してください。

例 2: 手動で作成されたスナップショットの数を調べるには

次の describe-db-snapshots の例は、--query オプションで length 演算子を使用して、特定の AWS リージョンに手動で作成されたスナップショットの数を返します。

```
aws rds describe-db-snapshots \
    --snapshot-type manual \
    --query "length(*[].{DBSnapshots:SnapshotType})" \
    --region eu-central-1
```

出力:

```
35
```

詳細については、「Amazon RDS ユーザーガイド」の「DB スナップショットの作成」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeDBSnapshots」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeDBSnapshots」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeDBSnapshots」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if
 error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「DescribeDBSnapshots」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
    ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
    func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
 String) async {
        var snapshotReady = false

        putString("Waiting for the snapshot to be ready...")

        do {
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }
```

```
                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeDBSnapshots」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeOrderableDBInstanceOptions** を使用する

次のサンプルコードは、DescribeOrderableDBInstanceOptions を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
   {
       // Use a paginator to get a list of DB instance options.
       var results = new List<OrderableDBInstanceOption>();
       var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
           new DescribeOrderableDBInstanceOptionsRequest()
           {
               Engine = engine,
               EngineVersion = engineVersion,
           });
       // Get the entire list using the paginator.
       await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
       {
           results.Add(instanceOptions);
       }
       return results;
   }
```

- API の詳細については、AWS SDK for .NET API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
```

```
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

CLI

AWS CLI

順序設定可能な DB インスタンスオプションを記述するには

次の describe-orderable-db-instance-options の例は、MySQL DB エンジンを実行する DB インスタンスの順序設定可能なオプションの詳細を取得します。

```
aws rds describe-orderable-db-instance-options \
    --engine mysql
```

出力:

```
{
    "OrderableDBInstanceOptions": [
        {
            "MinStorageSize": 5,
            "ReadReplicaCapable": true,
            "MaxStorageSize": 6144,
            "AvailabilityZones": [
                {
                    "Name": "us-east-1a"
                },
                {
                    "Name": "us-east-1b"
                },
                {
                    "Name": "us-east-1c"
                },
                {
                    "Name": "us-east-1d"
                }
            ],
            "SupportsIops": false,
            "AvailableProcessorFeatures": [],
            "MultiAZCapable": true,
```

```
            "DBInstanceClass": "db.m1.large",
            "Vpc": true,
            "StorageType": "gp2",
            "LicenseModel": "general-public-license",
            "EngineVersion": "5.5.46",
            "SupportsStorageEncryption": false,
            "SupportsEnhancedMonitoring": true,
            "Engine": "mysql",
            "SupportsIAMDatabaseAuthentication": false,
            "SupportsPerformanceInsights": false
        }
    ]
    ...some output truncated...
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
```

```
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by
 the DB instance.
        :return: The list of DB instance options that can be used to create a
 compatible DB instance.
        """
        try:
            inst_opts = []
```

```
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
```

```
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
TYPE 'I'.
   ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「DescribeOrderableDBInstanceOptions」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []
```

```
            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }

            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK で **GenerateRDSAuthToken** を使用する

次の例は、GenerateRDSAuthToken を使用する方法を説明しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

RdsUtilities クラスを使用して認証トークンを生成します。

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
```

```
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API の詳細については、AWS SDK for Java 2.xAPI リファレンスの「GenerateRDSAuthToken」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ModifyDBInstance** を使用する

次のサンプルコードは、ModifyDBInstance を使用する方法を説明しています。

CLI

AWS CLI

例 1: DB インスタンスを変更するには

次の modify-db-instance の例は、オプショングループとパラメータグループを互換性のある Microsoft SQL Server DB インスタンスに関連付けます。--apply-immediately パラメータを使用することで、次のメンテナンスウィンドウを待つことなく、オプショングループとパラメータグループをすぐに関連付けることができます。

```
aws rds modify-db-instance \
    --db-instance-identifier database-2 \
    --option-group-name test-se-2017 \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --apply-immediately
```

出力:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "database-2",
        "DBInstanceClass": "db.r4.large",
        "Engine": "sqlserver-se",
        "DBInstanceStatus": "available",

        ...output omitted...

        "DBParameterGroups": [
            {
                "DBParameterGroupName": "test-sqlserver-se-2017",
                "ParameterApplyStatus": "applying"
            }
        ],
        "AvailabilityZone": "us-west-2d",

        ...output omitted...

        "MultiAZ": true,
        "EngineVersion": "14.00.3281.6.v1",
        "AutoMinorVersionUpgrade": false,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "license-included",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "test-se-2017",
                "Status": "pending-apply"
            }
```

```
        ],
        "CharacterSetName": "SQL_Latin1_General_CP1_CI_AS",
        "SecondaryAvailabilityZone": "us-west-2c",
        "PubliclyAccessible": true,
        "StorageType": "gp2",

        ...output omitted...

        "DeletionProtection": false,
        "AssociatedRoles": [],
        "MaxAllocatedStorage": 1000
    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「Amazon RDS DB インスタンスを変更する」を参照してください。

例 2: VPC セキュリティグループを DB インスタンスと関連付けるには

次の modify-db-instance の例では、特定の VPC セキュリティグループを関連付け、DB インスタンスから DB セキュリティグループを削除します。

```
aws rds modify-db-instance \
    --db-instance-identifier dbName \
    --vpc-security-group-ids sg-ID
```

出力:

```
{
"DBInstance": {
    "DBInstanceIdentifier": "dbName",
    "DBInstanceClass": "db.t3.micro",
    "Engine": "mysql",
    "DBInstanceStatus": "available",
    "MasterUsername": "admin",
    "Endpoint": {
        "Address": "dbName.abcdefghijk.us-west-2.rds.amazonaws.com",
        "Port": 3306,
        "HostedZoneId": "ABCDEFGHIJK1234"
    },
    "AllocatedStorage": 20,
    "InstanceCreateTime": "2024-02-15T00:37:58.793000+00:00",
    "PreferredBackupWindow": "11:57-12:27",
```

```
    "BackupRetentionPeriod": 7,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-ID",
            "Status": "active"
        }
    ],
    ... output omitted ...
    "MultiAZ": false,
    "EngineVersion": "8.0.35",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",

    ... output omitted ...
    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「セキュリティグループによるアクセス制御」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「ModifyDBInstance」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master
 password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
```

```
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.xAPI リファレンスの「ModifyDBInstance」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「ModifyDBInstance」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ModifyDBParameterGroup** を使用する

次のサンプルコードは、ModifyDBParameterGroup を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Update a DB parameter group. Use the action
DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
```

```
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- API の詳細については、AWS SDK for .NET API リファレンス の「ModifyDBParameterGroup」を参照してください。

C++

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);
```

```
        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ModifyDBParameterGroup」を参照してください。

CLI

AWS CLI

DB パラメータグループを変更するには

次の modify-db-parameter-group の例は、DB パラメータグループの clr enabled パラメータの値を変更します。--apply-immediately パラメータを使用することで、次のメンテナンスウィンドウを待つことなく、DB パラメータグループをすぐに変更することができます。

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

出力:

```
{
    "DBParameterGroupName": "test-sqlserver-se-2017"
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB パラメータグループのパラメータの変更」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンス の「ModifyDBParameterGroup」を参照してください。

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
```

```
 } else {
  return nil
 }
}
```

- API の詳細については、AWS SDK for Go API リファレンス の「ModifyDBParameterGroup」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンス の「ModifyDBParameterGroup」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
```

```
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「ModifyDBParameterGroup」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " it_parameters - table containing parameter objects with:
    "   - parametername = 'max_connections'
    "   - parametervalue = '100'
```

```
    "   - applymethod = 'immediate' or 'pending-reboot'
    TRY.
        oo_result = lo_rds->modifydbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname
          it_parameters           = it_parameters ).
        MESSAGE 'DB parameter group modified.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「ModifyDBParameterGroup」を参照してください。

Swift

SDK for Swift

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import AWSRDS

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
```

```
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- API の詳細については、AWS SDK for Swift API リファレンスの「ModifyDBParameterGroup」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **RebootDBInstance** を使用する

次のサンプルコードは、RebootDBInstance を使用する方法を説明しています。

CLI

AWS CLI

DB インスタンスを再起動するには

次の reboot-db-instance の例は、指定された DB インスタンスを再起動します。

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier test-mysql-instance
```

出力:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "rebooting",
        "MasterUsername": "admin",
        "Endpoint": {
            "Address": "test-mysql-instance.############.us-
west-2.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z1PVIF0EXAMPLE"
        },

    ... output omitted...

    }
}
```

詳細については、「Amazon RDS ユーザーガイド」の「DB インスタンスの再起動」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「RebootDBInstance」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```

```
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「RebootDBInstance」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用した Amazon RDS のシナリオ

次のコード例は、AWS SDK を使用して Amazon RDS で一般的なシナリオを実装する方法を示しています。これらのシナリオでは、Amazon RDS 内で複数の関数を呼び出すか、その他の AWS のサービス と組み合わせることで、特定のタスクを実行する方法を説明しています。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- Aurora Serverless 作業項目トラッカーの作成
- リレーショナルデータベースインスタンスの作成

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

.NET

SDK for .NET

AWS SDK for .NET を使用して Amazon Aurora データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを E メールで送信するウェブアプリケーションを作成する方法を示します。この例では、React.js で構築されたフロントエンドを使用して RESTful .NET バックエンドと対話します。

- React ウェブアプリケーションを AWS のサービスと統合します。
- Aurora テーブルの項目を一覧表示、更新、削除します。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 付属の AWS CloudFormation スクリプトでサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

C++

SDK for C++

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

Amazon Aurora Serverless データをクエリし、React アプリケーションで使用するための C++ REST API の完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

Java

SDK for Java 2.x

Amazon RDS データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を説明します。

Amazon Aurora サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

完全なソースコードと JDBC API を使用する例のセットアップと実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript (v3) を使用して Amazon Aurora データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを E メールで送信する

ウェブアプリケーションを作成する方法を示します。この例では、React.js で構築されたフロントエンドを使用して Express Node.js バックエンドと対話します。

- React.js ウェブアプリケーションを AWS のサービス と統合します。
- Aurora テーブルの項目を一覧表示、追加、更新します。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 付属の AWS CloudFormation スクリプトでサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

Kotlin

SDK for Kotlin

Amazon RDS データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を説明します。

Amazon Aurora サーバーレスデータをクエリする Spring REST API と React アプリケーションで使用するための完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

PHP

SDK for PHP

AWS SDK for PHP を使用して Amazon RDS データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを E メールで送信するウェブアプリケーションを作成する方法を示します。この例では、React.js で構築されたフロントエンドを使用して RESTful PHP バックエンドと対話します。

- React.js ウェブアプリケーションを AWS のサービスと統合します。
- Amazon RDS テーブル内の項目の一覧表示、追加、更新、削除を行います。
- Amazon SES を使用して、フィルター処理された作業項目の E メールレポートを送信します。
- 付属の AWS CloudFormation スクリプトでサンプルリソースをデプロイおよび管理します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

Python

SDK for Python (Boto3)

AWS SDK for Python (Boto3) を使用して Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを E メールで送信する REST サービスを作成する方法を示します。この例では、Flask ウェブフレームワークを使用して HTTP ルーティングを処理し、React ウェブページと統合して完全に機能するウェブアプリケーションを提供します。

- AWS のサービス と統合する Flask REST サービスを構築します。
- Aurora Serverless データベースに保存されている作業項目の読み取り、書き込み、更新を行います。

- データベース認証情報を含む AWS Secrets Manager シークレットを作成し、それを使用してデータベースへの呼び出しを認証します。
- Amazon SES を使用して作業項目のレポートを E メールで送信します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

リレーショナルデータベースインスタンスの作成

次のコード例は、以下の操作方法を示しています。

- ネットワークコンポーネントをセットアップする
- DB サブネットグループを作成する
- DB インスタンスを作成する
- リソースをクリーンアップする

Bash

AWS CLIBash スクリプトを使用した

> **Note**
>
> GitHub には、その他のリソースもあります。デベロッパーチュートリアルのサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash
```

```
# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
```

```
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
```

```
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
```

```
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
```

```
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')
```

```
echo "============================================="
echo "DB Instance successfully created!"
echo "============================================="
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "============================================="

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"
```

```
    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - 待ちます。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon RDS のサーバーレス例

次のコード例は、AWS SDK で Amazon RDS を使用する方法を示します。

例

- Lambda 関数での Amazon RDS データベースへの接続

Lambda 関数での Amazon RDS データベースへの接続

次のコード例は、RDS データベースに接続する Lambda 関数を実装する方法を示しています。この関数は、シンプルなデータベースリクエストを実行し、結果を返します。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

.NET を使用した Lambda 関数での Amazon RDS データベースへの接続。

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
```

```
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                  $"Pwd={authToken};";


        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";
```

```
            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The
 sum is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server
 error" })
        };
    }
}
```

Go

SDK for Go V2

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Go を使用した Lambda 関数での Amazon RDS データベースへの接続

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")
```

```
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}
```

```
func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Java を使用した Lambda 関数での Amazon RDS データベースへの接続

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();
```

```
        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();
```

```
        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

JavaScript を使用した Lambda 関数での Amazon RDS データベースへの接続

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
```

```
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
```

```
  }
};
```

TypeScript を使用した Lambda 関数での Amazon RDS データベースへの接続

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
```

```
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK for PHP

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

PHP を使用した Lambda 関数での Amazon RDS データベースへの接続

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new
 AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }
```

```
    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();
```

```
        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Python を使用した Lambda 関数での Amazon RDS データベースへの接続

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
```

```
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK for Ruby

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Ruby を使用した Lambda 関数での Amazon RDS データベースへの接続

```
# Ruby code here.

require 'aws-sdk-rds'
```

```
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
```

```
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。サーバーレスサンプルリポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Rust を使用した Lambda 関数での Amazon RDS データベースへの接続

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
```

```
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、このサービスを AWS SDK で使用する を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon RDS でのセキュリティ

AWS ではクラウドのセキュリティが最優先事項です。AWS のお客様はセキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャから利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ - 「AWS」は、「AWS」クラウドで「AWS」のサービスを実行するインフラストラクチャを保護する責任を負います。また、「AWS」は、使用するサービスを安全に提供します。「AWS」 コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon RDS (Aurora) に適用するコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ - お客様の責任は使用する AWS のサービスによって決まります。また、お客様は、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Amazon RDS を使用する際に共有責任モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために Amazon RDS を設定する方法を示します。また Amazon RDS リソースのモニタリングやセキュリティ確保に役立つ他の AWS サービスの使用方法についても確認頂けます。

DB インスタンス上の Amazon RDS リソースとデータベースへのアクセスを管理できます。アクセスの管理に使用する方法は、ユーザーが Amazon RDS で実行する必要のあるタスクのタイプによって異なります。

- Amazon VPC サービスに基づき、仮想プライベートクラウド (VPC) 内で DB インスタンスを実行して、ネットワークアクセス制御を最大限に拡張します。VPC での DB インスタンスの作成の詳細については、「Amazon VPC と Amazon RDS」を参照してください。
- AWS Identity and Access Management (IAM) ポリシーを使用して、どのユーザーが Amazon RDS リソースの管理を許可されるかを決定するアクセス許可を割り当てます。例えば、IAM を使用して、いずれのユーザーが DB インスタンスの作成、情報入手、変更、削除、リソースのタグ付け、セキュリティグループの変更を許可されるかを決定します。

- セキュリティグループを使用して、どの IP アドレスまたは Amazon EC2 インスタンスが DB インスタンス上のデータベースに接続できるかを制御します。DB インスタンスを初めて作成すると、そのインスタンスのファイアウォールにより、関連付けられるセキュリティグループによって指定されたルールに従ったアクセスを除き、データベースへのアクセスはすべて禁止されます。
- Db2、MySQL、MariaDB、PostgreSQL、Oracle、または Microsoft SQL Server のデータベースエンジンを実行している DB インスタンスと Secure Socket Layer (SSL) または Transport Layer Security (TLS) の接続を使用します。DB インスタンスで SSL/TLS を使用する方法の詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
- Amazon RDS 暗号化を使用して、DB インスタンスおよび保管時のスナップショットのセキュリティを確保します。Amazon RDS 暗号化は、業界スタンダードの AES-256 暗号化アルゴリズムを使用して、DB インスタンスをホストしているサーバーでデータを暗号化します。詳細については、「Amazon RDS リソースの暗号化」を参照してください。
- Oracle DB インスタンスではネットワーク暗号化と Transparent Data Encryption を使用します。詳細については、「Oracle ネイティブネットワーク暗号化」と「Oracle Transparent Data Encryption」を参照してください。
- DB エンジンのセキュリティ機能を使用して、DB インスタンスのデータベースにログインできるユーザーを制御します。これらの機能は、データベースがローカルネットワーク上にあるかのように動作します。

> **Note**
>
> 目的のユースケースに対してのみ、セキュリティを設定する必要があります。Amazon RDS で管理されるプロセス用にセキュリティアクセスを設定する必要はありません。このプロセスには、バックアップの作成、プライマリ DB インスタンスとリードレプリカの間のデータのレプリケートなどがあります。

Amazon RDS リソースや DB インスタンス上のデータベースに対するアクセスの管理の詳細については、以下のトピックを参照してください。

トピック

- Amazon RDS でのデータベース認証
- Amazon RDS および AWS Secrets Manager によるパスワード管理
- Amazon RDS でのデータ保護

- Amazon RDS での Identity and Access Management
- Amazon RDS でのログ記録とモニタリング
- Amazon RDS のコンプライアンス検証
- Amazon RDS の耐障害性
- Amazon RDS でのインフラストラクチャセキュリティ
- Amazon RDS API とインターフェイス VPC エンドポイント (AWS PrivateLink)
- Amazon RDS のセキュリティのベストプラクティス
- セキュリティグループによるアクセス制御
- マスターユーザーアカウント権限
- Amazon RDS のサービスにリンクされたロールの使用
- Amazon VPC と Amazon RDS

Amazon RDS でのデータベース認証

Amazon RDS は、データベースユーザーを認証するいくつかの方法をサポートしています。

パスワード、Kerberos、および IAM データベース認証では、データベースに対する認証にはさまざまな方法が使用されます。したがって、特定のユーザーは、1 つの認証方法のみを使用してデータベースにログインできます。

PostgreSQL の場合は、特定のデータベースのユーザーに対して、次のロール設定の 1 つだけを使用します。

- IAM データベース認証を使用するには、`rds_iam`ロールをユーザーに割り当てます。
- Kerberos 認証を使用するには、`rds_ad`ロールをユーザーに割り当てます。
- パスワード認証を使用するには、`rds_iam` または `rds_ad` ロールをユーザーに割り当てないでください。

ネストされた許可アクセスによって直接的または間接的に PostgreSQL データベースのユーザーに `rds_iam` ロールと `rds_ad` ロールを両方を割り当てないでください。`rds_iam`ロールがマスターユーザーに追加されると、IAM 認証はパスワード認証よりも優先されるため、マスターユーザーは IAM ユーザーとしてログインする必要があります。

> **Important**
>
> アプリケーションではマスターユーザーを直接使用しないことを強くお勧めします。代わりに、アプリケーションに必要な最小の特権で作成されたデータベースユーザーを使用するというベストプラクティスに従ってください。

トピック

- パスワード認証
- IAM データベース認証
- Kerberos 認証

パスワード認証

パスワード認証を使用すると、データベースがユーザーアカウントのすべての管理を行います。DB エンジンがパスワードを指定するのに必要な正しい句を使用して、`CREATE USER` などの SQL 文でユーザーを作成します。例えば、MySQL の文は `CREATE USER ## IDENTIFIED BY #####` となりますが、PostgreSQLでは `CREATE USER ## WITH PASSWORD #####` となります。

パスワード認証を使用すると、データベースがユーザーアカウントを制御および認証します。DB エンジンに強力なパスワード管理機能がある場合は、セキュリティを強化できます。ユーザーコミュニティが小規模である場合は、パスワード認証を使用すると、データベース認証が管理しやすくなります。この場合、クリアテキストパスワードが生成されるため、AWS Secrets Manager との統合によってセキュリティが強化されます。

Amazon RDS での Secrets Manager の使用の詳細については、「AWS Secrets Manager ユーザーガイド」の「基本シークレットの作成」と「サポートされている Amazon RDS データベースのシークレットのローテーション」を参照してください。カスタムアプリケーションにおいてシークレットをプログラムで取得する方法については、AWS Secrets Manager ユーザーガイドの「シークレット値の取得」を参照してください。

IAM データベース認証

AWS Identity and Access Management (IAM) データベース認証を使用して、DB インスタンスを認証できます。この認証方法では、DB インスタンスに接続するときにパスワードを使用する必要はありません。代わりに、認証トークンを使用します。

特定の DB エンジンの可用性など、IAM データベース認証の詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。

Kerberos 認証

Amazon RDS で、Kerberos と Microsoft Active Directory を使用した、データベースユーザーの外部認証がサポートされるようになりました。Kerberos は、ネットワーク経由でパスワードを送信する必要をなくすためにチケットと対称キー暗号化を使用するネットワーク認証プロトコルです。Kerberos は Active Directory に組み込まれており、データベースなどのネットワークリソースに対するユーザー認証を行えるように設計されています。

Amazon RDS での Kerberos と Active Directory のサポートにより、データベースユーザーのシングルサインオンおよび一元化認証という利点が得られます。ユーザー資格情報を Active Directory に保持できます。Active Directory には、複数の DB インスタンスの資格情報を保存し、管理する一元的な場所が用意されています。

セルフマネージド Active Directory の認証情報を使用するには、DB インスタンスが参加している Directory Service for Microsoft Active Directory との信頼関係を確立する必要があります。

RDS for PostgreSQL および RDS for MySQL は、フォレスト全体の認証または選択的認証による一方向および双方向のフォレスト信頼関係をサポートしています。

シナリオによっては、外部信頼関係に Kerberos 認証を設定できます。これには、セルフマネージド Active Directory に追加の設定が必要です。これには、Kerberos フォレストの検索順序が含まれますが、これらに限定されません。

Microsoft SQL Server および PostgreSQL DB インスタンスは、一方向および双方向のフォレスト信頼関係をサポートしています。Oracle DB インスタンスは、一方向と双方向の外部およびフォレストの信頼関係をサポートしています。詳細については、Directory Service 管理ガイドの「信頼関係を作成する場合」を参照してください。

特定の DB エンジンを使用した Kerberos 認証については、以下を参照してください。

- RDS for SQL Server による AWS Managed Active Directory の操作
- Amazon RDS for MySQL での Kerberos 認証の使用
- Amazon RDS for Oracle の Kerberos 認証の設定
- Amazon RDS for PostgreSQL で Kerberos 認証を使用する
- RDS for Db2 による AWS Managed Active Directory の操作 .

> **Note**
>
> 現在、Kerberos 認証は MariaDB DB インスタンスではサポートされていません。

Amazon RDS および AWS Secrets Manager によるパスワード管理

Amazon RDS は Secrets Manager と統合して、DB インスタンスとマルチ AZ DB クラスターのマスターユーザーパスワードを管理します。

トピック

- Secrets Manager と Amazon RDS の統合に関する制限事項
- AWS Secrets Manager を使用したマスターユーザーパスワード管理の概要
- Secrets Manager でマスターユーザーパスワードを管理する利点
- Secrets Manager の統合に必要なアクセス許可
- RDS によるマスターユーザーパスワードの管理の強化AWS Secrets Manager
- Secrets Manager による DB インスタンスのマスターユーザーパスワードの管理
- Secrets Manager を使用した RDS for Oracle テナントデータベースのマスターユーザーパスワードの管理
- Secrets Manager によるマルチ AZ DB クラスターのマスターユーザーパスワードの管理
- DB インスタンスのマスターユーザーパスワードシークレットのローテーション
- マルチ AZ DB クラスターのマスターユーザーパスワードシークレットのローテーション
- DB インスタンスのシークレットに関する詳細を表示する
- マルチ AZ DB クラスターのシークレットに関する詳細の表示
- テナントデータベースのシークレットに関する詳細の表示
- 利用可能なリージョンとバージョン

Secrets Manager と Amazon RDS の統合に関する制限事項

Secrets Manager によるマスターユーザーパスワードの管理は、以下の機能ではサポートされていません。

- ソース DB または DB クラスターが Secrets Manager で認証情報を管理する場合のリードレプリカの作成。これは、RDS for SQL Server を除くすべての DB エンジンに適用されます。
- Amazon RDS ブルー/グリーンデプロイ
- Amazon RDS Custom
- Oracle Data Guard のスイッチオーバー

AWS Secrets Manager を使用したマスターユーザーパスワード管理の概要

AWS Secrets Manager を使用すると、コード内のハードコードされた認証情報 (データベースパスワードを含む) を Secrets Manager への API コールで置き換えて、プログラムでシークレットを取得することができます。Secrets Manager の詳細については、AWS Secrets Manager ユーザーガイドを参照してください。

データベースシークレットを Secrets Manager に保存すると、AWS アカウント に料金が発生します。料金については、「AWS Secrets Manager 料金表」を参照してください。

次のいずれかのオペレーションを実行するときに、RDS が Amazon RDS DB インスタンスまたはマルチ AZ DB クラスターのマスターユーザーパスワードを Secrets Manager で管理するように指定できます。

- DB インスタンスを作成する
- マルチ AZ DB クラスターを作成する
- RDS for Oracle CDB でテナントデータベースを作成する
- DB インスタンスの変更
- マルチ AZ DB クラスターを変更する
- テナントデータベースを変更する (RDS for Oracle のみ)
- DB インスタンスを Amazon S3 から復元する
- DB インスタンスをスナップショットまたはポイントインタイムに復元する (RDS for Oracle のみ)

RDS が Secrets Manager でマスターユーザーパスワードを管理するように指定すると、RDS はパスワードを生成して Secrets Manager に保存します。シークレットを直接操作して、マスターユーザーの認証情報を取得できます。また、カスタマーマネージドキーを指定してシークレットを暗号化したり、Secrets Manager が提供する KMS キーを使用したりすることもできます。

RDS はシークレットの設定を管理し、デフォルトで 7 日ごとにシークレットをローテーションします。ローテーションスケジュールなど、一部の設定を変更できます。Secrets Manager でシークレットを管理する DB インスタンスを削除すると、シークレットとそれに関連するメタデータも削除されます。

シークレット内の認証情報を使用して DB インスタンスまたはマルチ AZ DB クラスターに接続するには、Secrets Manager からシークレットを取得します。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager からのシークレットの取得、AWS Secrets Manager シークレットの認証情報を使用して SQL データベースに接続する」を参照してください。

Secrets Manager でマスターユーザーパスワードを管理する利点

Secrets Manager で RDS マスターユーザーのパスワードを管理することには、次の利点があります。

- RDS はデータベース認証情報を自動的に生成します。
- RDS はデータベース認証情報を AWS Secrets Manager に自動的に保存および管理します。
- RDS は、アプリケーションを変更することなく、データベースの認証情報を定期的にローテーションします。
- Secrets Manager は、データベースの認証情報を人間のアクセスやプレーンテキスト表示から保護します。
- Secrets Manager では、データベース接続用のシークレット内のデータベース認証情報を取得できます。
- Secrets Manager では、IAM を使用してシークレット内のデータベース認証情報へのアクセスをきめ細かく制御できます。
- 必要に応じて、さまざまな KMS キーを使用して、データベースの暗号化を資格情報の暗号化から分離できます。
- データベース認証情報の手動管理やローテーションが不要になります。
- AWS CloudTrail と Amazon CloudWatch を使用すると、データベースの認証情報を簡単にモニタリングできます。

Secrets Manager のメリットの詳細については、「AWS Secrets Manager ユーザーガイド」を参照してください。

Secrets Manager の統合に必要なアクセス許可

Secrets Manager の統合に関連するオペレーションを実行するには、ユーザーが必要なアクセス許可を持っている必要があります。必要な特定のリソースの API オペレーションを実行するためのアクセス許可を付与する IAM ポリシーを作成できます。その後、これらのポリシーを、それらのアクセス許可を必要とする IAM アクセス許可セットまたはロールにアタッチできます。詳細については、「Amazon RDS での Identity and Access Management」を参照してください。

作成、変更、または復元オペレーションの場合、Amazon RDS が Secrets Manager でマスターユーザーパスワードを管理するように指定するユーザーには、次のオペレーションを実行するアクセス許可が必要です。

- kms:DescribeKey
- secretsmanager:CreateSecret
- secretsmanager:TagResource

kms:DescribeKey アクセス許可は、MasterUserSecretKmsKeyId のカスタマーマネージドキーにアクセスし、aws/secretsmanager を記述するために必要です。

作成、変更、または復元オペレーションの場合、Secrets Manager でシークレットを暗号化するカスタマーマネージドキーを指定するユーザーには、次のオペレーションを実行するアクセス許可が必要です。

- kms:Decrypt
- kms:GenerateDataKey
- kms:CreateGrant

変更オペレーションの場合、Secrets Manager でマスターユーザーパスワードをローテーションするユーザーには、次のオペレーションを実行するアクセス許可が必要です。

- secretsmanager:RotateSecret

RDS によるマスターユーザーパスワードの管理の強化AWS Secrets Manager

IAM 条件キーを使用して、AWS Secrets Manager のマスターユーザーパスワードの RDS 管理を実施できます。次のポリシーでは、マスターユーザーパスワードが Secrets Manager で RDS によって管理されていない限り、ユーザーが DB インスタンスまたは DB クラスターを作成または復元するまたはテナントデータベースを作成または変更することはできません。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
```

```
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3",
                       "rds:RestoreDBInstanceFromDBSnapshot",
 "rds:RestoreDBInstanceToPointInTime", "rds:CreateTenantDatabase",
                       "rds:ModifyTenantDatabase"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> このポリシーは、作成時に AWS Secrets Manager でのパスワード管理を強制します。ただし、インスタンスを変更することで、Secrets Manager の統合を無効にして、マスターパスワードを手動で設定することができます。
> これを防ぐには、ポリシーのアクションブロックに `rds:ModifyDBInstance`、`rds:ModifyDBCluster` を含めます。これにより、Secrets Manager 統合が有効になっていない既存のインスタンスには、以降の変更ができなくなることに注意してください。

IAM ポリシーでの条件キーの使用の詳細については、「Amazon RDS のポリシー条件キー」および「ポリシー例: 条件キーの使用」を参照してください。

Secrets Manager による DB インスタンスのマスターユーザーパスワードの管理

以下のアクションを実行すると、Secrets Manager でマスターユーザーパスワードの RDS 管理を設定できます。

- Amazon RDS DB インスタンスの作成
- Amazon RDS DB インスタンスを変更する
- Amazon RDS for MySQL DB インスタンスへのバックアップの復元

- DB インスタンスへの復元 (RDS for Oracle のみ)
- Amazon RDS の DB インスタンスを特定の時点に復元する (RDS for Oracle のみ)

上記のオペレーションは、RDS コンソール、AWS CLI、または RDS API を使用して実行できます。

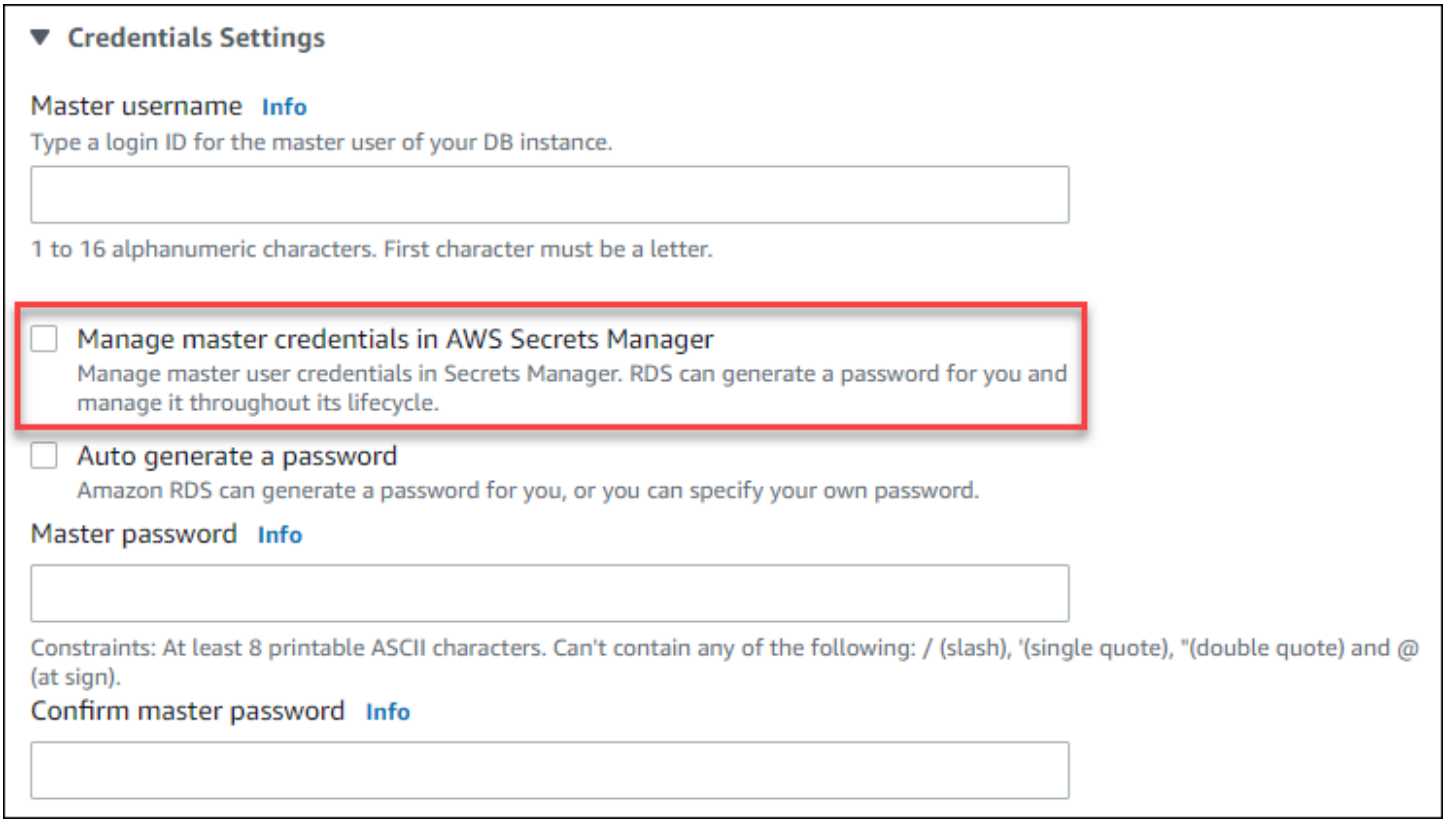
コンソール

RDS コンソールで DB インスタンスを作成または変更する手順に従います。

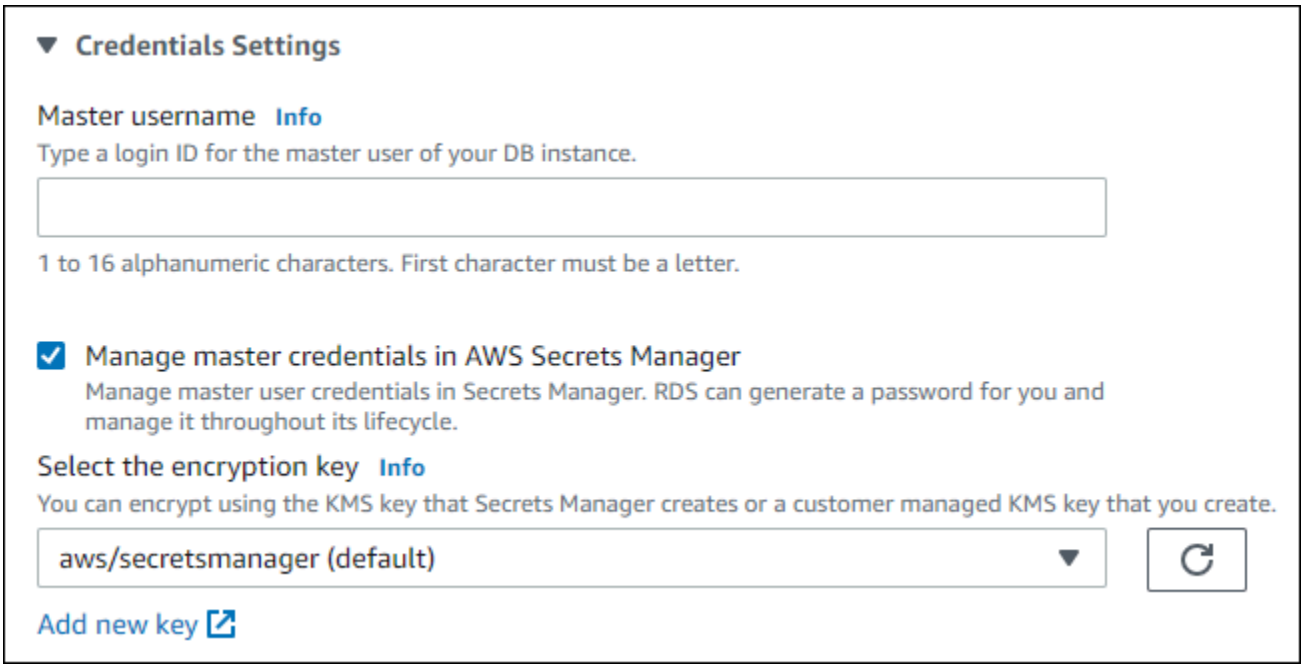
- DB インスタンスの作成
- Amazon RDS DB インスタンスを変更する
- Amazon S3 から新しい MySQL DB インスタンスにデータをインポートする

RDS コンソールを使用してこれらのオペレーションのいずれかを実行する場合、マスターユーザーパスワードを RDS が Secrets Manager で管理するように指定できます。DB インスタンスを作成または復元する場合、[認証情報設定] で [AWS Secrets Manager でマスター認証情報を管理する] を選択します。DB インスタンスを変更するときは、[設定] で [AWS Secrets Manager でマスター認証情報を管理する] を選択します。

以下の図は、DB インスタンスを作成または復元するときの AWS Secrets Manager でマスター認証情報を管理する 設定の例です。



このオプションを選択すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。



シークレットは、Secrets Manager が提供する KMS キーを使用して暗号化するか、自分で作成したカスタマーマネージドキーを使用して暗号化するかを選択できます。RDS で DB インスタンスの

データベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。DB インスタンスの作成時に使用できる設定の詳細については、「DB インスタンスの設定」を参照してください。DB インスタンスを変更する際に使用できる設定の詳細については、「DB インスタンスの設定」を参照してください。

AWS CLI

Secrets Manager で RDS を使用してマスターユーザーパスワードを管理するには、以下のいずれかの AWS CLI コマンドで `--manage-master-user-password` オプションを指定します。

- create-db-instance
- modify-db-instance
- restore-db-instance-from-s3
- restore-db-instance-from-db-snapshot (RDS for Oracle のみ)
- restore-db-instance-to-point-in-time (RDS for Oracle のみ)

これらのコマンドで `--manage-master-user-password` オプションを指定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`--master-user-secret-kms-key-id` オプションを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS で DB インスタンスのデータベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。DB インスタンスの作成時に使用できる設定の詳細については、「DB インスタンスの設定」を参照してください。DB インスタンスを変更するときに使用できる設定の詳細については、「DB インスタンスの設定」を参照してください。

次の例では DB インスタンスを作成し、RDS が Secrets Manager でマスターユーザーパスワードを管理するように指定します。シークレットは、Secrets Manager によって提供される KMS キーを使用して暗号化されます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --engine mysql \
    --engine-version 8.0.39 \
    --db-instance-class db.r5b.large \
    --allocated-storage 200 \
    --master-username testUser \
    --manage-master-user-password
```

Windows の場合:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --engine mysql ^
    --engine-version 8.0.39 ^
    --db-instance-class db.r5b.large ^
    --allocated-storage 200 ^
    --master-username testUser ^
    --manage-master-user-password
```

RDS API

RDS が Secrets Manager のマスターユーザーパスワードを管理するように指定するには、次の RDS API オペレーションのいずれかで `ManageMasterUserPassword` パラメータを `true` に設定します。

- CreateDBInstance
- ModifyDBInstance
- RestoreDBInstanceFromS3
- RestoreDBInstanceFromSnapshot (RDS for Oracle のみ)
- RestoreDBInstanceToPointInTime (RDS for Oracle のみ)

これらのオペレーションのいずれかで `ManageMasterUserPassword` パラメータを `true` に設定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`MasterUserSecretKmsKeyId` パラメータを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS で DB インスタンスのデータベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

Secrets Manager を使用した RDS for Oracle テナントデータベースのマスターユーザーパスワードの管理

以下のアクションを実行すると、Secrets Manager でマスターユーザーパスワードの RDS 管理を設定できます。

- RDS for Oracle テナントデータベースを CDB インスタンスに追加する
- RDS for Oracle テナントデータベースの変更

前述のアクションを実行するために、RDS コンソール、AWS CLI、または RDS API を使用できます。

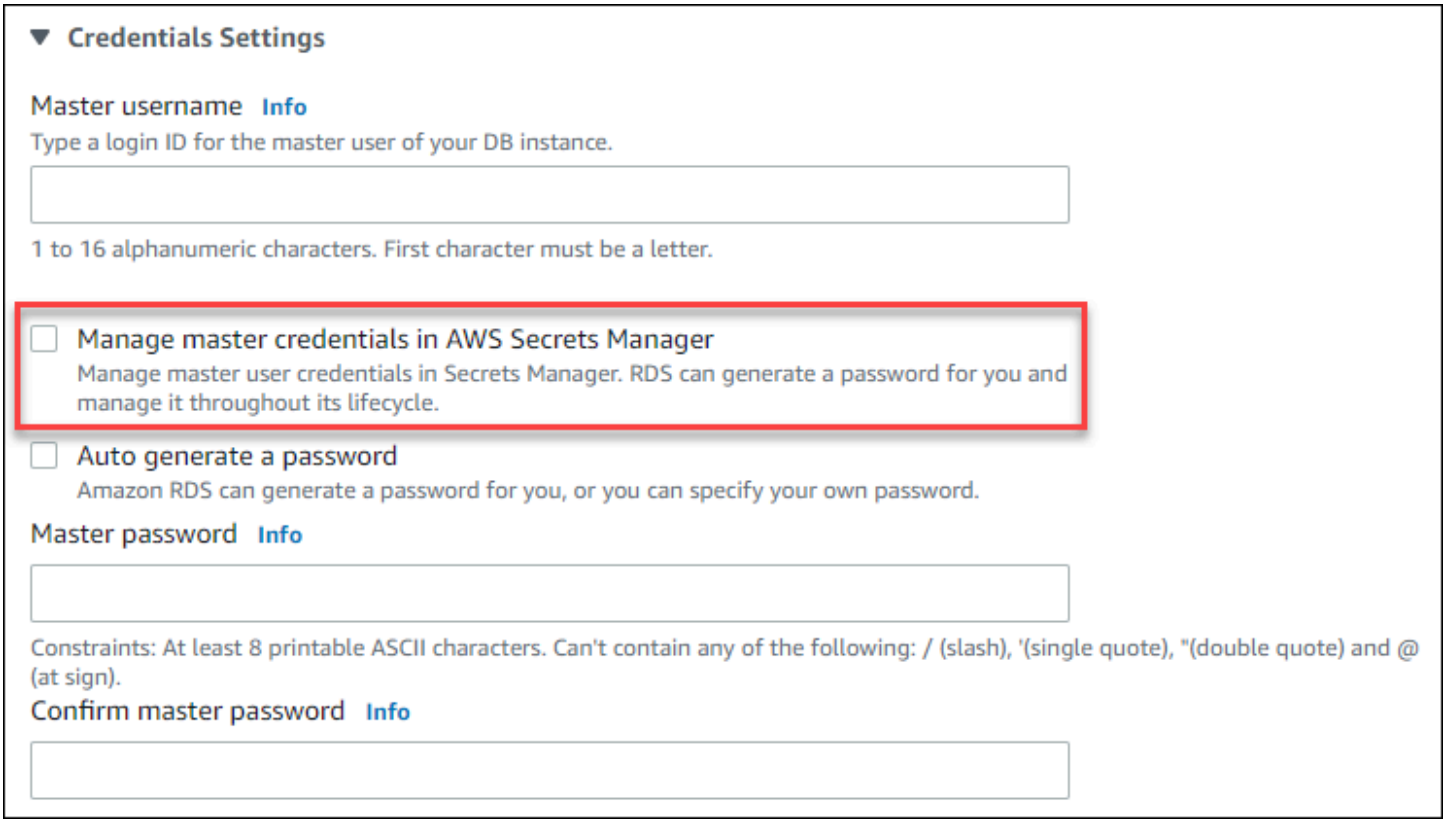
コンソール

RDS コンソールで RDS for Oracle テナントデータベースを作成または変更する手順に従います。

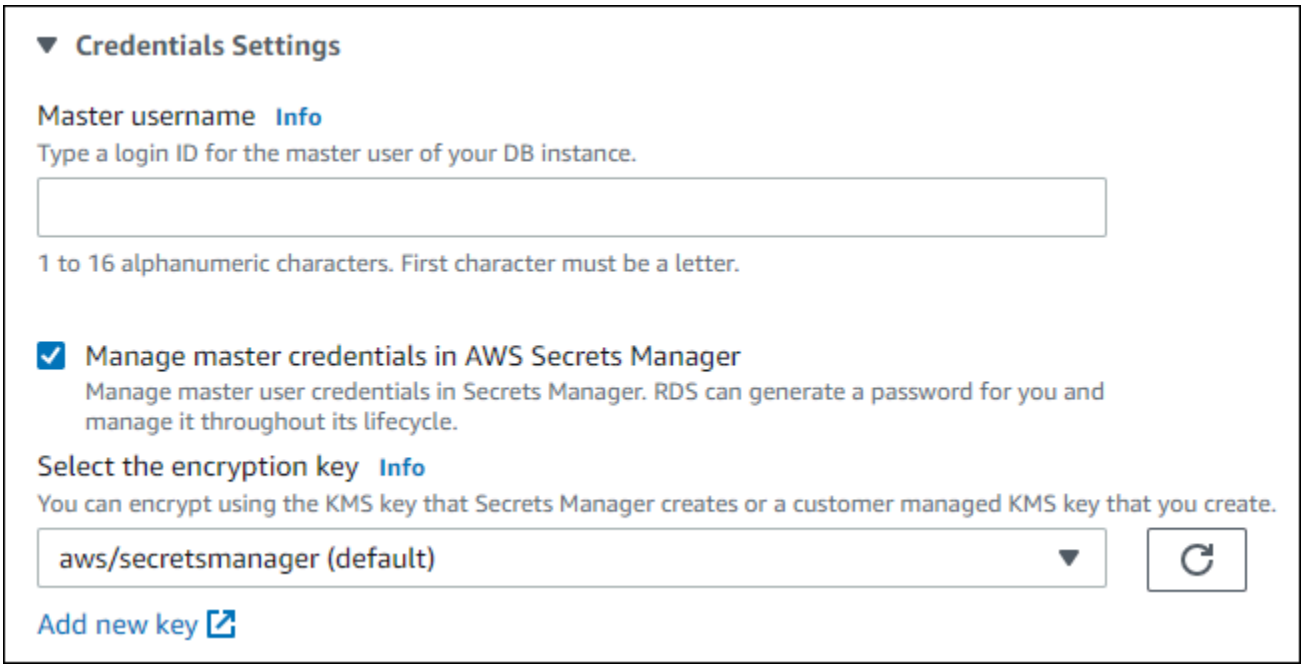
- RDS for Oracle テナントデータベースを CDB インスタンスに追加する
- RDS for Oracle テナントデータベースの変更

RDS コンソールを使用して前述のオペレーションのいずれかを実行する場合、RDS が Secrets Manager でマスターユーザーパスワードを管理するように指定できます。テナントデータベースを作成するときは、[認証情報設定] の [AWS Secrets Manager でマスター認証情報を管理する] を選択します。テナントデータベースを変更する場合は、[設定] で [AWS Secrets Manager でマスター認証情報を管理する] を選択します。

以下の図は、テナントデータベースを作成するときの [AWS Secrets Manager でマスター認証情報を管理する] 設定の例です。



このオプションを選択すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。



シークレットは、Secrets Manager が提供する KMS キーを使用して暗号化するか、自分で作成したカスタマーマネージドキーを使用して暗号化するかを選択できます。RDS でテナントデータベース

のデータベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。テナントデータベースの作成時に使用できる設定の詳細については、「DB インスタンスの設定」を参照してください。テナントデータベースの変更時に利用できる設定の詳細については、「DB インスタンスの設定」を参照してください。

AWS CLI

Secrets Manager で RDS を使用してマスターユーザーパスワードを管理するには、以下のいずれかの AWS CLI コマンドで `--manage-master-user-password` オプションを指定します。

- create-tenant-database
- modify-tenant-database

前述のコマンドで `--manage-master-user-password` オプションを指定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`--master-user-secret-kms-key-id` オプションを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS でテナントデータベースのデータベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。テナントデータベースの作成時に使用可能な設定の詳細については、「create-tenant-database」を参照してください。テナントデータベースを変更するときに使用できる設定の詳細については、「modify-tenant-database」を参照してください。

次の例では、RDS for Oracle テナントデータベースを作成し、RDS が Secrets Manager でマスターユーザーパスワードを管理するように指定します。シークレットは、Secrets Manager によって提供される KMS キーを使用して暗号化されます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-tenant-database --region us-east-1 \
```

```
    --db-instance-identifier my-cdb-inst \
    --tenant-db-name mypdb2 \
    --master-username mypdb2-admin \
    --character-set-name UTF-16 \
    --manage-master-user-password
```

Windows の場合:

```
aws rds create-tenant-database --region us-east-1 ^
    --db-instance-identifier my-cdb-inst ^
    --tenant-db-name mypdb2 ^
    --master-username mypdb2-admin ^
    --character-set-name UTF-16 ^
    --manage-master-user-password
```

RDS API

RDS が Secrets Manager のマスターユーザーパスワードを管理するように指定するには、次の RDS API オペレーションのいずれかで `ManageMasterUserPassword` パラメータを `true` に設定します。

- CreateTenantDatabase
- ModifyTenantDatabase

これらのオペレーションのいずれかで `ManageMasterUserPassword` パラメータを `true` に設定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`MasterUserSecretKmsKeyId` パラメータを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS でテナントデータベースのデータベース認証情報を管理したら、シークレットの暗号化で使用されている KMS キーを変更することはできません。

Secrets Manager によるマルチ AZ DB クラスターのマスターユーザーパスワードの管理

以下のアクションを実行すると、Secrets Manager でマスターユーザーパスワードの RDS 管理を設定できます。

- Amazon RDS 用のマルチ AZ DB クラスターの作成
- Amazon RDS のマルチ AZ DB クラスターの変更

これらのアクションを実行するには、RDS コンソール、AWS CLI、または RDS API を使用できます。

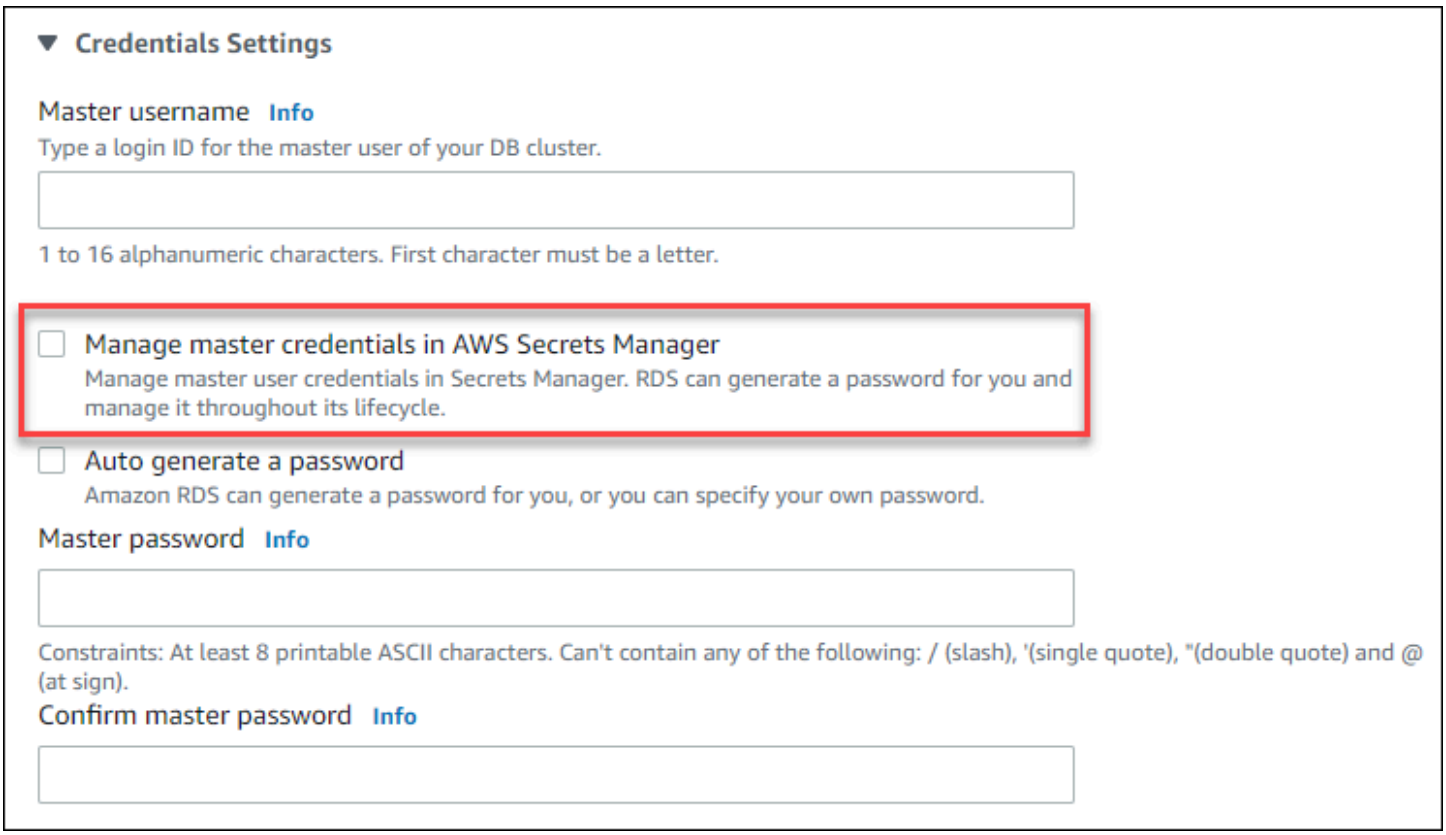
コンソール

RDS コンソールを使用してマルチ AZ DB クラスターを作成または変更する手順に従います。

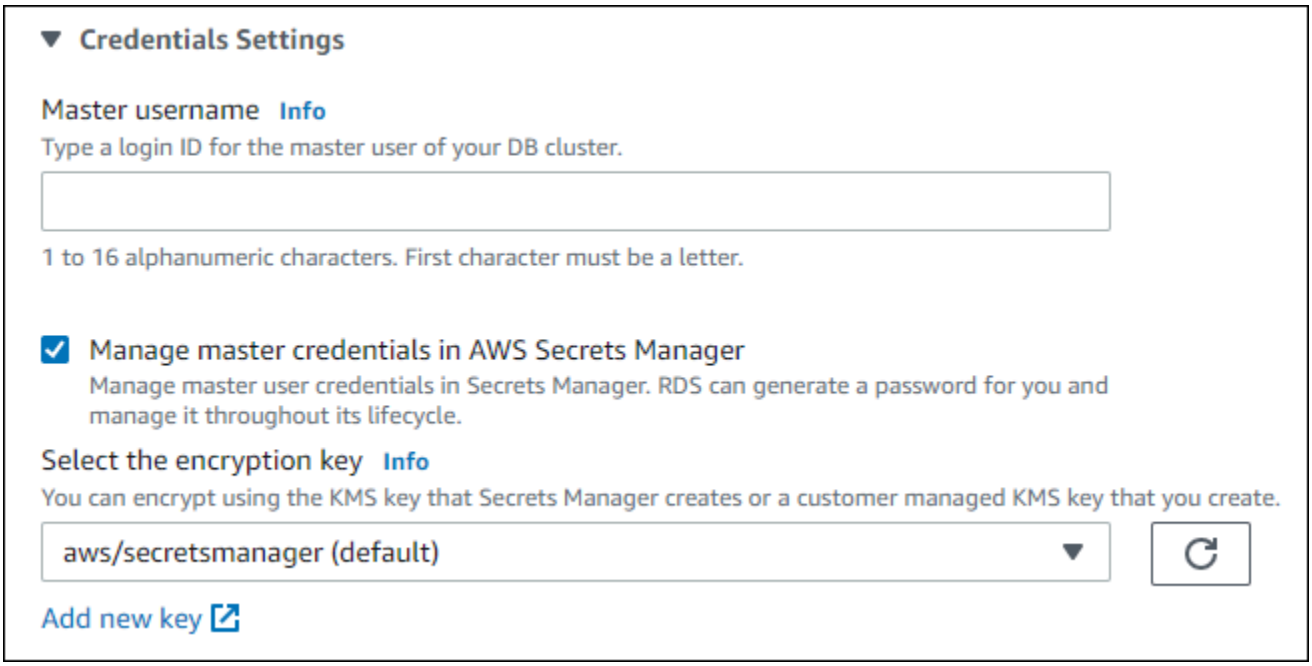
- DB クラスターの作成
- Amazon RDS のマルチ AZ DB クラスターの変更

RDS コンソールを使用してこれらのオペレーションのいずれかを実行する場合、マスターユーザーパスワードが RDS で Secrets Manager で管理されるように指定できます。これを行うには、DB クラスターを作成ときに、[Credential settings] (認証情報設定) で [Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択します。DB クラスターを変更する場合は、[Settings] (設定) で [Manage master credentials in AWS Secrets Manager] (でマスター認証情報を管理する) を選択します。

以下の図は、DB インスタンスを作成またはときの AWS Secrets Managerでマスター認証情報を管理する設定の例です。



このオプションを選択すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。



シークレットは、Secrets Manager が提供する KMS キーを使用して暗号化するか、自分で作成したカスタマーマネージドキーを使用して暗号化するかを選択できます。RDS が DB クラスターのデー

タベース認証情報を管理した後は、シークレットの暗号化に使用される KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。

各 マルチ AZ DB クラスターの作成時に使用できる設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。マルチ AZ DB クラスターの変更時に利用できる設定の詳細については、「マルチ AZ DB クラスターの変更の設定」を参照してください。

AWS CLI

RDS が Secrets Manager のマスターユーザーパスワードを管理するように指定するには、以下のいずれかの `--manage-master-user-password` コマンドでオプションを指定します。

- create-db-cluster
- modify-db-cluster

これらのコマンドで `--manage-master-user-password` オプションを指定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`--master-user-secret-kms-key-id` オプションを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS が DB クラスターのデータベース認証情報を管理した後は、シークレットの暗号化に使用される KMS キーを変更することはできません。

要件に合わせて他の設定を選択できます。

各 マルチ AZ DB クラスターの作成時に使用できる設定の詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。マルチ AZ DB クラスターの変更時に利用できる設定の詳細については、「マルチ AZ DB クラスターの変更の設定」を参照してください。

この例では、マルチ AZ DB クラスターを作成し、RDS が Secrets Manager でパスワードを管理するように指定しています。シークレットは、Secrets Manager によって提供される KMS キーを使用して暗号化されます。

Example

Linux、macOS、Unix の場合:

```
aws rds create-db-cluster \
   --db-cluster-identifier mysql-multi-az-db-cluster \
   --engine mysql \
   --engine-version 8.0.39  \
   --backup-retention-period 1  \
   --allocated-storage 4000 \
   --storage-type io1 \
   --iops 10000 \
   --db-cluster-instance-class db.r6gd.xlarge \
   --master-username testUser \
   --manage-master-user-password
```

Windows の場合:

```
aws rds create-db-cluster ^
   --db-cluster-identifier mysql-multi-az-db-cluster ^
   --engine mysql ^
   --engine-version 8.0.39 ^
   --backup-retention-period 1 ^
   --allocated-storage 4000 ^
   --storage-type io1 ^
   --iops 10000 ^
   --db-cluster-instance-class db.r6gd.xlarge ^
   --master-username testUser ^
   --manage-master-user-password
```

RDS API

RDS が Secrets Manager のマスターユーザーパスワードを管理するように指定するには、次のいずれかのオペレーションで `ManageMasterUserPassword` パラメータを `true` に設定します。

- CreateDBCluster
- ModifyDBCluster

これらのオペレーションのいずれかで `ManageMasterUserPassword` パラメータを `true` に設定すると、RDS はマスターユーザーパスワードを生成し、そのライフサイクル全体を通じて Secrets Manager で管理します。

シークレットを暗号化するには、カスタマーマネージドキーを指定するか、Secrets Manager によって提供されるデフォルトの KMS キーを使用できます。`MasterUserSecretKmsKeyId` パラメータを使用して、カスタマーマネージドキーを指定します。AWS KMS キー識別子は、KMS キーのキー ARN、キー ID、エイリアス ARN、またはエイリアス名です。別の AWS アカウント で KMS キーを使用するには、キー ARN またはエイリアス ARN を指定します。RDS が DB クラスターのデータベース認証情報を管理した後は、シークレットの暗号化に使用される KMS キーを変更することはできません。

DB インスタンスのマスターユーザーパスワードシークレットのローテーション

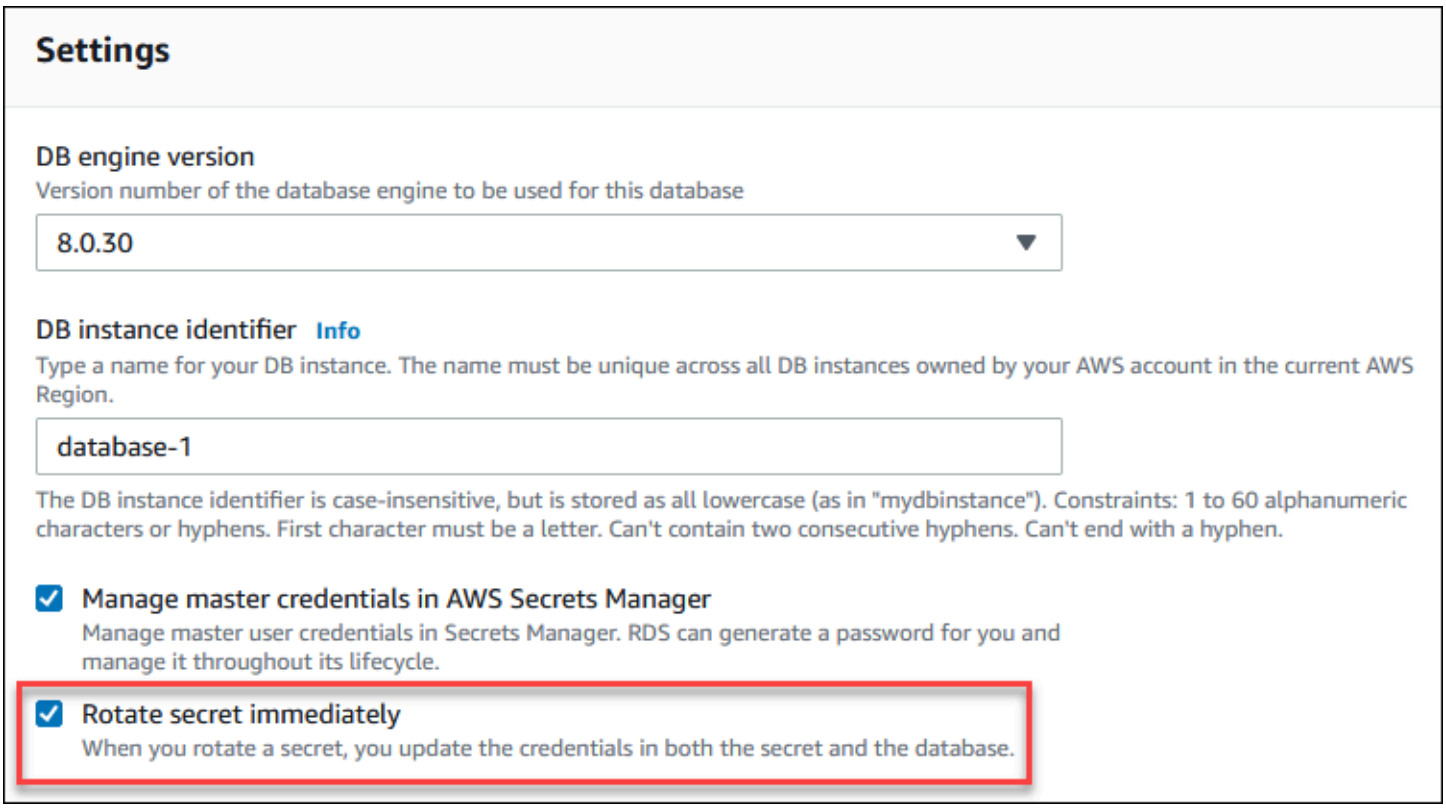
RDS がマスターユーザーパスワードシークレットをローテーションすると、Secrets Manager は既存のシークレットの新しいシークレットバージョンを生成します。新しいバージョンのシークレットには、新しいマスターユーザーパスワードが含まれています。Amazon RDS は DB インスタンスのマスターユーザーパスワードを、新しいシークレットバージョンのパスワードと一致するように変更します。

スケジュールされたローテーションを待つ代わりに、シークレットをすぐにローテーションできます。Secrets Manager でマスターユーザーのパスワードシークレットを更新するには、DB インスタンスを変更します。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

RDS コンソール、AWS CLI、または RDS API を使用して、マスターユーザーのパスワードシークレットをすぐに更新できます。新しいパスワードは常に 28 文字で、少なくとも 1 つの大文字と小文字、1 つの数字、1 つの句読点が含まれます。

コンソール

RDS コンソールを使用してマスターユーザーのパスワードシークレットをローテーションするには、DB インスタンスを変更し、[Settings] (設定) で [Rotate secret immediately] (シークレットを直ちにローテーションする) を選択します。



Amazon RDS DB インスタンスを変更する の RDS コンソールで DB インスタンスを変更する手順に従います。確認ページで [Apply immediately] (すぐに適用) を選択する必要があります。

AWS CLI

AWS CLI を使用してマスターユーザーパスワードシークレットをローテーションするには、modify-db-instance コマンドを使用して `--rotate-master-user-password` オプションを指定します。マスターパスワードをローテーションするときは、`--apply-immediately` オプションを指定する必要があります。

この例では、マスターユーザーパスワードシークレットをローテーションします。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --rotate-master-user-password \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --rotate-master-user-password ^
    --apply-immediately
```

RDS API

ModifyDBInstance オペレーションを使用して RotateMasterUserPassword パラメータを true に設定すると、マスターユーザーパスワードシークレットをローテーションできます。マスターパスワードを変更するときは、ApplyImmediately パラメータを true に設定する必要があります。

マルチ AZ DB クラスターのマスターユーザーパスワードシークレットのローテーション

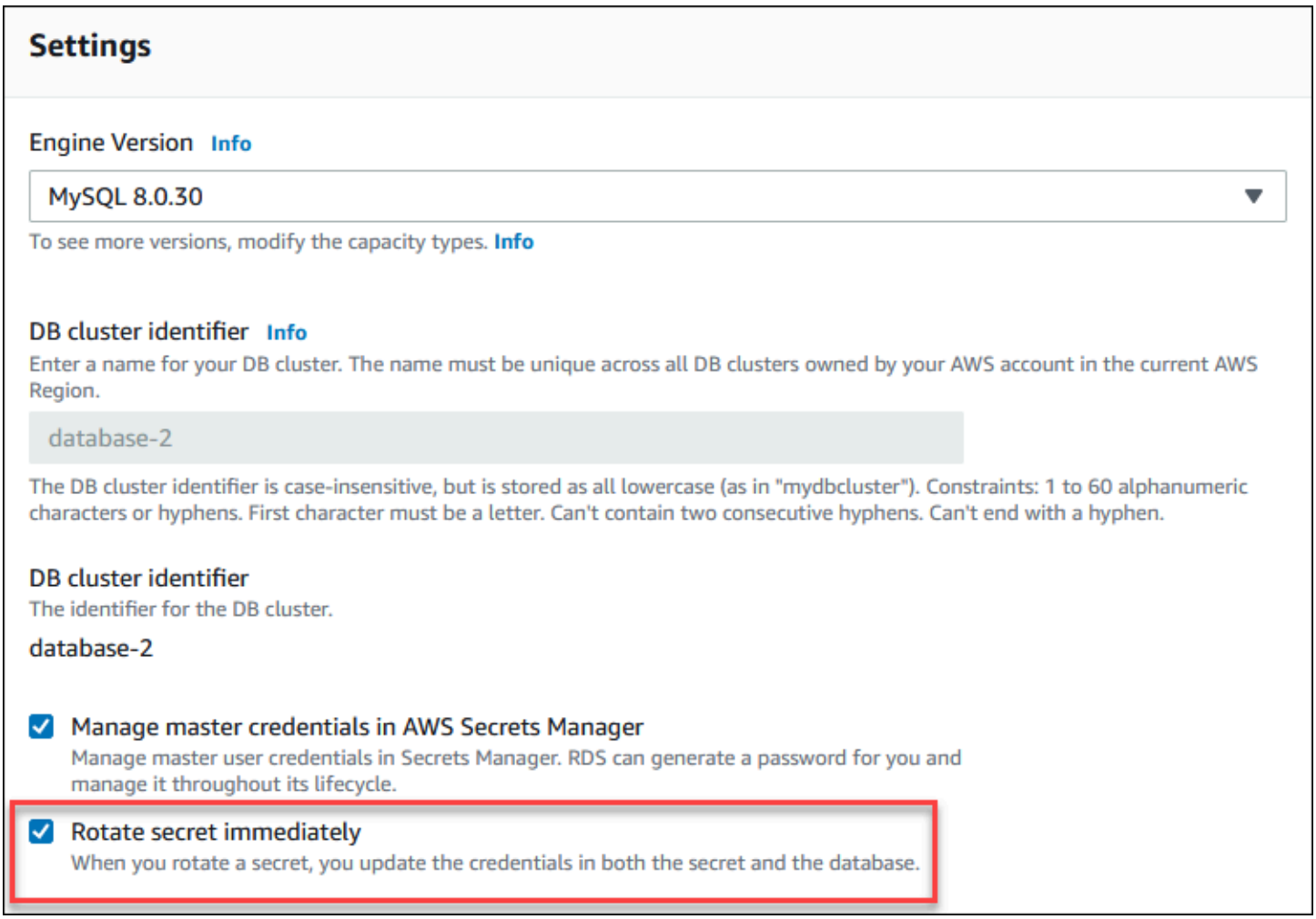
RDS がマスターユーザーパスワードシークレットをローテーションすると、Secrets Manager は既存のシークレットの新しいシークレットバージョンを生成します。新しいバージョンのシークレットには、新しいマスターユーザーパスワードが含まれています。Amazon RDS は、マルチ AZ DB クラスターのマスターユーザーパスワードを、新しいシークレットバージョンのパスワードと一致するように変更します。

スケジュールされたローテーションを待つ代わりに、シークレットをすぐにローテーションできます。Secrets Manager でマスターユーザーパスワードシークレットをローテーションするには、マルチ AZ DB クラスターを変更します。マルチ AZ DB クラスターの変更については、「Amazon RDS のマルチ AZ DB クラスターの変更」を参照してください。

RDS コンソール、AWS CLI、または RDS API を使用して、マスターユーザーのパスワードシークレットをすぐに更新できます。新しいパスワードは常に 28 文字で、少なくとも 1 つの大文字と小文字、1 つの数字、1 つの句読点が含まれます。

コンソール

RDS コンソールを使用してマスターユーザーパスワードシークレットをローテーションするには、マルチ AZ DB クラスターを変更し、[Settings] (設定) で [Rotate secret immediately] (シークレットを直ちにローテーションする) を選択します。



RDS コンソールを使用して、Amazon RDS のマルチ AZ DB クラスターの変更 および のマルチ AZ DB クラスターを変更する手順に従います。確認ページで [Apply immediately] (すぐに適用) を選択する必要があります。

AWS CLI

AWS CLI を使用してマスターユーザーパスワードシークレットをローテーションするには、modify-db-cluster コマンドを使用して `--rotate-master-user-password` オプションを指定します。マスターパスワードをローテーションするときは、`--apply-immediately` オプションを指定する必要があります。

この例では、マスターユーザーパスワードシークレットをローテーションします。

Example

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

RDS API

ModifyDBCluster オペレーションを使用して RotateMasterUserPassword パラメータを true に設定すると、マスターユーザーパスワードシークレットをローテーションできます。マスターパスワードを変更するときは、ApplyImmediately パラメータを true に設定する必要があります。

DB インスタンスのシークレットに関する詳細を表示する

コンソール (https://console.aws.amazon.com/secretsmanager/) または AWS CLI (get-secret-value Secrets Manager コマンド) を使用して自分のシークレットを取得できます。

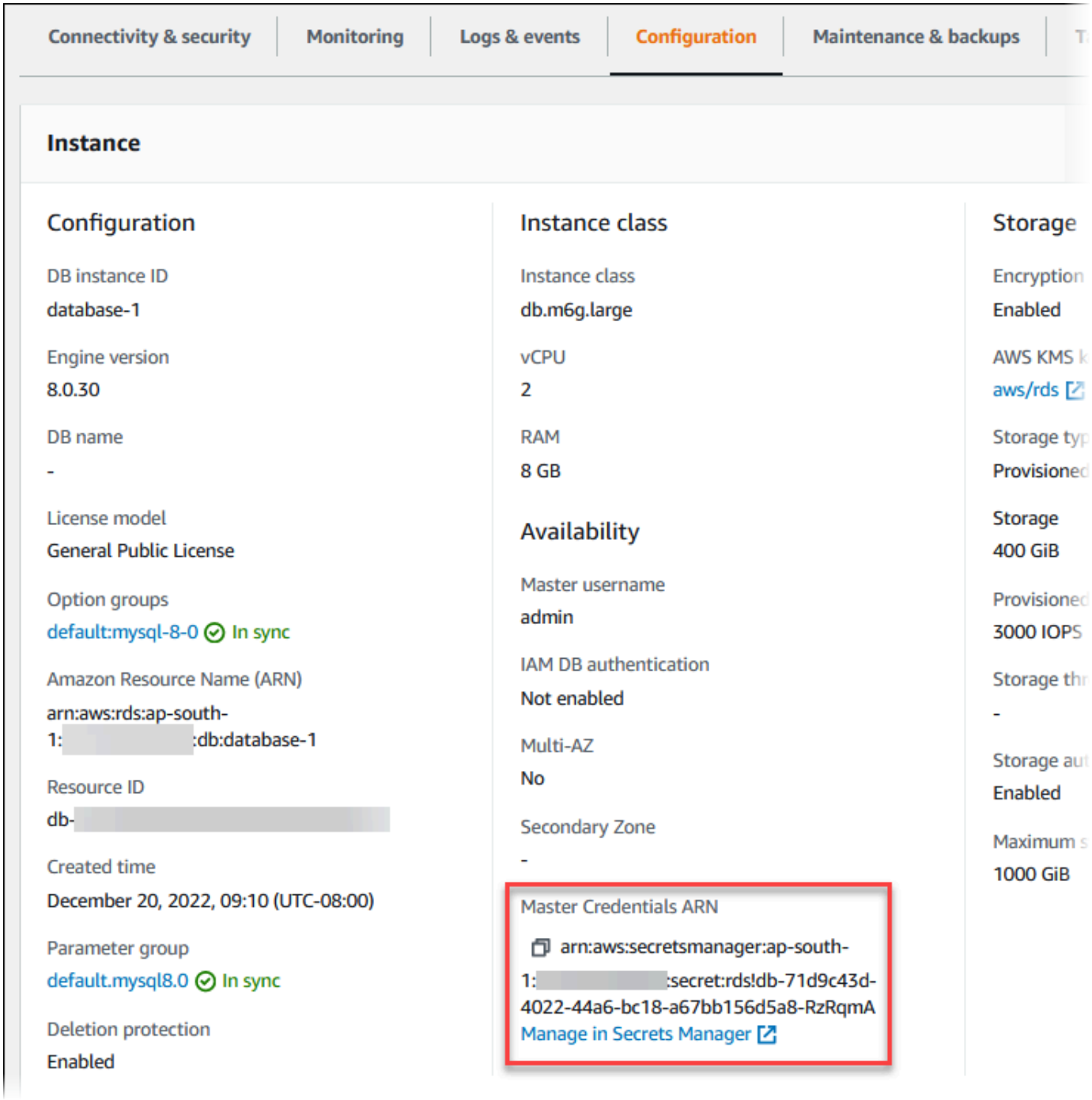
RDS によって管理されているシークレットの Amazon リソースネーム (ARN) は、RDS コンソール、AWS CLI、または RDS API の Secrets Manager で確認できます。

コンソール

RDS で管理されているシークレットの詳細を Secrets Manager で表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. DB インスタンスの名前を選択して、その詳細を表示します。
4. [設定] タブを選択します。

   [Master Credentials ARN] (マスター認証情報の ARN) では、シークレット ARN を表示できます。



[Manage in Secrets Manager] (Secrets Managerで管理) で管理リンクをクリックすると、Secrets Manager コンソールでシークレットを表示および管理できます。

AWS CLI

describe-db-instances RDS CLI コマンドを使用すると、Secrets Manager で RDS によって管理されているシークレットに関する次の情報を検索できます。

- SecretArn – シークレットの ARN
- SecretStatus – シークレットのステータス

  設定可能なステータス値は以下のとおりです。

  - creating – シークレットは作成中です。
  - active – シークレットは通常の使用とローテーションで利用可能です。
  - rotating – シークレットはローテーション中です。
  - impaired - シークレットはデータベースの認証情報へのアクセスに使用できますが、ローテーションはできません。例えば、アクセス許可が変更されて RDS がシークレットやシークレットの KMS キーにアクセスできなくなった場合、シークレットがこのステータスになる可能性があります。

    シークレットのステータスがこの場合は、ステータスの原因となった状態を修正できます。ステータスの原因となった条件を修正すると、ステータスは次のローテーションまで impaired のままです。または、DB インスタンスを変更してデータベース認証情報の自動管理をオフにしてから、DB インスタンスを再度変更してデータベース認証情報の自動管理をオンにすることもできます。DB インスタンスを変更するには、modify-db-instance コマンドの --manage-master-user-password オプションを使用します。
- KmsKeyId – シークレットの暗号化に使用する KMS キーの ARN。

特定の DB インスタンスの出力を表示する --db-instance-identifier オプションを指定します。この例は、DB インスタンスが使用するシークレットの出力を示しています。

Example

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

シークレットの出力例は次のとおりです。

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

シークレット ARN がある場合は、get-secret-value Secrets Manager CLI コマンドを使用してシークレットの詳細を表示できます。

この例は、前のサンプル出力のシークレットの詳細を示しています。

Example

Linux、macOS、Unix の場合:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Windows の場合:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
db-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

RDS API

RDS で管理されているシークレットの ARN、ステータス、および KMS キーを Secrets Manager で表示するには、DescribeDBInstances オペレーションを使用し、`DBInstanceIdentifier` パラメータを DB インスタンス識別子に設定します。シークレットの詳細は、出力に含まれています。

シークレット ARN がある場合は、GetSecretValue Secrets Manager オペレーションを使用してシークレットの詳細を表示できます。

マルチ AZ DB クラスターのシークレットに関する詳細の表示

コンソール (https://console.aws.amazon.com/secretsmanager/) または AWS CLI (get-secret-value Secrets Manager コマンド) を使用して自分のシークレットを取得できます。

RDS によって管理されているシークレットの Amazon リソースネーム (ARN) は、RDS コンソール、AWS CLI、または RDS API の Secrets Manager で確認できます。

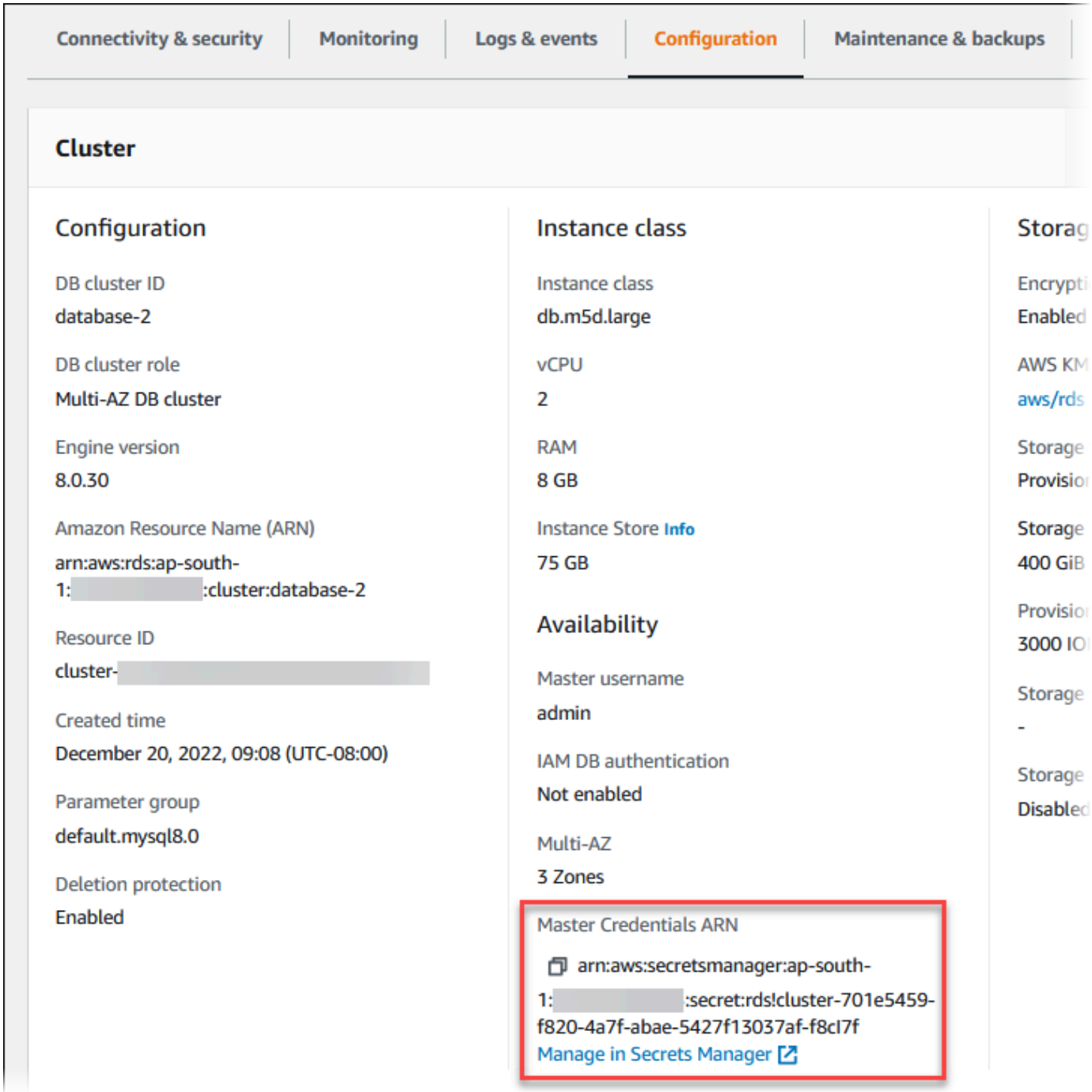
コンソール

RDS が管理するシークレットの詳細を Secrets Manager で表示する

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、データベース を選択します。
3. 詳細を表示する マルチ AZ DB クラスターの名前を選択します。
4. [設定] タブを選択します。

   [Master Credentials ARN] (マスター認証情報の ARN) では、シークレット ARN を表示できます。



[Manage in Secrets Manager] (Secrets Managerで管理) で管理リンクをクリックすると、Secrets Manager コンソールでシークレットを表示および管理できます。

AWS CLI

RDS AWS CLI describe-db-cluster コマンドを使用すると、Secrets Manager で RDS によって管理されているシークレットに関する次の情報を検索できます。

- `SecretArn` – シークレットの ARN
- `SecretStatus` – シークレットのステータス

  設定可能なステータス値は以下のとおりです。

  - `creating` – シークレットは作成中です。
  - `active` – シークレットは通常の使用とローテーションで利用可能です。
  - `rotating` – シークレットはローテーション中です。
  - `impaired` - シークレットはデータベースの認証情報へのアクセスに使用できますが、ローテーションはできません。例えば、アクセス許可が変更されて RDS がシークレットやシークレットの KMS キーにアクセスできなくなった場合、シークレットがこのステータスになる可能性があります。

    シークレットのステータスがこの場合は、ステータスの原因となった状態を修正できます。ステータスの原因となった条件を修正すると、ステータスは次のローテーションまで `impaired` のままです。または、DB クラスターを変更してデータベース認証情報の自動管理をオフにしてから、DB クラスターを再度変更してデータベース認証情報の自動管理をオンにすることもできます。DB クラスターを変更するには、modify-db-cluster コマンドの `--manage-master-user-password` を使用します。
- `KmsKeyId` – シークレットの暗号化に使用する KMS キーの ARN。

特定の DB クラスターの出力を表示する `--db-cluster-identifier` オプションを指定します。この例は、DB クラスターが使用するシークレットの出力を示しています。

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

以下は、シークレットの出力例を示しています。

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

シークレット ARN がある場合は、get-secret-value Secrets Manager CLI コマンドを使用してシークレットの詳細を表示できます。

この例は、前のサンプル出力のシークレットの詳細を示しています。

Example

Linux、macOS、Unix の場合:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Windows の場合:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

RDS API

RDS によって管理されているシークレットの ARN、ステータス、および KMS キーは、DescribeDBClusters RDS オペレーションを使用して Secrets Manager で表示し、`DBClusterIdentifier` パラメータを DB クラスター識別子に設定できます。シークレットの詳細は、出力に含まれています。

シークレット ARN がある場合は、GetSecretValue Secrets Manager オペレーションを使用してシークレットの詳細を表示できます。

テナントデータベースのシークレットに関する詳細の表示

コンソール (https://console.aws.amazon.com/secretsmanager/) または AWS CLI (get-secret-value Secrets Manager コマンド) を使用して自分のシークレットを取得できます。

AWS Secrets Manager の Amazon RDS によって管理されているシークレットの Amazon リソースネーム (ARN) は、Amazon RDS コンソール、AWS CLI、または Amazon RDS API にあります。

コンソール

テナントデータベースの AWS Secrets Manager で Amazon RDS によって管理されるシークレットの詳細を表示するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、データベース を選択します。
3. テナントデータベースを含む DB インスタンスの名前を選択して、その詳細を表示します。
4. [設定] タブを選択します。

   テナントデータベースセクションで、テナントデータベースを検索し、マスター認証情報の ARN を表示します。

   [Manage in Secrets Manager] (Secrets Managerで管理) で管理リンクをクリックすると、Secrets Manager コンソールでシークレットを表示および管理できます。

AWS CLI

describe-tenant-databases Amazon RDS AWS CLI コマンドを使用して、テナントデータベースの AWS Secrets Manager で Amazon RDS によって管理されるシークレットに関する次の情報を検索できます。

- `SecretArn` – シークレットの ARN
- `SecretStatus` – シークレットのステータス

  設定可能なステータス値は以下のとおりです。

  - `creating` – シークレットは作成中です。
  - `active` – シークレットは通常の使用とローテーションで利用可能です。
  - `rotating` – シークレットはローテーション中です。
  - `impaired` - シークレットはデータベースの認証情報へのアクセスに使用できますが、ローテーションはできません。例えば、アクセス許可が変更されて Amazon RDS がシークレットやシークレットの KMS キーにアクセスできなくなった場合、シークレットがこのステータスになる可能性があります。

シークレットのステータスがこの場合は、ステータスの原因となった状態を修正できます。ステータスの原因となった条件を修正すると、ステータスは次のローテーションまで `impaired` のままです。または、テナントデータベースを変更してデータベース認証情報の自動管理をオフにしてから、テナントデータベースを再度変更してデータベース認証情報の自動管理をオンにすることもできます。テナントデータベースを変更するには、modify-tenant-database コマンドで `--manage-master-user-password` オプションを使用します。

- `KmsKeyId` – シークレットの暗号化に使用する KMS キーの ARN。

特定の DB インスタンスにあるテナントデータベースの出力を表示するには、`--db-instance-identifier` オプションを指定します。特定のテナントデータベースの出力を表示する `--tenant-db-name` オプションを指定することもできます。この例は、テナントデータベースが使用するシークレットの出力を示しています。

Example

```
aws rds describe-tenant-databases \
    --db-instance-identifier database-3 \
    --query "TenantDatabases[0].MasterUserSecret"
```

シークレットの出力例は次のとおりです。

```
{
    "SecretArn": "arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123",
    "SecretStatus": "active",
    "KmsKeyId": "arn:aws:kms:us-east-2:123456789012:key/aa11bb22-####-####-####-
fedcba123456"
}
```

シークレット ARN がある場合は、get-secret-value Secrets Manager AWS CLI コマンドを使用してシークレットの詳細を表示できます。

この例は、前のサンプル出力のシークレットの詳細を示しています。

Example

Linux、macOS、Unix の場合:

```
aws secretsmanager get-secret-value \
```

```
    --secret-id 'arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123'
```

Windows の場合:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:us-east-2:123456789012:secret:rds!db-ABC123'
```

Amazon RDS API

AWS Secrets Manager で Amazon RDS によって管理されているシークレットの ARN、ステータス、および KMS キーを表示するには、DescribeTenantDatabases オペレーションを使用し、`DBInstanceIdentifier` パラメータを DB インスタンス識別子に設定します。`TenantDBName` パラメータを特定のテナントデータベース名に設定することもできます。シークレットの詳細は、出力に含まれています。

シークレット ARN がある場合は、GetSecretValue Secrets Manager オペレーションを使用してシークレットの詳細を表示できます。

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョンと AWS リージョン によって異なります。Secrets Manager を Amazon RDS と統合した場合のバージョンとリージョンの可用性の詳細については、「Secrets Manager と Amazon RDS の統合でサポートされているリージョンと DB エンジン」を参照してください。

Amazon RDS でのデータ保護

AWS 責任共有モデルは、Amazon Relational Database Service のデータ保護に適用されます。このモデルで説明されているように、「AWS」は、「AWS クラウド」のすべてを実行するグローバルインフラストラクチャを保護する責任があります。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーに関する詳細については、 Data Privacy FAQ をご覧ください。欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データを保護するため、「AWS アカウント」認証情報を保護し、AWS IAM アイデンティティセンター または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- AWS CloudTrail で API とユーザーアクティビティロギングを設定します。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の「CloudTrail 証跡の使用」を参照してください。
- AWS のサービス内のすべてのデフォルトセキュリティコントロールに加え、AWS 暗号化ソリューションを使用します。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して AWS にアクセスする際に FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これには、コンソール、API、AWS CLI、または AWS SDK を使用して Amazon RDS またはその他の AWS のサービスで作業する場合が含まれます。タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

トピック

- 暗号化を使用したデータの保護
- ネットワーク間のトラフィックのプライバシー

暗号化を使用したデータの保護

データベースリソースの暗号化を有効にすることができます。また、DB インスタンスへの接続を暗号化することもできます。

トピック

- Amazon RDS リソースの暗号化
- AWS KMS key 管理
- SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化

- SSL/TLS 証明書のローテーション

Amazon RDS リソースの暗号化

Amazon RDS は、Amazon RDS DB インスタンスを暗号化することができます。保管中に暗号化されるデータには、DB インスタンス、ログ、自動バックアップ、リードレプリカ、スナップショットの基本的なストレージが含まれます。

Amazon RDS の暗号化された DB インスタンスでは、Amazon RDS DB インスタンスをホストしているサーバーでデータを暗号化するために、業界標準の AES-256 暗号化アルゴリズムを使用します。

データが暗号化されると、Amazon RDS はパフォーマンスの影響を最小限に抑えながら、データへのアクセスと復号の認証を透過的に処理します。暗号化を使用するために、データベースのクライアントアプリケーションを変更する必要はありません。

> **Note**
>
> 暗号化された/されていない DB インスタンスのでは、AWS リージョン間でレプリケートする場合でも、ソースとリードレプリカ間で送信されるデータは暗号化されます。

トピック

- Amazon RDS リソースの暗号化の概要
- DB インスタンスの暗号化
- DB インスタンスの暗号化が有効になっているかの判別
- Amazon RDS の暗号化の可用性
- 転送中の暗号化
- Amazon RDS の暗号化された DB インスタンスの制限事項

Amazon RDS リソースの暗号化の概要

Amazon RDS の暗号化された DB インスタンスは、基になるストレージへの不正アクセスからデータを保護することによって、データ保護の追加レイヤーを提供します。Amazon RDS の暗号化を使用して、クラウドにデプロイされるアプリケーションのデータ保護を強化することや、保管時のデータ暗号化に関するコンプライアンスの要件を達成することができます。 Amazon RDS の暗号化された DB インスタンスでは、すべてのログ、バックアップ、スナップショットが暗号化されます。暗号

化の可用性と制限の詳細については、「Amazon RDS の暗号化の可用性」および「Amazon RDS の暗号化された DB インスタンスの制限事項」を参照してください。

Amazon RDS は、これらのリソースを暗号化するために AWS Key Management Service キーを使用します。AWS KMS は、安全で可用性の高いハードウェアとソフトウェアを組み合わせて、クラウド向けに拡張されたキー管理システムを提供します。AWS マネージドキー を使用することも、カスタマーマネージドキーを作成することもできます。

暗号化された DB インスタンスを作成するときは、カスタマーマネージドキーまたは Amazon RDS の AWS マネージドキー を選択して、DB インスタンスを暗号化できます。カスタマーマネージドキーのキー識別子を指定しない場合、Amazon RDS は新しい DB インスタンスに AWS マネージドキー を使用します。Amazon RDS は、Amazon RDS 用の AWS マネージドキー を AWS アカウントに作成します。AWS アカウントには、AWS リージョンごとに Amazon RDS の AWS マネージドキー が別々にあります。

Amazon RDS リソースの暗号化と復号に使用するカスタマーマネージドキーを管理するには、AWS Key Management Service (AWS KMS).を使用します。

AWS KMS を使用して、カスタマーマネージドキーを作成し、これらのカスタマーマネージドキーの使用を制御するポリシーを定義できます。AWS KMS は CloudTrail をサポートしているため、KMS キーの使用状況を監査して、カスタマーマネージドキーが適切に使用されていることを確認できます。カスタマーマネージドキーは、Amazon Aurora およびサポートされている AWS のサービス (Amazon S3、Amazon EBS、Amazon Redshift など) で使用できます。AWS KMS と統合しているサービスのリストについては、「AWS サービス統合」を参照ください。KMS キーの使用に関する考慮事項:

- 暗号化された DB インスタンスを作成したら、その DB インスタンスで使用されている KMS キーを変更することはできません。したがって、暗号化された DB インスタンスを作成する前に、KMS キーの要件を必ず確認してください。

  DB インスタンスの暗号化キーを変更する必要がある場合は、インスタンスの手動スナップショットを作成し、スナップショットのコピー中に暗号化を有効にします。詳細については、re:Post 情報センターの記事を参照してください。
- 暗号化されたスナップショットをコピーする場合、ソーススナップショットの暗号化に使用した KMS キーとは異なる KMS キーを使用して、ターゲットスナップショットを暗号化できます。
- また、Amazon RDS の暗号化されたインスタンスのリードレプリカとプライマリ DB インスタンスの両方が同じ AWS リージョンにある場合、リードレプリカはプライマリ DB インスタンスと同じ KMS キーを使用して暗号化する必要があります。

- プライマリ DB インスタンスとリードレプリカが異なる AWS リージョンにある場合には、その AWS リージョンの KMS キー を使用してリードレプリカを暗号化します。
- スナップショットを共有する AWS アカウントの AWS マネージドキー を使って暗号化されたスナップショットを共有することはできません。
- Amazon RDS は、Transparent Data Encryption (TDE) による Oracle または SQL Server の DB インスタンスの暗号化もサポートします。TDE は、RDS 保管時の暗号化とともに使用できますが、TDE と RDS 保管時の暗号化を同時に使用すると、データベースのパフォーマンスに若干影響する可能性があります。個々の暗号化方式ごとに異なるキーを管理する必要があります。TDE の詳細については「Oracle Transparent Data Encryption」または「SQL サーバーの透過的なデータの暗号化サポート」を参照してください。

Important

KMS キーを無効にすると、Amazon RDS は DB インスタンス用の KMS キーにアクセスできなくなります。KMS キーへのアクセスを失った場合、バックアップが有効になっているインスタンスでは、暗号化された DB インスタンスは検出から 2 時間後に `inaccessible-encryption-credentials-recoverable` 状態になります。DB インスタンスは 7 日間この状態のままであり、その間、インスタンスは停止します。この間に DB インスタンスに対して行われた API コールは成功しない場合があります。DB インスタンスを復旧するには、KMS キーを有効にして、この DB インスタンスを再起動します。AWS マネジメントコンソール、AWS CLI、または RDS API から KMS キーを有効にします。AWS CLI コマンド start-db-instance または AWS マネジメントコンソール を使用して DB インスタンスを再起動します。

`inaccessible-encryption-credentials-recoverable` 状態は、停止できる DB インスタンスにのみ適用されます。この回復可能な状態は、リードレプリカやリードレプリカを持つインスタンスなど、停止できないインスタンスには適用されません。詳細については、「DB インスタンスの停止に関する制限事項」を参照してください。

DB インスタンスが 7 日以内に復旧されない場合、終了 `inaccessible-encryption-credentials` 状態になります。この状態では、DB インスタンスは使用できなくなり、バックアップからのみ DB インスタンスを復元できます。データベース内の暗号化されたデータの消失を防ぐために、暗号化された DB インスタンスのバックアップは常に有効にしておくことを強くお勧めします。

DB インスタンスの作成中に、Amazon RDS は呼び出し元のプリンシパルが KMS キーにアクセスできるかどうかを確認し、DB インスタンスの存続期間全体に使用する KMS キーから許可を生成します。呼び出し元のプリンシパルの KMS キーへのアクセス権を取り消しても、実行中のデータベースには影響しません。スナップショットを別のアカウントにコピー

するなど、クロスアカウントシナリオで KMS キーを使用する場合は、KMS キーを他のアカウントと共有する必要があります。別の KMS キーを指定せずにスナップショットから DB インスタンスを作成すると、新しいインスタンスはソースアカウントの KMS キーを使用します。DB インスタンスの作成後にキーへのアクセス権を取り消しても、インスタンスには影響しません。ただし、キーを無効にすると、そのキーで暗号化されたすべての DB インスタンスに影響します。これを防ぐには、スナップショットのコピー操作時に別のキーを指定します。
バックアップが無効化されている DB インスタンスは、インスタンスの変更または復旧中にボリュームがホストからデタッチされるまで使用できます。RDS は、該当する場合、インスタンスを `inaccessible-encryption-credentials-recoverable` 状態または `inaccessible-encryption-credentials` 状態に移行します。

KMS キーの詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS keys」と「AWS KMS key 管理」を参照してください。

DB インスタンスの暗号化

新しい DB インスタンスを暗号化するには、Amazon RDS コンソールで [Enable encryption] (暗号を有効化) を選択します。DB インスタンスの作成については、「Amazon RDS DB インスタンスの作成」を参照してください。

AWS CLI の create-db-instance コマンドを使用して、暗号化された DB インスタンスを作成するには、`--storage-encrypted` パラメータを設定します。CreateDBInstance API オペレーションを使用する場合は、`StorageEncrypted` パラメータを true に設定します。

AWS CLI `create-db-instance` コマンドを使用して、カスタマーマネージドキーで暗号化された DB インスタンスを作成する場合は、`--kms-key-id` パラメータを KMS キーの任意のキー識別子に設定します。Amazon RDS API `CreateDBInstance` オペレーションを使用する場合は、`KmsKeyId` パラメータを KMS キーの任意のキー識別子に設定します。カスタマーマネージドキーを別の AWS アカウントで使用するには、キー ARN またはエイリアス ARN を指定します。

DB インスタンスの暗号化が有効になっているかの判別

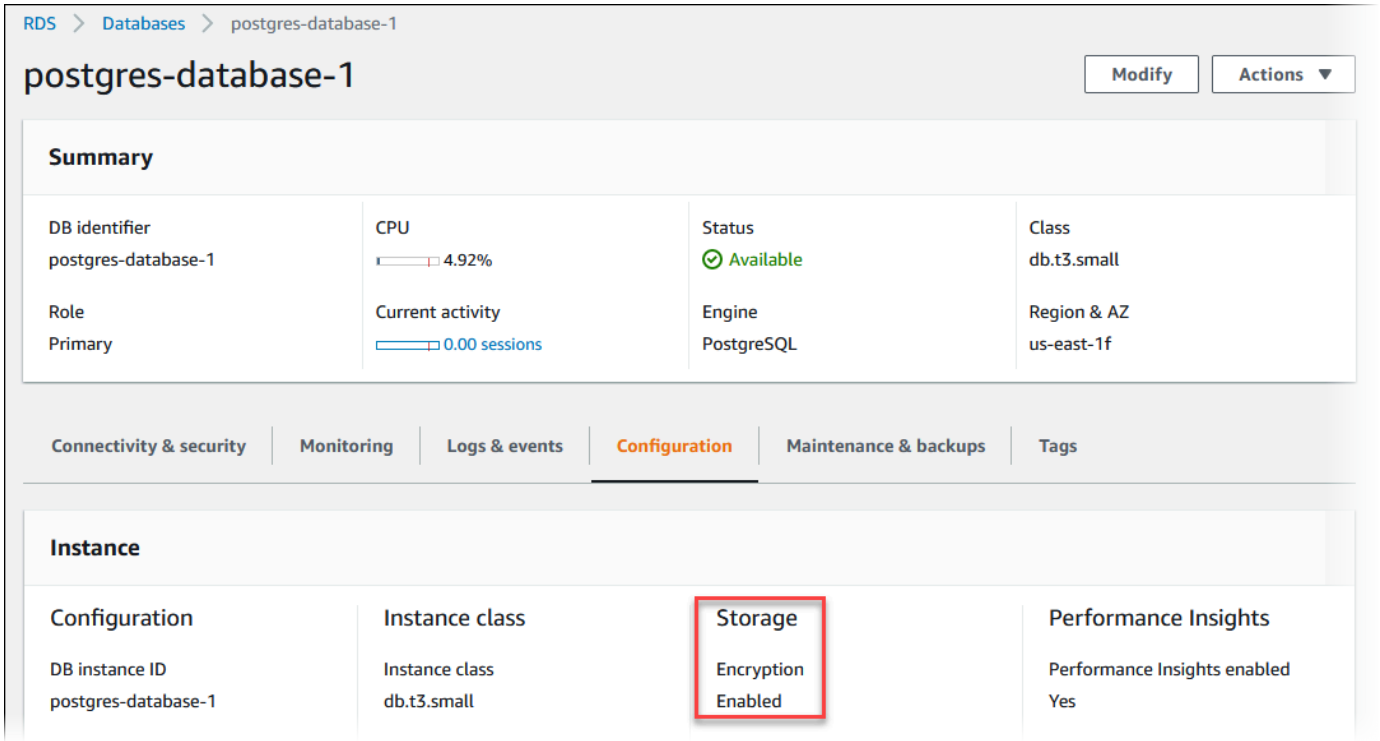
AWS マネジメントコンソール、AWS CLI、または RDS API を使用して、DB インスタンスの保存時の暗号化が有効になっているか判別できます。

コンソール

DB インスタンスに対して保存時の暗号化がオンになっているかどうかを判別します

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 詳細を表示するためにチェックする DB インスタンスの名前を選択します。
4. 設定タブを選択し、ストレージ下にある暗号化 値 をチェックします。

   有効または無効のいずれかが表示されています。



AWS CLI

AWS CLI を使用して DB インスタンスの保存時の暗号化が有効になっているかを判断するには、以下のオプションで describe-db-instances コマンドを呼び起こします:

- `--db-instance-identifier` - DB インスタンスの名前です。

次の例では、mydb DB インスタンスの保管時の暗号化に関して TRUE または FALSE の いずれかを返すクエリを使用しています。

Example

```
aws rds describe-db-instances --db-instance-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

RDS API

Amazon RDS API を使用して DB インスタンスの保管時の暗号化が有効であるかを判断するには、以下のパラメータで DescribeDBInstances オペレーションを呼び起こします:

- `DBInstanceIdentifier` - DB インスタンスの名前です。

Amazon RDS の暗号化の可用性

Amazon RDS 暗号化は、現在すべてのデータベースエンジンおよびストレージタイプに使用できます。

Amazon RDS 暗号化は、ほとんどの DB インスタンスクラスで使用できます。次の表は、Amazon RDS 暗号化をサポートしていない DB インスタンスクラスの一覧です。

インスタンスタイプ	インスタンスクラス
汎用 (M1)	db.m1.small db.m1.medium db.m1.large db.m1.xlarge
メモリ最適化 (M2)	db.m2.xlarge db.m2.2xlarge db.m2.4xlarge
バースト可能 (T2)	db.t2.micro

転送中の暗号化

物理レイヤーでの暗号化

AWS グローバルネットワーク上の AWS リージョンを流れるすべてのデータは、AWS の安全な施設を離れる前に、物理層で自動的に暗号化されます。AZ 間のトラフィックはすべて暗号化されます。追加的な暗号化レイヤーでは、このセクションに記載されているもの以外にも、保護が提供されている場合があります。

Amazon VPC ピアリングおよび Transit Gateway のクロスリージョンピアリング接続によって得られる暗号化

Amazon VPC およびTransit Gateway のピアリング接続を使用する、すべてのクロスリージョントラフィックは、リージョンからの送信時に自動的に一括で暗号化されます。すべてのトラフィックにおける物理レイヤーには、そのトラフィックが AWS の保護された設備を離れる前に、追加の暗号化レイヤーが自動的に提供されています。

インスタンス間での暗号化

AWS では、すべてのタイプの DB インスタンス間において安全でプライベートな接続を提供しています。さらに、一部のインスタンスタイプでは、基盤となる Nitro System ハードウェアのオフロード機能を使用して、インスタンス間の転送中のトラフィックを自動的に暗号化します。この暗号化では、256 ビットの暗号化による関連データによる認証暗号化 (AEAD) アルゴリズムを使用します。ネットワークのパフォーマンスには影響しません。インスタンス間でこの追加の転送中トラフィック暗号化をサポートするには、次の要件を満たす必要があります。

- インスタンスは、次のインスタンスタイプを使用します。
  - 汎用: M6i、M6id、M6in、M6idn、M7g
  - メモリ最適化: R6i、R6id、R6in、R6idn、R7g、X2idn、X2iedn、X2iezn
- 各インスタンスは同じ AWS リージョンにあるものとします。
- 各インスタンスは同じ VPC 内、あるいはピア接続された VPC 内にあり、トラフィックは仮想ネットワークのデバイスもしくはサービス (ロードバランサーや Transit Gateway など) を通過しないものとします。

Amazon RDS の暗号化された DB インスタンスの制限事項

Amazon RDS の暗号化された DB インスタンスには、以下の制限事項があります。

- Amazon RDS DB インスタンスは、DB インスタンスの作成時にのみ暗号化できます。作成後には暗号化できません。

ただし、暗号化されていないスナップショットのコピーは暗号化できるので、暗号化されていない DB インスタンスに効果的に暗号化を追加できます。つまり、DB インスタンスのスナップショットを作成し、そのスナップショットの暗号化済みコピーを作成します。この暗号化されたスナップショットから DB インスタンスを復元することで、元の DB インスタンスの暗号化されたコピーを作成できます。詳しくは、「Amazon RDS の DB スナップショットのコピー」を参照してください。

- 暗号化された DB インスタンスの暗号化をオフにすることはできません。
- 暗号化されていない DB インスタンスの暗号化されたスナップショットを作成することはできません。
- 暗号化された DB インスタンスのスナップショットは、DB インスタンスと同じ KMS キーを使用して暗号化する必要があります。
- 暗号化されていない DB インスタンスのリードレプリカを暗号化することや、暗号化されている DB インスタンスのリードレプリカを暗号化しないようにすることはできません。
- 暗号化されたリードレプリカとソース DB インスタンスの両方が同じ AWS リージョンにある場合、リードレプリカはソース DB インスタンスと同じ KMS キーで暗号化する必要があります。
- 暗号化されていないバックアップやスナップショットを、暗号化された DB インスタンスに復元することはできません。
- ある AWS リージョンから別のリージョンに暗号化されたスナップショットをコピーするには、送信先 AWS リージョンの KMS キーを指定する必要があります。これは、KMS キーが、作成される AWS リージョンに固有のものであるためです。

  ソーススナップショットはコピープロセス全体で暗号化されたままになります。Amazon RDSは、コピー処理中にエンベロープ暗号化を使用してデータを保護します。エンベロープ暗号化の仕組みの詳細については、AWS Key Management Service デベロッパーガイドの「エンベロープ暗号化」を参照してください。
- 暗号化された DB インスタンスの暗号化を解除することはできません。ただし、暗号化された DB インスタンスからデータをエクスポートし、暗号化されていない DB インスタンスにデータをインポートすることはできます。

AWS KMS key 管理

Amazon RDS には、キー管理のために AWS Key Management Service (AWS KMS) が自動的に統合されます。Amazon RDS では、エンベロープ暗号化が使用されます。エンベロープ暗号化の仕組みの詳細については、AWS Key Management Service デベロッパーガイドの「エンベロープ暗号化」を参照してください。

2 種類の AWS KMS キーを使用して、DB インスタンスを暗号化できます。

- KMS キーに対するフル制御の権限が必要な場合は、カスタマーマネージドキーを作成する必要があります。カスタマーマネージドキーの詳細については、AWS Key Management Service デベロッパーガイドの「Customer managed keys」を参照してください。
- AWS マネージドキー は、お客様のアカウントにある KMS キーであり、AWS KMS と統合されている AWS のサービスがお客様に代わって作成し、管理し、使用します。デフォルトでは、RDS AWS マネージドキー (`aws/rds`) が暗号化に使用されます。RDS AWS マネージドキー の管理、ローテーション、削除はできません。AWS マネージドキー の詳細については、「AWS Key Management Service Developer Guide」の「AWS マネージドキー 」を参照してください。

Amazon RDS の暗号化された DB インスタンスに使用される KMS キーを管理するには、AWS KMS コンソールの AWS Key Management Service (AWS KMS)、AWS CLI、または AWS KMS API を使用します。AWS マネージドキーまたはカスタマーマネージドキーで実行された、すべてのアクションの監査ログを表示するには、AWS CloudTrail を使用します。キーのローテーションの詳細については、「AWS KMS キーのローテーション」を参照してください。

カスタマーマネージドキーの使用の承認

RDS が暗号化オペレーションでカスタマーマネージドキーを使用すると、RDS リソースを作成または変更するユーザーに代わって動作します。

カスタマーマネージドキーを使用して、RDS リソースを作成するには、カスタマーマネージドキーで次の操作を呼び出すためのアクセス許可がユーザーに必要になります。

- `kms:CreateGrant`
- `kms:DescribeKey`

これらの必要なアクセス許可は、キーポリシーか、キーポリシーで許可されている場合は IAM ポリシーで指定できます。

> **Important**
>
> Amazon RDS などのマネージドサービスを使用する AWS KMS キーポリシーですべてのリソース (*) に明示的な拒否ステートメントを使用する場合は、リソース所有アカウントを許可する条件を指定する必要があります。拒否ルールに IAM ユーザーの例外が含まれている場合でも、この条件なしで操作が失敗することがあります。

> **Tip**
>
> 最小権限のプリンシパルに従うには、`kms:CreateGrant` へのフルアクセスを許可しないでください。代わりに、ユーザーに代わって AWS サービスによって許可が作成された場合にのみ、kms:ViaService 条件キーを使用して、ユーザーが KMS キーに許可を作成できるようにします。

さまざまな方法で、IAM ポリシー をより厳しくすることができます。例えば、RDS から発信されるリクエストにのみカスタマーマネージドキーの使用を許可する場合、`rds.<region>.amazonaws.com` 値を指定して kms:ViaService 条件キーを使用します。また、暗号化にカスタマーマネージドキーを使用する条件として、Amazon RDS 暗号化コンテキストでキーまたは値を使用することもできます。

詳細については、「AWS Key Management Service デベロッパーガイド」の「他のアカウントのユーザーに KMS キーの使用を許可する」と「AWS KMS のキーポリシー」を参照してください。

Amazon RDS 暗号化コンテキスト

RDS が KMS キーを使用する場合、または Amazon EBS が RDS の代わりに KMS キーを使用する場合、サービスは暗号化コンテキストを指定します。暗号化コンテキストは、データの整合性を保証するために AWS KMS で使用される追加の認証データ (AAD) です。暗号化オペレーションで暗号化コンテキストを指定すると、サービスは復号オペレーションでも同じ暗号化コンテキストを指定する必要があります。そうしないと、復号は失敗します。暗号化コンテキストは AWS CloudTrail ログにも書き込まれるため、特定の KMS キーが使用された理由を理解するのに役立ちます。CloudTrail ログには KMS キーの使用を説明する多くのエントリが含まれている場合があります。各ログエントリの暗号化コンテキストは、その特定の使用理由を判断するのに役立ちます。

少なくとも Amazon RDS は、次の JSON 形式の例のように、暗号化コンテキストに DB インスタンス ID を常に使用します。

```
{ "aws:rds:db-id": "db-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

この暗号化コンテキストにより、KMS キーが使用された DB インスタンスを識別することができます。

KMS キーが特定の DB インスタンスと特定の Amazon EBS ボリュームに使用されると、次の JSON 形式の例のように、DB インスタンス ID と Amazon EBS ボリューム ID の両方が暗号化コンテキストに使用されます。

```
{
  "aws:rds:db-id": "db-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化

アプリケーションで Secure Socket Layer (SSL) または Transport Layer Security (TLS) を使用することで、Db2、MariaDB、Microsoft SQL Server、MySQL、Oracle、または PostgreSQL を実行するデータベースへの接続を暗号化できます。

SSL/TLS 接続は、クライアントと DB インスタンスまたはクラスターの間を移動するデータを暗号化することによって、1 つのセキュリティ層を提供します。オプションで、データベースにインストールされたサーバー証明書を検証することで、SSL/TLS 接続でサーバー ID 検証を実行できます。サーバーの ID 検証を必須にするには、次の一般的な手順に従ってください。

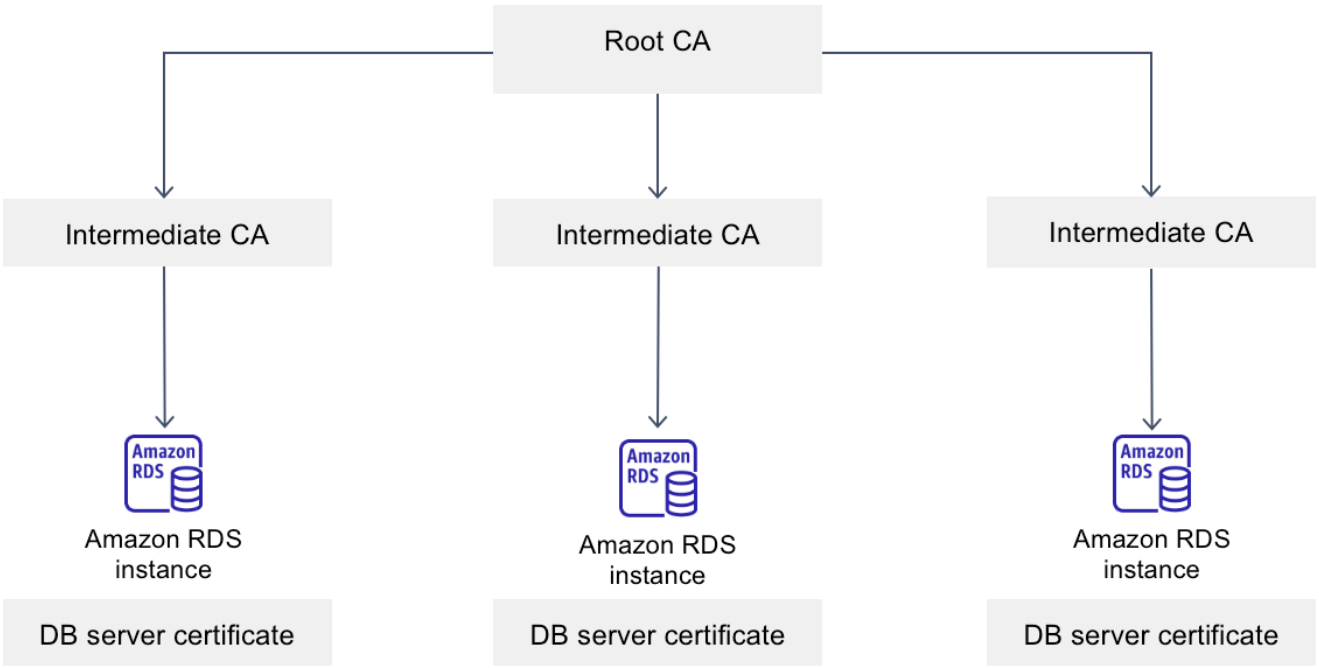
1. データベースの DB サーバー証明書に署名する認証局 (CA) を選択します。認証局の詳細については、「認証局」を参照してください。
2. データベースに接続するときに使用する証明書バンドルをダウンロードします。証明書バンドルをダウンロードするには、「AWS リージョン による証明書バンドル」を参照してください。

   > **Note**
   >
   > 証明書はすべて、SSL/TLS 接続を使用したダウンロードでのみ使用可能です。

3. DB エンジンのプロセスで、SSL/TLS Connect を実装するプロセスで、データベースに接続します。各 DB エンジンには SSL/TLS を実装する独自のプロセスがあります。ご使用のデータベースの SSL/TLS の実装方法については、ご使用の DB エンジンに対応する以下のリンクを使用してください。
   - Amazon RDS for Db2 DB インスタンスでの SSL/TLS の使用
   - Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート
   - Microsoft SQL Server DB インスタンスでの SSL の使用
   - Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート
   - RDS for Oracle DB インスタンスでの SSL の使用
   - PostgreSQL DB インスタンスで SSL を使用する

認証局

認証局 (CA) は、証明書チェーンの最上位にあるルート CA を識別する証明書です。CA は、各 DB インスタンスにインストールされているサーバー証明書である、DB インスタンス証明書に署名します。DB サーバー証明書は DB インスタンスを信頼できるサーバーとして識別します。



Amazon RDS にはデータベース用の DB サーバー証明書に署名するための、以下の CA が用意されています。

認証局 (CA)	説明	共通名 (CN)
rds-ca-rsa2048-g1	大半の AWS リージョン では、RSA 2048 プライベートキーアルゴリズムと SHA256 署名アルゴリズムを備えた認証局を使用します。 AWS GovCloud (US) Regions では、CA が RSA 2048 プライベートキーアルゴリズムと SHA384 署名アルゴリズムを備えた認証局を使用します。	Amazon RDS *region-id entifier* ルート CA RSA2048 G1

認証局 (CA)	説明	共通名 (CN)
	この CA はサーバー証明書の自動ローテーションをサポートします。	
rds-ca-rsa4096-g1	RSA 4096 プライベートキーアルゴリズムと SHA384 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。	Amazon RDS *region-id entifier* ルート CA RSA4096 G1
rds-ca-ecc384-g1	ECC 384 プライベートキーアルゴリズムと SHA384 署名アルゴリズムを備えた認証局を使用します。この CA はサーバー証明書の自動ローテーションをサポートします。	Amazon RDS *region-id entifier* ルート CA ECC384 G1

> **Note**
>
> AWS CLI を使用している場合は、describe-certificates を使用して、上記の認証局の有効性を確認できます。

これらの CA 証明書は、地域およびグローバル証明書バンドルに含まれています。rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、または rds-ca-ecc384-g1 CA をデータベースで使用すると、RDS はデータベース上で DB サーバー証明書を管理します。RDS は、DB サーバーの証明書の有効期限が切れる前に証明書のローテーションを行います。

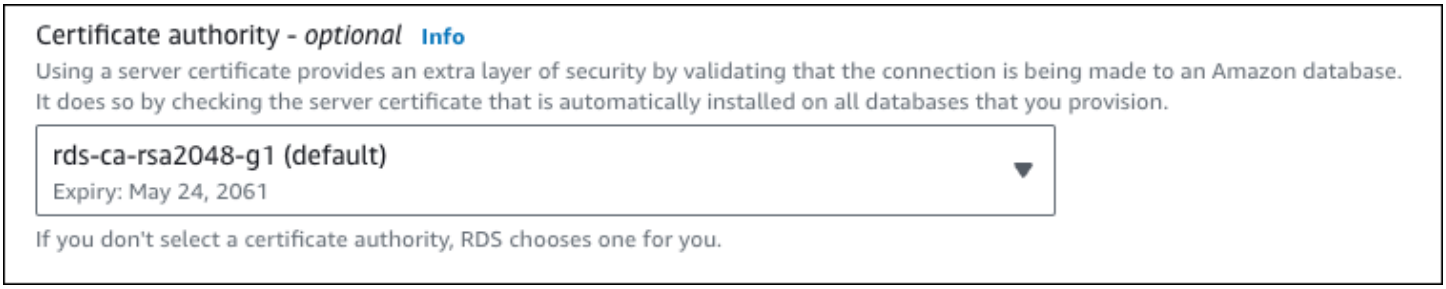
データベースに CA を設定する

データベースの CA は、以下のタスクの実行時に設定できます。

- DB インスタンスまたはマルチ AZ DB クラスターの作成 - DB インスタンスまたはクラスターを作成する際に CA を設定できます。手順については、「the section called “DB インスタンスの作成”」または「the section called “マルチ AZ DB クラスターの作成”」を参照してください。
- DB インスタンスまたはマルチ AZ DB クラスターの変更 – DB インスタンスまたはクラスターを変更することで CA を設定できます。手順については、「the section called “DB インスタンスを変更する”」または「the section called “マルチ AZ DB クラスターの変更”」を参照してください。

> **Note**
>
> デフォルトの CA は rds-ca-rsa2048-g1 に設定されています。modify-certificates コマンドを使用して、AWS アカウント のデフォルト CA をオーバーライドできます。

使用可能な CA は、DB エンジンと DB エンジンのバージョンによって異なります。AWS マネジメントコンソール を使用するとき、次の図に示すように、認証局の設定を使用して CA を選択できます。



コンソールには、DB エンジンおよび DB エンジンのバージョンで利用可能な CA のみが表示されます。AWS CLI を使用しているとき、create-db-instance コマンドまたは modify-db-instance コマンドを使用して DB インスタンスの CA を設定できます。マルチ AZ DB クラスターの CA は、create-db-cluster コマンドまたは modify-db-cluster コマンドを使用して指定できます。

AWS CLI を使用しているとき、describe-certificates コマンドを使用して、アカウントで使用可能な CA を確認できます。このコマンドでは、出力内の `ValidTill` に各 CA の有効期限も表示されます。describe-db-engine-versions コマンドを使用すると、特定の DB エンジンと DB エンジンのバージョンで使用できる CA を検索できます。

次の例は、PostgreSQL DB エンジンのバージョンのデフォルト RDS で使用できる CA を示しています。

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

以下のような出力が生成されます。使用可能な CA は `SupportedCACertificateIdentifiers` に記載されます。出力には、DB エンジンのバージョンで、`SupportsCertificateRotationWithoutRestart` の再起動なしで証明書のローテーションがサポートされるかどうかも表示されます。

```
{
```

```
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

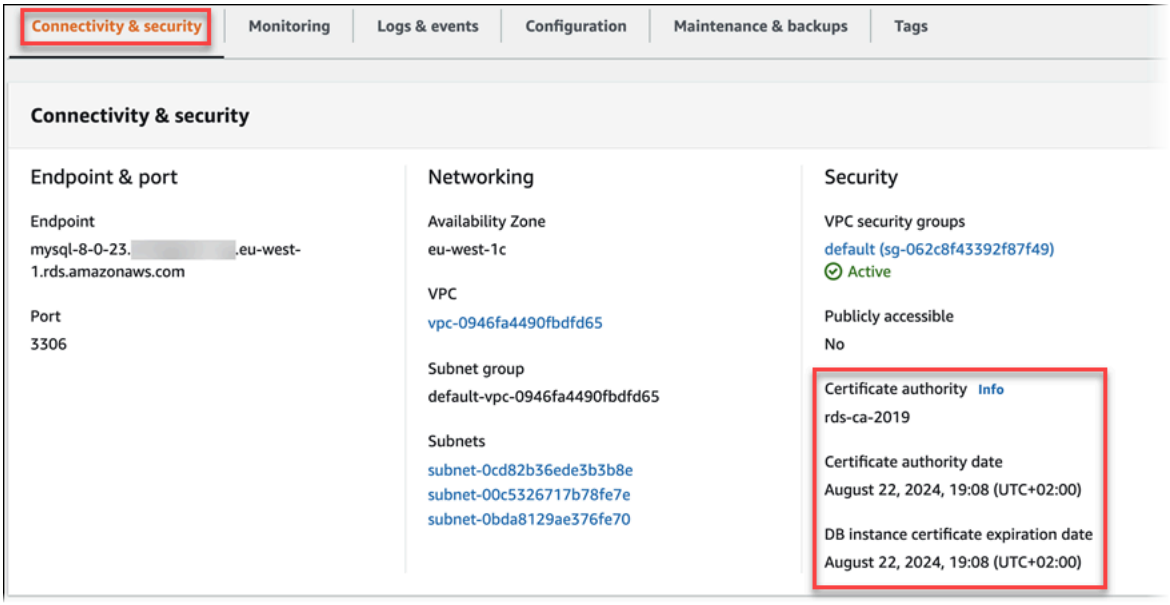

DB サーバーの証明書の有効性

DB サーバー証明書の有効性は、DB エンジンのバージョンと、DB エンジンのバージョンによって異なります。DB エンジンのバージョンで、DB エンジンのバージョンで、DB サーバーのローテーションがサポートされる場合、DB サーバーの証明書の有効期間は 1 年です。それ以外の場合、有効期間は 3 年間です。

DB サーバー証明書ローテーションの詳細については、「サーバー証明書の自動ローテーション」を参照してください。

DB インスタンスの CA の表示

データベースの CA に関する詳細を確認するには、次の画像のように、コンソール内の [接続とセキュリティ] タブを表示します。



AWS CLI を使用している場合は、describe-db-instances コマンドを使用して DB インスタンスの CA の詳細を表示できます。マルチ AZ DB クラスターの CA の詳細は、describe-db-clusters コマンドを使用して確認できます。

Amazon RDS の証明書バンドルをダウンロードする

SSL または TLS を使用してデータベースに接続する場合、データベースインスタンスには Amazon RDS からの信頼証明書が必要です。次の表の適切なリンクを選択して、データベースをホストする AWS リージョン に対応するバンドルをダウンロードします。

AWS リージョン による証明書バンドル

すべての AWS リージョンおよび GovCloud (米国) リージョンの証明書バンドルには、次のルート CA 証明書が含まれています。

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

`rds-ca-rsa4096-g1` および `rds-ca-ecc384-g1` 証明書は、以下のリージョンでは使用できません。

- アジアパシフィック (ムンバイ)
- アジアパシフィック (メルボルン)

- カナダ西部 (カルガリー)
- 欧州 (チューリッヒ)
- 欧州 (スペイン)
- イスラエル (テルアビブ)

アプリケーション信頼ストアでは、ルート CA 証明書の登録のみが必要です。RDS が DB サーバー証明書を自動的にローテーションするときに接続の問題が発生する可能性があるため、中間 CA 証明書を信頼ストアに登録しないでください。

> **Note**
>
> Amazon RDS Proxy は、AWS Certificate Manager (ACM) の証明書を使用します。RDS Proxy を使用している場合は、Amazon RDS 証明書をダウンロードしたり、RDS Proxy 接続を使用するアプリケーションを更新したりする必要はありません。詳細については、「RDS Proxy での TLS/SSL の使用」を参照してください。

AWS リージョン の証明書バンドルをダウンロードするには、次の表で、データベースをホストする AWS リージョン のリンクを選択します。

AWS リージョン	証明書バンドル (PEM)	証明書バンドル (PKCS7)
商用 AWS リージョン	global-bundle.pem	global-bundle.p7b
米国東部 (バージニア北部)	us-east-1-bundle.pem	us-east-1-bundle.p7b
米国東部 (オハイオ)	us-east-2-bundle.pem	us-east-2-bundle.p7b
米国西部 (北カリフォルニア)	us-west-1-bundle.pem	us-west-1-bundle.p7b
米国西部 (オレゴン)	us-west-2-bundle.pem	us-west-2-bundle.p7b
アフリカ (ケープタウン)	af-south-1-bundle.pem	af-south-1-bundle.p7b
アジアパシフィック (香港)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
アジアパシフィック (ハイデラバード)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b

AWS リージョン	証明書バンドル (PEM)	証明書バンドル (PKCS7)
アジアパシフィック (ジャカルタ)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
アジアパシフィック (マレーシア)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
アジアパシフィック (メルボルン)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
アジアパシフィック (ムンバイ)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
アジアパシフィック (大阪)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b
アジアパシフィック (タイ)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
アジアパシフィック (東京)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
アジアパシフィック (ソウル)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
アジアパシフィック (シンガポール)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
アジアパシフィック (シドニー)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b
カナダ (中部)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
カナダ西部 (カルガリー)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
欧州 (フランクフルト)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
欧州 (アイルランド)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
欧州 (ロンドン)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
ヨーロッパ (ミラノ)	eu-south-1-bundle.pem	eu-south-1-bundle.p7b
欧州 (パリ)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b

AWS リージョン	証明書バンドル (PEM)	証明書バンドル (PKCS7)
欧州 (スペイン)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
欧州 (ストックホルム)	eu-north-1-bundle.pem	eu-north-1-bundle.p7b
欧州 (チューリッヒ)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
イスラエル (テルアビブ)	il-central-1-bundle.pem	il-central-1-bundle.p7b
メキシコ (中部)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
中東 (バーレーン)	me-south-1-bundle.pem	me-south-1-bundle.p7b
中東 (アラブ首長国連邦)	me-central-1-bundle.pem	me-central-1-bundle.p7b
南米 (サンパウロ)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
任意の AWS GovCloud (US) Region	global-bundle.pem	global-bundle.p7b
AWS GovCloud (US-East)	us-gov-east-1-bundle.pem	us-gov-east-1-bundle.p7b
AWS GovCloud (US-West)	us-gov-west-1-bundle.pem	us-gov-west-1-bundle.p7b

CA 証明書の内容の表示

CA 証明書バンドルの内容を確認するには、次のコマンドを使用します。

```
keytool -printcert -v -file global-bundle.pem
```

SSL/TLS 証明書のローテーション

Amazon RDS 認証局証明書 rds-ca-2019 は、2024 年 8 月に期限切れになりました。RDS DB インスタンスまたはマルチ AZ DB クラスターへの接続に証明書検証付きの Secure Sockets Layer (SSL) または Transport Layer Security (TLS) を使用しているか、使用する予定がある場合は、新しい CA 証明書 rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、または rds-ca-ecc384-g1 のいずれかの使用を検討してください。現在、証明書検証付きで SSL/TLS を使用していない場合でも、CA 証明書の有効期限が切れている可能性があり、証明書検証付きで SSL/TLS を使用して RDS データベースに接続する予定がある場合は、新しい CA 証明書に更新する必要があります。

Amazon RDS では、AWS セキュリティのベストプラクティスとして、新しい CA 証明書を提供しています。新しい証明書およびサポートしている AWS リージョンの詳細については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

データベースの CA 証明書を更新するには、次の方法を使用します。

- DB インスタンスまたはクラスターの変更による CA 証明書の更新
- メンテナンスの適用による CA 証明書の更新

新しい CA 証明書を使用するように DB インスタンスまたはマルチ AZ DB クラスターを更新する前に、RDS データベースに接続するクライアントまたはアプリケーションを必ず更新します。

証明書を更新する際の考慮事項

証明書を更新する前に、次の状況を考慮してください。

- Amazon RDS Proxy は、AWS Certificate Manager (ACM) の証明書を使用します。RDS Proxy を使用している場合は、SSL/TLS 証明書を更新するときに、RDS Proxy 接続を使用するアプリケーションを更新する必要はありません。詳細については、「RDS Proxy での TLS/SSL の使用」を参照してください。
- 2020 年 7 月 28 日より前に作成された、または rds-ca-2019 証明書にアップデートされた DB インスタンスまたはマルチ AZ DB クラスターで Go バージョン 1.15 アプリケーションを使用している場合は、証明書を再度更新する必要があります。エンジンに応じて、証明書を rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、または rds-ca-ecc384-g1 に更新してください。

  新しい CA 証明書識別子を使用して、DB インスタンスの場合は `modify-db-instance` コマンド、またはマルチ AZ DB クラスターの場合は `modify-db-cluster` コマンドを使用します。`describe-db-engine-versions` コマンドを使用すると、特定の DB エンジンと DB エンジンのバージョンで使用できる CA を検索できます。

  2020 年 7 月 28 日以降にデータベースを作成したか、その証明書を更新した場合、必要なアクションはありません。詳細については、Go GitHub issue #39568 を参照してください。

DB インスタンスまたはクラスターの変更による CA 証明書の更新

次の例では、CA 証明書を rds-ca-2019 から rds-ca-rsa2048-g1 に更新します。別の証明書を選択できます。詳細については、「認証局」を参照してください。

アプリケーションの信頼ストアを更新して、CA 証明書の更新に関連するダウンタイムを短縮します。CA 証明書のローテーションに関連する再起動の詳細については、「サーバー証明書の自動ローテーション」を参照してください。

DB インスタンスまたはクラスターを変更して CA 証明書を更新するには

1. 「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」の説明に従って、新しい SSL/TLS 証明書をダウンロードします。
2. 新しい SSL/TLS 証明書を使用するようにアプリケーションを更新します。

   新しい SSL/TLS 証明書のアプリケーションを更新する方法は、特定のアプリケーションにより異なります。アプリケーション開発者と協力して、アプリケーションの SSL/TLS 証明書を更新します。

   SSL/TLS 接続の確認および各 DB エンジン用アプリケーションの更新については、以下のトピックを参照してください。

   - 新しい SSL/TLS 証明書を使用して MariaDB インスタンスに接続するようにアプリケーションを更新する
   - 新しい SSL/TLS 証明書を使用して Microsoft SQL Server DB インスタンスに接続するようにアプリケーションを更新する
   - 新しい SSL/TLS 証明書を使用して MySQL DB インスタンスに接続するようにアプリケーションを更新する
   - 新しい SSL/TLS 証明書を使用して Oracle DB インスタンスに接続するようにアプリケーションを更新する
   - 新しい SSL/TLS 証明書を使用して PostgreSQL DB インスタンスに接続するようにアプリケーションを更新する

   Linux オペレーティングシステムの信頼ストアを更新するサンプルスクリプトについては、「証明書を信頼ストアにインポートするためのサンプルスクリプト」を参照してください。

   > **Note**
   >
   > 証明書バンドルには古い CA と新しい CA の両方の証明書が含まれます。そのため、アプリケーションを安全に更新し、移行期間に接続を維持することができます。AWS Database Migration Service を使用してデータベースを DB インスタンスまたはクラス

ターに移行する場合は、移行中の接続を確保するために証明書バンドルを使用することをお勧めします。

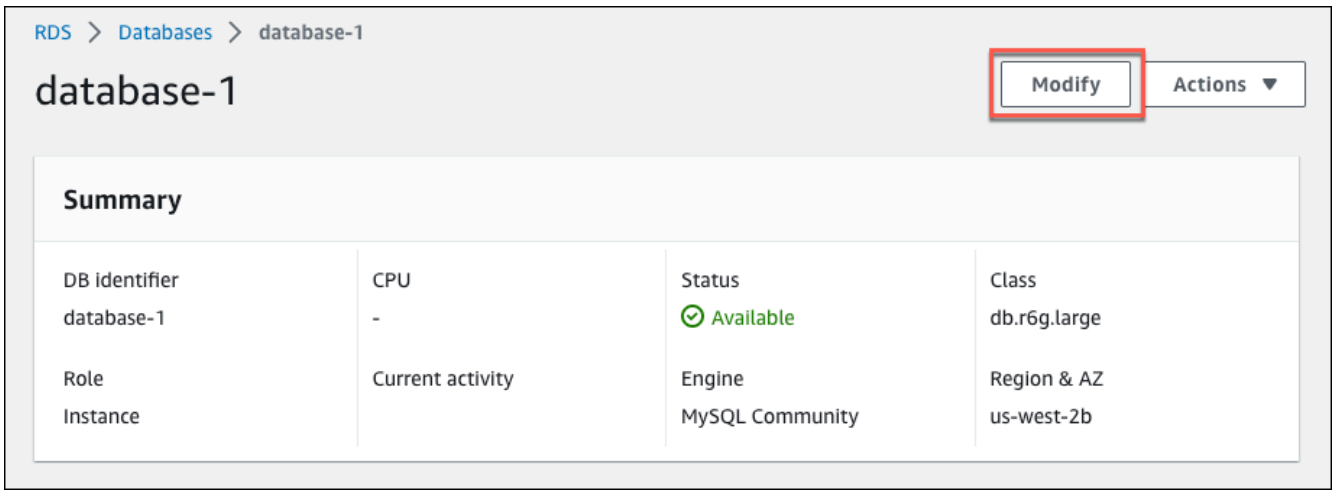
3. DB インスタンスまたはマルチ AZ DB クラスターを変更して、CA を rds-ca-2019 から rds-ca-rsa2048-g1 に変更します。CA 証明書を更新するためにデータベースの再起動が必要かどうかを確認するには、describe-db-engine-versions コマンドを使用して、`SupportsCertificateRotationWithoutRestart` フラグをチェックします。

> **Important**
>
> 証明書の有効期限が切れた後に接続の問題が発生した場合は、コンソールで [すぐに適用] を指定するか、`--apply-immediately` を使用して AWS CLI オプションを指定します。デフォルトで、このオペレーションは次のメンテナンスウィンドウの間に実行するようスケジュールされています。
> RDS for Oracle DB インスタンスでは、接続エラーを防ぐために Oracle DB を再起動することをお勧めします。
> 常時オプションやミラーリングオプションが有効になっている RDS for SQL Server のマルチ AZ インスタンスの場合、証明書ローテーション後にインスタンスを再起動すると、フェイルオーバーが予想されます。
> デフォルト RDS CA と異なるインスタンス CA のオーバーライドを設定するには、modify-certificates CLI コマンドを使用します。

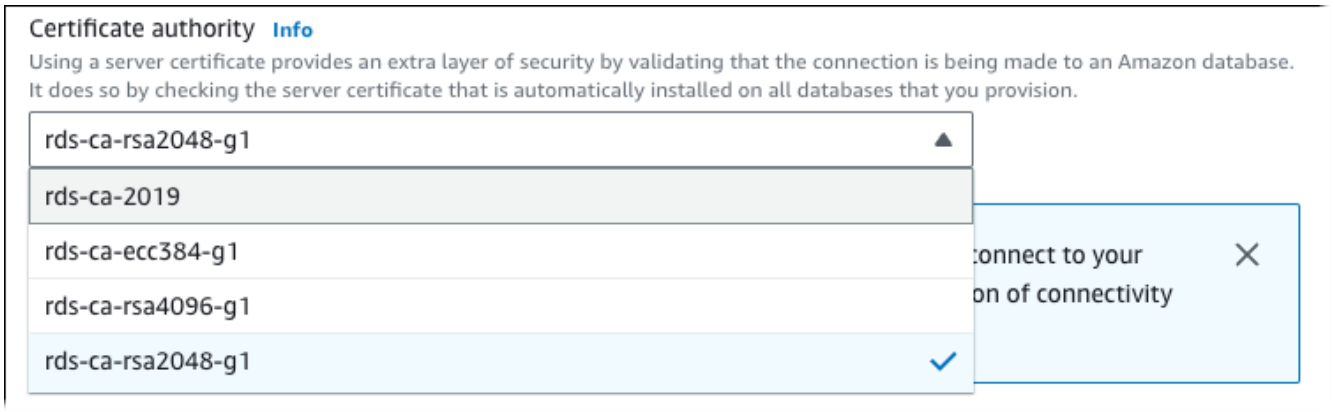
AWS マネジメントコンソール または AWS CLI を使用して、DB インスタンスまたはマルチ AZ DB クラスターの CA 証明書を rds-ca-2019 から rds-ca-rsa2048-g1 に変更できます。

Console

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択し、変更する DB インスタンスまたはマルチ AZ DB クラスターを選択します。
3. [Modify] (変更) を選択します。



4. [接続] セクションで、[rds-ca-rsa2048-g1] を選択します。



5. [続行] を選択して、変更の概要を確認します。
6. 変更をすぐに反映させるには、[Apply immediately] を選択します。
7. 確認ページで、変更内容を確認します。変更内容が正しい場合は、[DB インスタンスを変更] または [クラスターを変更] を選択して変更を保存します。

> **Important**
>
> このオペレーションをスケジュールする場合は、必ずクライアント側の信頼ストアを事前に更新します。

または、[戻る] を選択して変更を編集するか、[キャンセル] を選択して変更をキャンセルします。

AWS CLI

AWS CLI を使用して DB インスタンスまたはマルチ AZ DB クラスターの CA を rds-ca-2019 から rds-ca-rsa2048-g1 に変更するには、modify-db-instance または modify-db-cluster コマンドを呼び出します。DB インスタンスまたはクラスター識別子および `--ca-certificate-identifier` オプションを指定します。

`--apply-immediately` パラメータを使用して更新を直ちに適用します。デフォルトで、このオペレーションは次のメンテナンスウィンドウの間に実行するようスケジュールされています。

> **Important**
>
> このオペレーションをスケジュールする場合は、必ずクライアント側の信頼ストアを事前に更新します。

Example

DB インスタンス

次の例では、CA 証明書を `rds-ca-rsa2048-g1` に設定することにより、`mydbinstance` を変更しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> インスタンスを再起動する必要がある場合、modify-db-instance CLI コマンドを使用して、`--no-certificate-rotation-restart` オプションを指定できます。

Example

マルチ AZ DB クラスター

次の例では、CA 証明書を rds-ca-rsa2048-g1 に設定することにより、mydbcluster を変更しています。

Linux、macOS、Unix の場合:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Windows の場合:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

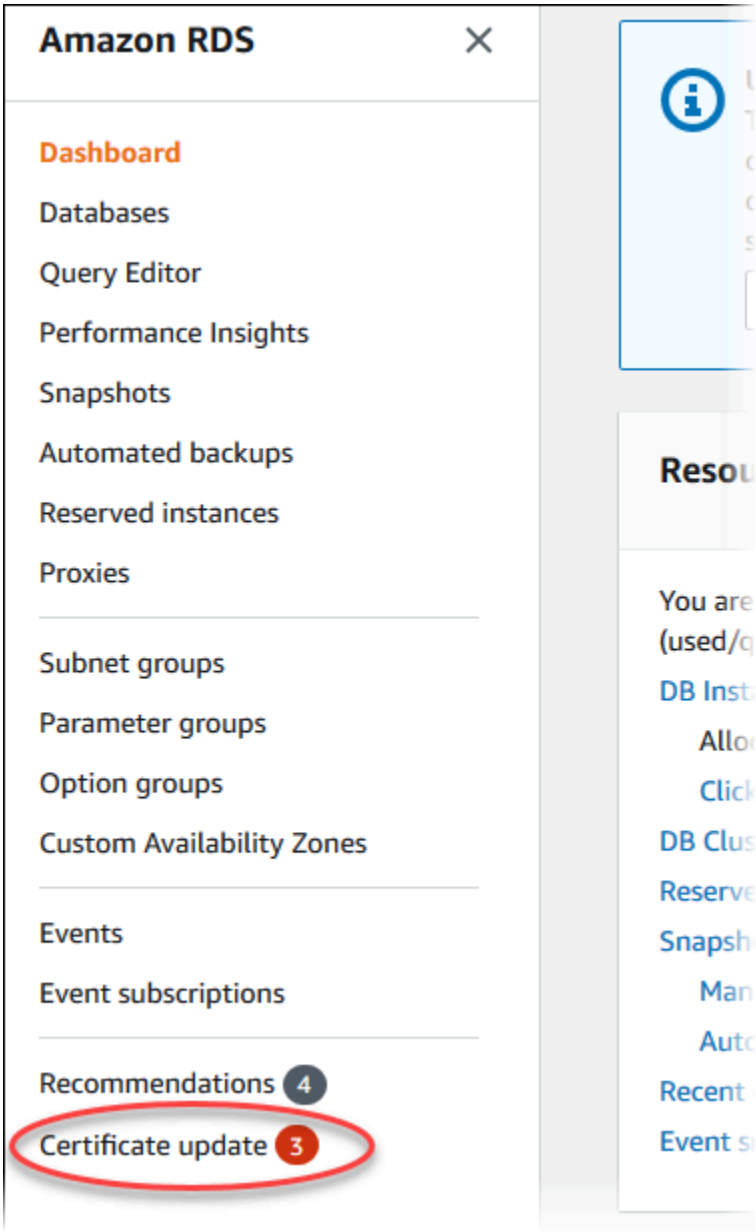
メンテナンスの適用による CA 証明書の更新

メンテナンスを適用して CA 証明書を更新するには、次のステップを実行します。

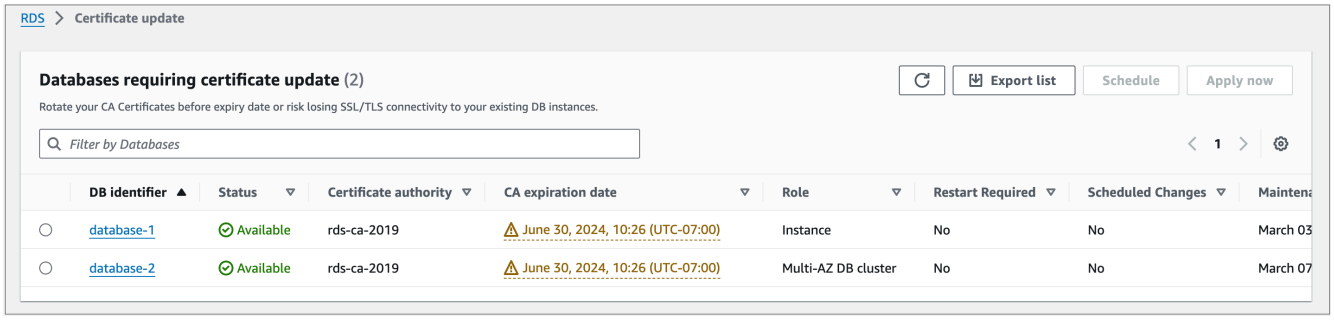
Console

メンテナンスを適用して CA 証明書を更新するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで [証明書の更新] を選択します。



[証明書の更新が必要なデータベース] ページが表示されます。



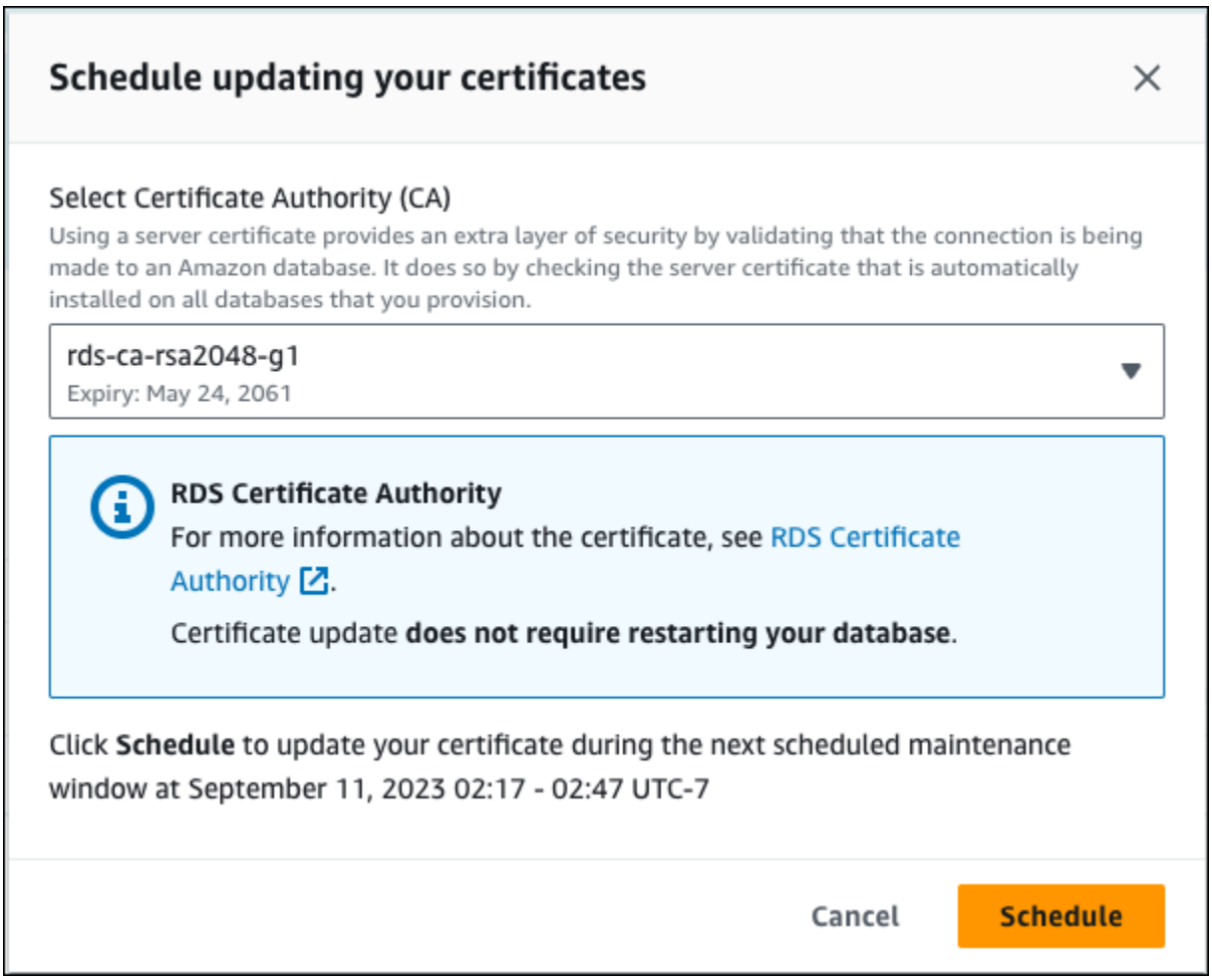
> **Note**
>
> このページには、現在の AWS リージョン リージョンの DB インスタンスおよびクラスターのみが表示されます。複数の AWS リージョン にデータベースがある場合は、このページを AWS リージョン ごとにチェックし、古い SSL/TLS 証明書を使用しているすべての DB インスタンスを確認します。

3. 更新する DB インスタンスまたはマルチ AZ DB クラスターを選択します。

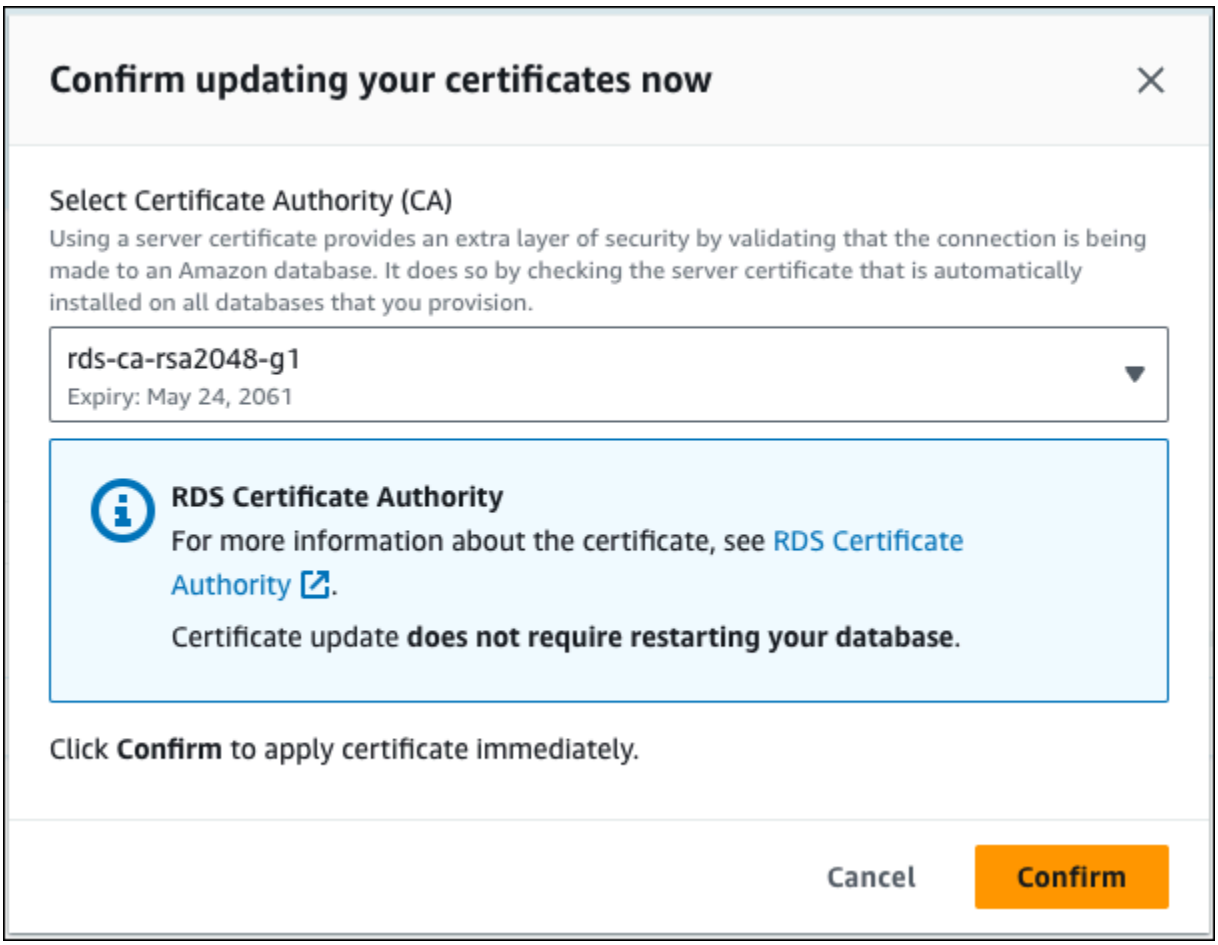
   [スケジュール] を選択すると、次のメンテナンスウィンドウでの証明書の更新をスケジュールできます。[今すぐ適用] を選択して、直ちに更新を適用します。

   > **Important**
   >
   > 証明書の有効期限が切れた後に接続の問題が発生した場合は、[今すぐ適用] オプションを使用します。

4. a. [スケジュール] を選択すると、CA 証明書のローテーションの確認を求められます。このプロンプトには、アップデートのスケジュール期間も記載されています。



b. [今すぐ適用] を選択すると、CA 証明書のローテーションの確認を求められます。



> **Important**
>
> データベースでの CA 証明書の更新をスケジュールする前に、SSL/TLS とサーバー証明書を接続に使用するクライアントアプリケーションを更新します。これらの更新は、DB エンジンに固有です。これらのクライアントアプリケーションを更新したら、CA 証明書の更新を確認できます。

続行するには、チェックボックスをオンにし、[確認] を選択します。

5. 更新する DB インスタンスおよびクラスターごとに、ステップ 3 と 4 を繰り返します。

サーバー証明書の自動ローテーション

ルート CA がサーバー証明書の自動ローテーションをサポートしている場合、RDS は DB サーバー証明書のローテーションを自動的に処理します。RDS はこの自動ローテーションに同じルート CA

を使用するため、新しい CA バンドルをダウンロードする必要はありません。「認証局」を参照してください。

DB サーバー証明書のローテーションと有効期間は DB エンジンによって異なります。

- DB エンジンが再起動なしのローテーションをサポートしている場合、ユーザーによるアクションがなくても、RDS は DB サーバー証明書を自動的にローテーションします。RDS は、DB サーバー証明書の半減期になった時点で、希望するメンテナンス期間に DB サーバー証明書のローテーションを試みます。新しい DB サーバー証明書は、12 か月間有効です。
- DB エンジンが再起動せずにローテーションをサポートしていない場合、Amazon RDS は、証明書の半分、または有効期限の少なくとも 3 か月前に、Describe-pending-maintenance-actions API を介して `server-certificate-rotation` 保留中のメンテナンスアクションを表示します。apply-pending-maintenance-action API を使用してローテーションを適用できます。新しい DB サーバー証明書は、36 か月間有効です。

describe-db-engine-versions コマンドを使用して、`SupportsCertificateRotationWithoutRestart` フラグを点検することで、再起動なしで DB エンジンバージョンが証明書のローテーションをサポートするかどうかを特定します。詳細については、「データベースに CA を設定する」を参照してください。

> **Important**
>
> Amazon RDS for Oracle DB インスタンスの場合、`FALSE` とマークされた DB エンジンバージョンの `SupportsCertificateRotationWithoutRestart` フラグが表示されます。ただし、Amazon RDS for Oracle DB インスタンスは再起動する必要はありませんが、サーバー証明書のローテーション中にデータベースリスナーが再起動されます。既存のデータベース接続は影響を受けませんが、新しい接続ではリスナーが再起動されるまで短期間だけエラーが発生します。サーバー証明書を手動でローテーションする場合は、apply-pending-maintenance-action AWS CLI コマンドを使用します。

証明書を信頼ストアにインポートするためのサンプルスクリプト

次のサンプルシェルスクリプトでは、証明書バンドルを信頼ストア内にインポートします。

各サンプルシェルスクリプトでは、Java 開発キット (JDK) の一部である keytool が使用されます。JDK のインストールの詳細については、「JDK インストールガイド」を参照してください。

Linux

Linux オペレーティングシステムで、証明書バンドルを信頼ストアにインポートするサンプルシェルスクリプトを次に示します。

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

macOS で証明書バンドルを信頼ストアにインポートするサンプルシェルスクリプトを次に示します。

```
mydir=tmp/certs
```

```
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

ネットワーク間のトラフィックのプライバシー

Amazon RDS とオンプレミスのアプリケーション間、および同じ AWS リージョン内の Amazon RDS と他の AWS リソース間では、接続が保護されています。

サービスとオンプレミスのクライアントおよびアプリケーションとの間のトラフィック

プライベートネットワークと AWS との間には 2 つの接続オプションがあります

- AWS Site-to-Site VPN 接続。詳細については、「AWS Site-to-Site VPN とは」を参照してください。
- Direct Connect 接続。詳細については、「Direct Connect とは?」を参照してください。

AWS が公開した API オペレーションを使用することにより、ネットワークを通じて、Amazon RDS へのアクセスを取得できます。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon RDS での Identity and Access Management

AWS Identity and Access Management (IAM) は管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰 (のサインイン) を認証して、Amazon RDS リソースの使用を承認 (許可) するかを管理します。IAM は、追加費用なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon RDS と IAM の連携
- Amazon RDS のアイデンティティベースのポリシーの例
- AWSAmazon RDS の マネージドポリシー
- Amazon RDS の AWS 管理ポリシーに関する更新
- サービス間での混乱した代理問題の防止
- MariaDB、MySQL、および PostgreSQL の IAM データベース認証
- Amazon RDS のアイデンティティおよびアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon RDS で行う作業によって異なります。

サービスユーザー - ジョブを実行するために Amazon RDS サービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。作業を実行するためにさらに多くの Amazon RDS 機能を使用するとき、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切なアクセス許可をリクエストするのに役に立ちます。Amazon RDS の機能にアクセスできない場合は、「Amazon RDS のアイデンティティおよびアクセスのトラブルシューティング」を参照してください。

サービス管理者 - 社内の Amazon RDS リソースを担当している場合は、おそらく Amazon RDS へのフルアクセスがあります。従業員がどの Amazon RDS 機能とリソースアクセスする必要がある

かを決定するのは管理者の仕事です。その後で、サービスユーザーのアクセス許可を変更するために、 管理者にリクエストを送信する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。お客様の会社で Amazon RDS で IAM を利用する方法の詳細については、「Amazon RDS と IAM の連携」を参照して ください。

管理者 - 管理者は、Amazon RDS へのアクセスを管理するポリシーの作成方法の詳細について確認したい場合があります。IAM で使用できる Amazon RDS アイデンティティベースのポリシーの例を表示するには、「Amazon RDS のアイデンティティベースのポリシーの例」を参照して ください。

アイデンティティを使用した認証

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザー、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーテッドアイデンティティとしてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「How to sign in to your AWS アカウント」を参照してください。

プログラムによるアクセスの場合、AWS はリクエストに暗号で署名するための SDK と CLI を提供します。詳細については、「IAM ユーザーガイド」の「API リクエストに対する AWS Signature Version 4」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成すると、すべての AWS のサービスとリソースに対する完全なアクセス権を持つ AWS アカウントルートユーザーと呼ばれる 1 つのサインイン ID を使用して開始します。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスでは、人間のユーザーが一時的な認証情報を使用して AWS のサービス にアクセスする際、アイデンティティプロバイダーとのフェデレーションを使用することが求められます。

フェデレーテッドアイデンティティは、エンタープライズディレクトリ、ウェブ ID プロバイダー、Directory Service のユーザーであり、ID ソースからの認証情報を使用して AWS のサービスに

アクセスするユーザーです。フェデレーティッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、AWS IAM アイデンティティセンター をお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「What is IAM Identity Center?」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細は「IAM ユーザーガイド」の「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。

IAM グループは IAM ユーザーのコレクションを指定し、大量のユーザーのアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM データベース認証を使用して、DB インスタンスを認証できます。

IAM データベース認証は、次の DB エンジンで使用できます。

- RDS for MariaDB
- RDS for MySQL
- RDS for PostgreSQL

IAM を使用した DB インスタンスの認証の詳細については、「MariaDB、MySQL、および PostgreSQL の IAM データベース認証」を参照してください。

IAM ロール

IAM ロールは、特定の許可を持つ、AWS アカウント内のアイデンティティです。これはユーザーに似ていますが、特定のユーザーに関連付けられていません。ロールを切り替えることによって、AWS マネジメントコンソールで IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLIまたは AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- 一時的なユーザーアクセス許可 - ユーザーは、特定のタスクのための複数の異なるアクセス許可を一時的に受け取るために、IAM ロールを引き受けることができます。
- フェデレーションユーザーアクセス – フェデレーテッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーテッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM アイデンティティセンター User Guide」の「Permission sets」を参照してください。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスでのロールとリソースベースのポリシーの違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス権 - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション – 転送アクセスセッション (FAS) は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービス に許可を委任するロールを作成する」を参照してください。
  - サービスにリンクされたロール - サービスにリンクされたロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

IAM ロールを使用するべきかどうかについては、IAM ユーザーガイドの「IAM ロールの作成が適している場合 (ユーザーではなく)」を参照してください。

ポリシーを使用したアクセスの管理

AWS でのアクセスは、ポリシーを作成し、それらを IAM アイデンティティまたは AWS リソースに添付することで制御できます。ポリシーは AWS のオブジェクトであり、ID やリソースに関連付けられて、これらのアクセス許可を定義します。AWS は、エンティティ (ルートユーザー、ユーザー、または IAM ロール) によってリクエストが行われると、それらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、AWS リソースへのアクセス権を持つユーザーと、これらのリソースに対して実行できるアクションを指定できます。すべての IAM エンティティ (アクセス許可セットまたはロール) は、アクセス許可のない状態からスタートします。言い換えると、デフォルト設定では、ユーザーは何もできず、自分のパスワードを変更することすらできません。何かを実行する許可をユーザーに付与するには、管理者がユーザーに許可ポリシーをアタッチする必要があります。また、管理者は、必要な許可があるグループにユーザーを追加できます。管理者がグループに許可を付与すると、そのグループ内のすべてのユーザーにこれらの許可が付与されます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS マネジメントコンソール、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

ID ベースのポリシーは、 アクセス許可セットやロールなどの ID にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティが実行できるアクション、

リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの IAM ポリシーの作成を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のアクセス許可セットまたはロールに直接埋め込まれます。マネージドポリシーは、AWS アカウント内の複数のアクセス許可セットおよびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、IAM ユーザーガイドのマネージドポリシーとインラインポリシーの比較を参照してください。

Amazon RDSに固有の AWS マネージドポリシーの詳細については、「AWSAmazon RDS の マネージドポリシー」を参照してください。

他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、ID ベースのポリシーによって IAM エンティティ (アクセス許可セットまたはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティに許可の境界を設定できます。結果として許可される範囲は、エンティティのアイデンティティベースポリシーとその許可の境界の共通部分になります。`Principal` フィールドでアクセス許可セットまたはロールを指定するリソースベースのポリシーは、アクセス許可の境界によって制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) – SCP は、AWS Organizations で組織や組織単位 (OU) に最大アクセス許可を指定する JSON ポリシーです。AWS Organizationsは、お客様のビジネスが所有する複数の AWS アカウントをグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、AWS Organizationsユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてのセッションのアクセス許可は、アクセス許可セットまたはロールの ID ベースのポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合

もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、IAM ユーザーガイドのポリシーの評価ロジックを参照してください。

Amazon RDS と IAM の連携

IAM を使用して、Amazon RDS へのアクセスを管理するには、Amazon RDS で使用できる IAM の機能を理解しておく必要があります。

次の表は、Amazon RDS で使用できる IAM 機能の一覧です。

IAM 機能	Amazon RDS のサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	いいえ
属性ベースのアクセスコントロール (ABAC) (ポリシーのタグ)	はい
一時的な認証情報	はい
転送アクセスセッション	はい
サービスロール	あり

IAM 機能	Amazon RDS のサポート
サービスリンクロール	はい

Amazon RDS や AWS の他のサービスと IAM との連携の概要については、IAM ユーザーガイドの「IAM と連携する AWS のサービス」を参照してください。

トピック

- Amazon RDS アイデンティティベースのポリシー
- Amazon RDS 内のリソースベースのポリシー
- Amazon RDS のポリシーアクション
- Amazon RDS のポリシーリソース
- Amazon RDS のポリシー条件キー
- Amazon RDS のアクセスコントロールリスト (ACL)
- Amazon RDS タグを使ったポリシーにおける属性ベースのアクセスコントロール (ABAC)
- Amazon RDS での一時的な認証情報の使用
- Amazon RDS のフォワードアクセスセッション
- Amazon RDS のサービスロール
- Amazon RDS のサービスリンクロール

Amazon RDS アイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり。

アイデンティティベースのポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、IAM ユーザーガイドの「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon RDS アイデンティティベースのポリシーの例

Amazon RDS アイデンティティベースのポリシーの例を表示するには、「Amazon RDS のアイデンティティベースのポリシーの例」を参照してください。

Amazon RDS 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし。

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

Amazon RDS のポリシーアクション

ポリシーアクションのサポート: あり。

管理者は AWS JSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのようなリソースにどのような条件でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Amazon RDS のポリシーアクションは、アクションの前にプレフィックス `rds:` を使用します。例えば、Amazon RDS `DescribeDBInstances` API オペレーションを使用して DB インスタンスを指定するアクセス許可を付与するには、ポリシーに `rds:DescribeDBInstances` アクションを含めます。ポリシーステートメントには、`Action` または `NotAction` エレメントを含める必要があります。Amazon RDS は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには、次のようにコンマで区切ります。

```
"Action": [
      "rds:action1",
      "rds:action2"
```

ワイルドカード *を使用して複数のアクションを指定することができます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "rds:Describe*"
```

Amazon RDS アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon RDS によって定義されるアクション」を参照してください。

Amazon RDS のポリシーリソース

ポリシーリソースのサポート: あり。

管理者は AWS JSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

DB インスタンスリソースには、次の Amazon リソースネーム (ARN) があります。

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

ARN の形式の詳細については、「Amazon リソースネーム ARN と AWS のサービスの名前空間」を参照してください。

例えば、ステートメントで dbtest DB インスタンスを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

特定のアカウントに属するすべての DB インスタンスを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

リソースの作成など、一部の RDS API オペレーションは、特定のリソースで実行できません。このような場合は、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon RDS API オペレーションの多くが複数のリソースと関連します。例えば、`CreateDBInstance` では、DB インスタンスが作成されます。DB インスタンス作成時に特定のセキュリティグループおよびパラメータグループを使用するように ユーザーに義務付けることができます。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
      "resource1",
      "resource2"
```

Amazon RDS のリソースタイプおよび ARN のリストについては、「サービス認可リファレンス」の「Amazon RDS で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon RDS で定義されるアクション」を参照してください。

Amazon RDS のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり。

管理者は AWS JSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイドのAWS グローバル条件コンテキストキーを参照してください。

Amazon RDS は独自の条件キーを定義し、一部のグローバル条件キーの使用をサポートしています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイドの「AWS グローバル条件コンテキストキー」を参照してください。

すべての RDS API オペレーションは、aws:RequestedRegion 条件キーをサポートします。

Amazon RDS の条件キーのリストについては、「サービス認可リファレンス」の「Amazon RDS の条件キー」を参照してください。どのアクションおよびリソースと条件キーを使用できるかについては、「Amazon RDS で定義されるアクション」を参照してください。

Amazon RDS のアクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) をサポート: なし。

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon RDS タグを使ったポリシーにおける属性ベースのアクセスコントロール (ABAC)

ポリシーにおける属性ベースのアクセスコントロール (ABAC) タグをサポート: あり。

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグを付けることで、プリンシパルのタグがリソースタグと一致するときに操作を許可する ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon RDS リソースのタグ付けの詳細については、「条件の指定: カスタムタグの使用」を参照してください。リソースのタグに基づいてリソースへのアクセスを制限するためのアイデンティティベースのポリシーの例を表示するには、「2 つの異なる値を持つタグが付いたリソースに対するアクションにアクセス許可を付与する」を参照してください。

Amazon RDS での一時的な認証情報の使用

一時的な認証情報のサポート: あり。

一時的な認証情報は、AWSリソースへの短期的なアクセスを提供し、フェデレーションの使用時またはロールの切り替え時に自動的に作成されます。AWS では、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

Amazon RDS のフォワードアクセスセッション

転送アクセスセッションをサポート: あり。

転送アクセスセッション (FAS) は、AWS のサービス を呼び出すプリンシパルのアクセス許可を AWS のサービス のリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon RDS のサービスロール

サービスロールのサポート: あり。

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービス に許可を委任するロールを作成するを参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon RDS のサービスロール の機能が破損する可能性があります。Amazon RDS が指示する場合以外は、サービスロールを編集しないでください。

Amazon RDS のサービスリンクロール

サービスリンクロールのサポート: あり。

サービスにリンクされたロールは、AWS のサービス にリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。

サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

Amazon RDS サービスにリンクされたロールの使用の詳細については、「Amazon RDS のサービスにリンクされたロールの使用」を参照してください。

Amazon RDS のアイデンティティベースのポリシーの例

デフォルトでは、アクセス許可セットとロールには、Amazon RDS リソースを作成または変更するアクセス許可はありません。AWS マネジメントコンソール、AWS CLI、または AWS API を使用してタスクを実行することもできません。管理者は、指定されたリソースに対して特定の API オペレーションを実行するために必要なアクセス許可をアクセス許可セットとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者は、それらのアクセス許可を必要とするアクセス許可セットまたはロールに、そのポリシーをアタッチします。

これらの JSON ポリシードキュメント例を使用して IAM のアイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon RDS コンソールの使用
- コンソールの使用に必要なアクセス許可
- 自分の権限の表示をユーザーに許可する
- Amazon RDS でリソースを作成、変更、削除するアクセス許可ポリシー
- ポリシー例: 条件キーの使用
- 条件の指定: カスタムタグの使用
- Amazon RDS リソースの作成時にタグ付けするアクセス許可の付与

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon RDS リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションでは、AWS アカウント に費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS マネージドポリシーの使用を開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードへのアクセス許可の付与を開始するには、多くの一般的なユースケースのためにアクセス許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能の AWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、CloudFormation などの特定の AWS のサービス を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウントで IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

Amazon RDS コンソールの使用

Amazon RDS コンソールにアクセスするには、一連の最小限のアクセス許可が必要です。これらのアクセス許可により、AWS アカウント の Amazon RDS リソースの詳細をリストおよび表示できるようにする必要があります。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベース

のポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

これらのエンティティが Amazon RDS コンソールを引き続き使用できるように、エンティティに次の AWS 管理ポリシーもアタッチします。

```
AmazonRDSReadOnlyAccess
```

詳細については、IAM ユーザーガイド の「ユーザーへのアクセス許可の追加」を参照してください。

コンソールの使用に必要なアクセス許可

コンソールを使用するユーザーには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、ユーザーは AWS アカウントの Amazon RDS リソースを記述し、Amazon EC2 セキュリティやネットワーク情報など、その他の関連情報を提供できます。

これらの最小限必要なアクセス権限よりも制限された IAM ポリシーを作成している場合、その IAM ポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しません。「AmazonRDSReadOnlyAccess」で説明されているとおり、ユーザーがコンソールを使用できること、および ポリシーを使用したアクセスの管理 管理ポリシーがユーザーにアタッチされていることを確認してください。

AWS CLI または Amazon RDS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。

以下のポリシーでは、ルート AWS アカウントの Amazon RDS リソースへのフルアクセスが付与されます。

```
AmazonRDSFullAccess
```

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、

または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon RDS でリソースを作成、変更、削除するアクセス許可ポリシー

以下のセクションでは、リソースへのアクセスを許可および制限するアクセス許可ポリシーの例を示します。

AWS アカウントでの DB インスタンスの作成をユーザーに許可する

以下は、123456789012 アカウントで ID が AWS のアカウントが DB インスタンスを作成できるようにするポリシーの例です。ポリシーは、test で始める新しい DB インスタンスの名前である必要があります。また、新しい DB インスタンスは、MySQL データベースエンジンと DB インスタンスの db.t2.micro クラスを使用する必要があります。さらに、新しい DB インスタンスでは、オプショングループと default で始まる DB パラメータグループ、および default サブネットグループを使用する必要があります。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
   ]
}
```

ポリシーには、 ユーザー用の以下のアクセス許可を指定する単一のステートメントが含まれます。

- ポリシーを使用すると、アカウントは CreateDBInstance API オペレーションを使用して DB インスタンスを作成できます (これは create-db-instance AWS CLI コマンドと AWS マネジメントコンソール にも適用されます)。
- Resource 要素では、ユーザーがリソースでアクションを実行できることを指定できます。Amazon Resources Name (ARN) を使用してリソースを指定します。この ARN には、リソースが属しているサービスの名前 (rds)、AWS リージョン (* はこの例のリージョンを示します)、AWS アカウント番号 (123456789012 はこの例のアカウント番号です)、およびリソースのタイプが含まれます。ARN の作成の詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

  例の Resource 要素は、ユーザーのリソースで、以下のポリシーの制約を指定します。

  - 新しい DB インスタンスの DB インスタンス識別子は、test で始まる必要があります (例: testCustomerData1、test-region2-data)。
  - 新しい DB インスタンスのオプショングループは、default で始まる必要があります。
  - 新しい DB インスタンスの DB パラメータグループは、default で始まる必要があります。
  - 新しい DB インスタンスのサブネットグループは、default サブネットグループである必要があります。
- Condition 要素は、DB エンジンが MySQL で、DB インスタンスクラスが db.t2.micro である必要があることを指定します。Condition 要素は、ポリシーが有効になる条件を指定します。Condition 要素を使用して、アクセス許可または制約を追加できます。条件を指定する方法については、「Amazon RDS のポリシー条件キー」を参照してください。この例では、rds:DatabaseEngine および rds:DatabaseClass を条件として指定します。rds:DatabaseEngine の有効な条件値については、CreateDBInstance の Engine パラメータのリストを参照してください。rds:DatabaseClass の有効な条件値については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。

アイデンティティベースのポリシーでアクセス権限を得るプリンシパルを指定していないため、ポリシーでは Principal 要素を指定していません。ユーザーにポリシーをアタッチすると、そのユーザーが暗黙のプリンシパルになります。IAM ロールにアクセス権限ポリシーをアタッチすると、ロールの信頼ポリシーで識別されたプリンシパルがアクセス権限を得ることになります。

Amazon RDS アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon RDS によって定義されるアクション」を参照してください。

RDS リソースに対する Describe アクションの実行をユーザーに許可する

以下のアクセス権限ポリシーは、Describe で始まるすべてのアクションを実行するためのアクセス権限をユーザーに付与します。これらのアクションは、DB インスタンスなど RDS リソースに関する情報を表示します。Resource 要素内のワイルドカード文字 (*) は、アカウントによって所有されるすべての Amazon RDS リソースに対してそれらのアクションが許可されることを示します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

指定した DB パラメータグループとサブネットグループを使用する DB インスタンスの作成をユーザーに許可する

以下の許可ポリシーは、mydbpg DB パラメータグループと mydbsubnetgroup DB サブネットグループを使用する必要のある DB インスタンスを作成することのみをユーザーに許可するための許可を付与します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
```

```
         ]
      }
   ]
}
```

2 つの異なる値を持つタグが付いたリソースに対するアクションにアクセス許可を付与する

アイデンティティベースのポリシーの条件を使用して、タグに基づいて Amazon RDS リソースへのアクセスを制御できます。次のポリシーでは、`stage` タグが `development` または `test` に設定された DB インスタンスに対して `CreateDBSnapshot` API オペレーションを実行するためのアクセス許可が付与されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
```

```
   ]
}
```

次のポリシーでは、stage タグが development または test に設定された DB インスタンスに対して ModifyDBInstance API オペレーションを実行するためのアクセス許可が付与されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

ユーザーによる DB インスタンスの削除を禁止する

以下のアクセス権限ポリシーは、特定の DB インスタンスを削除することをユーザーに禁止するためのアクセス権限を付与します。例えば、管理者以外のすべてのユーザーに対して、本稼働 DB インスタンスの削除を拒否することができます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

リソースへのすべてのアクセスを拒否する

リソースへのアクセスを明示的に拒否できます。拒否ポリシーは許可ポリシーよりも優先されます。以下のポリシーは、リソースを管理する機能をユーザーに明示的に拒否します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
```

```
   ]
}
```

ポリシー例: 条件キーの使用

以下に示しているのは、Amazon RDS IAM アクセス許可ポリシーでの条件キーの使用例です。

例 1: 特定の DB エンジンを使用し、マルチ AZ ではない DB インスタンスを作成するためのアクセス許可を付与する

以下のポリシーでは、RDS 条件キーを使用して、MySQL データベースエンジンを使用するがマルチ AZ でない DB インスタンスのみをユーザーが作成できるようにします。Condition 要素では、データベースエンジンが MySQL であることが要件になることを示しています。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

例 2: 特定の DB インスタンスクラスの DB インスタンスを作成するためのアクセス許可と、プロビジョンド IOPS を使用する DB インスタンスを作成するためのアクセス許可を明示的に拒否する

以下のポリシーでは、最もサイズが大きくてコストの高いインスタンスである DB インスタンスクラス `r3.8xlarge` と `m4.10xlarge` を使用する DB インスタンスの作成のためのアクセス許可を明示的に拒否しています。このポリシーでは、追加のコストが発生するプロビジョンド IOPS を使用する DB インスタンスの作成もユーザーに禁止しています。

明示的に拒否するアクセス権限は、付与する他のいずれのアクセス権限よりも優先されます。これにより、決して付与されることのないアクセス権限を ID が誤って取得することがなくなります。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
            }
         }
      }
   ]
```

```
}
```

例 3: リソースにタグを付けるために使用できるタグキーと値のセットを制限する

次のポリシーは、RDS 条件キーを使用し、キー `stage` を持つタグの追加を値 `test`、`qa`、および `production` を持つリソースに追加することができます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagEdits",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "arn:aws:rds:us-east-1:123456789012:db:db-123456",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/stage": [
            "test",
            "qa",
            "production"
          ]
        }
      }
    }
  ]
}
```

条件の指定: カスタムタグの使用

Amazon RDS では、カスタムタグを使用して IAM ポリシーで条件を指定することがサポートされています。

例えば、environment という名前のタグを、beta、staging、production などの値で DB インスタンスに追加するとします。追加する場合、特定のユーザーを environment タグ値に基づく DB インスタンスに制限するポリシーを作成することができます。

> **Note**
>
> カスタムタグ識別子は大文字と小文字が区別されます。

以下の表では、Condition 要素で使用できる RDS タグ識別子を示しています。

RDS タグ識別子	適用先
db-tag	リードレプリカを含む DB インスタンス
snapshot-tag	DB スナップショット
ri-tag	リザーブド DB インスタンス
og-tag	DB オプショングループ
pg-tag	DB パラメータグループ
subgrp-tag	DB サブネットグループ
es-tag	イベントサブスクリプション
cluster-tag	DB クラスター
cluster-pg-tag	DB クラスターのパラメータグループ
cluster-snapshot-tag	DB クラスタースナップショット

カスタムタグの条件の構文は次のとおりです。

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name":
["value"]} }
```

例えば、次の Condition 要素は、environment という名前のタグを持ち、タグの値が production である DB インスタンスに適用されます。

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

タグの作成の詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

> **Important**
>
> タグを使用して RDS リソースへのアクセスを管理する場合は、RDS リソースのタグへのアクセスを保護することをお勧めします。AddTagsToResource および RemoveTagsFromResource アクションのポリシーを作成することによって、タグへのアクセスを管理できます。例えば、次のポリシーは、ユーザーがすべてのリソースのタグを追加または削除することを拒否します。次に、特定のユーザーがタグを追加または削除することを許可するポリシーを作成できます。
>
> JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Amazon RDS アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon RDS によって定義されるアクション」を参照してください。

ポリシー例: カスタムタグの使用

以下に示しているのは、Amazon RDS IAM アクセス許可ポリシーでのカスタムタグの使用例です。Amazon RDS リソースへのタグの追加の詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

> **Note**
>
> すべての例で、us-west-2 リージョンを使用し、架空のアカウント ID を含めています。

例 1: 2 つの異なる値を持つタグが付いたリソースに対するアクションにアクセス許可を付与する

次のポリシーでは、`stage` タグが `development` または `test` に設定された DB インスタンスに対して `CreateDBSnapshot` API オペレーションを実行するためのアクセス許可が付与されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

次のポリシーでは、stage タグが development または test に設定された DB インスタンスに対して ModifyDBInstance API オペレーションを実行するためのアクセス許可が付与されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
          "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
            ]
       },
       {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
               "rds:db-tag/stage":[
                  "development",
                  "test"
                  ]
               }
            }
       }
    ]
}
```

例 2: 指定した DB パラメータグループを使用する DB インスタンスを作成するためのアクセス許可を明示的に拒否する

以下のポリシーでは、特定のタグ値が設定された DB パラメータグループを使用する DB インスタンスの作成のためのアクセス権限を明示的に拒否しています。DB インスタンスを作成するときに特定のユーザー定義の DB パラメータグループの使用を必須とする場合にも、このポリシーを適用できます。Deny を使用するポリシーは、ほとんどの場合、適用範囲のより広いポリシーによって付与されるアクセス許可を制限するために使用します。

明示的に拒否するアクセス権限は、付与する他のいずれのアクセス権限よりも優先されます。これにより、決して付与されることのないアクセス権限を ID が誤って取得することがなくなります。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

例 3: インスタンス名にユーザー名がプレフィックスとして付加されている DB インスタンスに対するアクションにアクセス許可を付与する

以下のポリシーでは、DB インスタンス名の前にユーザー名が付いている DB インスタンスのうち、AddTagsToResource と同等の RemoveTagsFromResource というタグが付いているか、または stage というタグが付いていない DB インスタンスに対する、API (devo または stage を除く) の呼び出しのためのアクセス権限を付与しています。

ポリシーの Resource 行では、リソースをその Amazon Resource Name (ARN) により識別しています。ARN と Amazon RDS リソースの使用の詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

Amazon RDS リソースの作成時にタグ付けするアクセス許可の付与

一部の RDS API オペレーションでは、リソースの作成時にタグを指定できます。リソースタグを使用して、属性ベースの制御 (ABAC) を実装できます。詳細については、「AWS のABAC とは」および「タグを使用した AWS リソースへのアクセスの制御」を参照してください。

ユーザーがリソースの作成時にタグを付けるには、リソースを作成するアクション (rds:CreateDBInstance など) を使用するためのアクセス許可が必要です。作成アクションでタグが指定されている場合、RDS は rds:AddTagsToResource アクションに対して追加の認可を実行して、ユーザーがタグを作成する認可を持っているかどうかを確認します。そのため、ユーザーにはrds:AddTagsToResource アクションを使用する明示的なアクセス権限が必要です。

rds:AddTagsToResource アクションの IAM ポリシー定義では、aws:RequestTag 条件キーを使用して、リソースにタグを付けるリクエストでタグを要求できます。

例えば、次のポリシーでは、特定のタグキー (environment または project) でのみ、ユーザーが DB インスタンスを作成し、DB インスタンスの作成中にタグを適用することが許可されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:RequestTag/environment": ["production", "development"],
                   "aws:RequestTag/project": ["dataanalytics", "webapp"]
               },
               "ForAllValues:StringEquals": {
                   "aws:TagKeys": ["environment", "project"]
               }
           }
       }
   ]
}
```

このポリシーは、environment または project タグ以外のタグを含む DB インスタンスの作成リクエスト、またはこれらのタグのいずれかを指定しない DB インスタンスの作成リクエストを拒否します。さらに、ユーザーはポリシーで許可されている値と一致するタグの値を指定する必要があります。

次のポリシーでは、ユーザーが DB クラスターを作成し、作成時に environment=prod タグ以外のタグを適用することを許可します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBCluster"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringNotEquals": {
                   "aws:RequestTag/environment": "prod"
               }
           }
       }
   ]
}
```

作成時のタグ付けでサポートされている RDS API アクション

次の RDS API アクションでは、リソース作成時のタグの指定がサポートされます。これらのアクションでは、リソースの作成時にタグを指定できます。

- CreateBlueGreenDeployment
- CreateCustomDBEngineVersion
- CreateDBCluster
- CreateDBClusterEndpoint
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance

- CreateDBInstanceReadReplica
- CreateDBParameterGroup
- CreateDBProxy
- CreateDBProxyEndpoint
- CreateDBSecurityGroup
- CreateDBShardGroup
- CreateDBSnapshot
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- CreateIntegration
- CreateOptionGroup
- CreateTenantDatabase
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CopyDBParameterGroup
- CopyDBSnapshot
- CopyOptionGroup
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime
- PurchaseReservedDBInstancesOffering

AWS CLI または API を使用してタグ付きのリソースを作成する場合、Tags パラメータを使用して作成時にリソースにタグを適用します。

これらの API アクションでは、タグ付けが失敗した場合、リソースは作成されず、リクエストはエラーで失敗します。これにより、リソースがタグ付きで作成されるか、まったく作成されないため、意図したタグのないリソースは作成されません。

AWSAmazon RDS の マネージドポリシー

アクセス許可セットとロールにアクセス許可を追加するには、自分でポリシーを作成するよりも、AWS マネージドポリシーを使用する方が簡単です。チームに必要な権限のみを提供する IAM カスタマーマネージドポリシーを作成するには時間と専門知識が必要です。すぐに使用を開始するために、AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、AWS アカウント で利用できます。AWS マネージドポリシーの詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS のサービス は、AWS マネージドポリシーを維持し、更新します。AWS マネージドポリシーの権限を変更することはできません。サービスでは新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、ポリシーがアタッチされている、すべてのアイデンティティ (アクセス許可セットとロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは、AWS マネージドポリシーから許可を削除しないため、ポリシーの更新によって既存の許可が破棄されることはありません。

さらに、AWS は複数のサービスにまたがるジョブ機能の特徴に対するマネージドポリシーもサポートしています。例えば、`ReadOnlyAccess` AWS マネージドポリシーでは、すべての AWS のサービス およびリソースへの読み取り専用アクセスを許可します。あるサービスで新しい機能を立ち上げる場合は、AWS は、追加された演算とリソースに対し、読み込み専用の許可を追加します。職務機能ポリシーのリストと説明については、IAM ユーザーガイドの「ジョブ機能の AWS マネージドポリシー」を参照してください。

トピック

- AWS マネージドポリシー: AmazonRDSReadOnlyAccess
- AWS マネージドポリシー: AmazonRDSFullAccess
- AWS マネージドポリシー: AmazonRDSDataFullAccess
- AWS マネージドポリシー: AmazonRDSEnhancedMonitoringRole
- AWS マネージドポリシー: AmazonRDSPerformanceInsightsReadOnly
- AWS 管理ポリシー: AmazonRDSPerformanceInsightsFullAccess
- AWS マネージドポリシー: AmazonRDSDirectoryServiceAccess
- AWS マネージドポリシー: AmazonRDSServiceRolePolicy
- AWS マネージドポリシー: AmazonRDSCustomServiceRolePolicy
- AWS マネージドポリシー: AmazonRDSCustom Instance ProfileRolePolicy

- AWS マネージドポリシー: AmazonRDSPreviewServiceRolePolicy
- AWS マネージドポリシー: AmazonRDSBetaServiceRolePolicy

AWS マネージドポリシー: AmazonRDSReadOnlyAccess

このポリシーは、AWS マネジメントコンソール を通じた Amazon RDS への読み取り専用アクセスを許可します。

許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `rds` — プリンシパルが Amazon RDS リソースを記述し、Amazon RDS リソースのタグを一覧表示することを許可します。
- `cloudwatch` — プリンシパルが Amazon CloudWatch メトリクスの統計情報を取得することを許可します。
- `ec2` — プリンシパルがアベイラビリティーゾーンとネットワークリソースを記述することを許可します。
- `logs` — プリンシパルがロググループの CloudWatch Logs ログストリームを記述し、CloudWatch Logs ログイベントを取得することを許可します。
- `devops-guru` - プリンシパルが Amazon DevOps Guru の対象となるリソースを記述できるようにします。リソースは、CloudFormation スタック名またはリソースタグで指定されます。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSReadOnlyAccess」を参照してください。

AWS マネージドポリシー: AmazonRDSFullAccess

このポリシーは、AWS マネジメントコンソール を通じて Amazon RDS へのフルアクセスを提供します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `rds` - プリンシパルに Amazon RDS へのフルアクセスを許可します。
- `application-autoscaling` — プリンシパルがアプリケーションオートスケーリングのターゲットとポリシーを記述し、管理することを許可します。

- `cloudwatch` — プリンシパルが CloudWatch メトリクスの統計を取得し、CloudWatch アラームを管理することを許可します。
- `ec2` — プリンシパルがアベイラビリティーゾーンとネットワークリソースを記述することを許可します。
- `logs` — プリンシパルがロググループの CloudWatch Logs ログストリームを記述し、CloudWatch Logs ログイベントを取得することを許可します。
- `outposts` — プリンシパルが AWS Outposts インスタンスタイプを取得することを許可します。
- `pi` — プリンシパルが Performance Insights メトリクスを取得することを許可します。
- `sns` — プリンシパルが Amazon Simple Notification Service (Amazon SNS) のサブスクリプションとトピックにアクセスして、Amazon SNS メッセージを発行することを許可します。
- `devops-guru` - プリンシパルが Amazon DevOps Guru の対象となるリソースを記述できるようにします。リソースは、CloudFormation スタック名またはリソースタグで指定されます。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSFullAccess」を参照してください。

AWS マネージドポリシー: AmazonRDSDataFullAccess

このポリシーは、特定の AWS アカウント 内の Aurora Serverless クラスターに対して Data API とクエリエディタを使用するためのフルアクセスを許可します。このポリシーは、AWS アカウント が AWS Secrets Manager からシークレットの値を取得することを許可します。

`AmazonRDSDataFullAccess` ポリシーを IAM IDにアタッチできます。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `dbqms` — プリンシパルにクエリへのアクセス、作成、削除、記述、および更新を許可します。データベースクエリメタデータサービス (dbqms) は、内部専用のサービスです。このサービスでは、Amazon RDS を含む複数の AWS のサービス について、最新および保存済みのクエリを、AWS マネジメントコンソール のクエリエディタ用に提供します。
- `rds-data` — プリンシパルが Aurora Serverless データベースに対して SQL ステートメントを実行するのを許可します。
- `secretsmanager` — プリンシパルが からシークレットの値を取得するのを許可します。AWS Secrets Manager

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSDataFullAccess」を参照してください。

AWS マネージドポリシー: AmazonRDSEnhancedMonitoringRole

このポリシーは、Amazon RDS 拡張モニタリング用の Amazon CloudWatch Logs へのアクセスを提供します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `logs` — プリンシパルが CloudWatch Logs ロググループと保持ポリシーを作成し、ロググループの CloudWatch Logs ログストリームを作成および記述することを許可します。また、プリンシパルが CloudWatch Logs ログイベントを設定および取得することも許可します。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSEnhancedMonitoringRole」を参照してください。

AWS マネージドポリシー: AmazonRDSPerformanceInsightsReadOnly

このポリシーは、Amazon RDS DB インスタンスと Amazon Aurora DB クラスター用の Amazon RDS Performance Insights への読み取り専用アクセスを提供します。

このポリシーには、ポリシードキュメントの識別子として `Sid` (ステートメント ID) が含まれるようになりました。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `rds` — プリンシパルが Amazon RDS DB インスタンスと Amazon Aurora DB クラスターを記述することを許可します。
- `pi` — プリンシパルが Amazon RDS Performance Insights API を呼び出し、Performance Insights メトリクスにアクセスすることを許可します。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSPerformanceInsightsReadOnly」を参照してください。

AWS 管理ポリシー: AmazonRDSPerformanceInsightsFullAccess

このポリシーは、Amazon RDS DB インスタンスと Amazon Aurora DB クラスター用の Amazon RDS Performance Insights へのフルアクセスを提供します。

このポリシーには、ポリシードキュメントの識別子として `Sid` (ステートメント ID) が含まれるようになりました。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `rds` — プリンシパルが Amazon RDS DB インスタンスと Amazon Aurora DB クラスターを記述することを許可します。
- `pi` — プリンシパルが Amazon RDS Performance Insights API を呼び出したり、パフォーマンス分析レポートを作成、表示、削除したりすることを許可します。
- `cloudwatch` — プリンシパルが Amazon CloudWatch メトリクスを一覧表示し、メトリクスデータと統計を取得するのを許可します。

JSON ポリシードキュメントを含め、このポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「AmazonRDSPerformanceInsightsFullAccess」を参照してください。

AWS マネージドポリシー: AmazonRDSDirectoryServiceAccess

このポリシーは、Amazon RDS が Directory Service を呼び出すことを許可します。

アクセス許可の詳細

このポリシーには、以下の許可が含まれています。

- `ds` — プリンシパルが Directory Service ディレクトリを記述し、Directory Service ディレクトリへの認可を制御することを許可します。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSDirectoryServiceAccess」を参照してください。

AWS マネージドポリシー: AmazonRDSServiceRolePolicy

IAM エンティティに `AmazonRDSServiceRolePolicy` をアタッチすることはできません。このポリシーは、Amazon RDS がユーザーに代わってアクションを実行することを許可するサービスリン

クロールにアタッチされます。詳細については、「Amazon RDS のサービスにリンクされたロールのアクセス許可」を参照してください。

AWS マネージドポリシー: AmazonRDSCustomServiceRolePolicy

IAM エンティティに AmazonRDSCustomServiceRolePolicy をアタッチすることはできません。このポリシーは、Amazon RDS が RDS DB リソースに代わって AWS のサービスを呼び出すことを許可するサービスにリンクされたロールに関連付けられています。

このポリシーには、以下のアクセス許可が含まれています。

- ec2 - RDS Custom が、ポイントインタイム復元機能を提供する DB インスタンスでバックアップ操作を実行することを許可します。
- secretsmanager - RDS Custom が RDS Custom によって作成された DB インスタンス固有のシークレットを管理できるようにします。
- cloudwatch - RDS Custom が CloudWatch エージェントを介して DB インスタンスメトリクスとログを CloudWatch にアップロードすることを許可します。
- events、sqs- RDS Custom が DB インスタンスに関するステータス情報を送受信できるようにします。
- cloudtrail - RDS Custom が DB インスタンスに関する変更イベントを受信することを許可します。
- servicequotas - RDS Custom が DB インスタンスに関するサービスクォータを読み取ることを許可します
- ssm - RDS Custom が DB インスタンスの基盤となる EC2 インスタンスを管理することを許可します。
- rds - RDS Custom が DB インスタンスの RDS リソースを管理することを許可します。
- iam - RDS Custom がインスタンスプロファイルを検証して DB インスタンスの基盤となる EC2 インスタンスにアタッチすることを許可します。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSCustomServiceRolePolicy」を参照してください。

AWS マネージドポリシー: AmazonRDSCustom Instance ProfileRolePolicy

IAM エンティティに AmazonRDSCustomInstanceProfileRolePolicy をアタッチしないでください。このポリシーは、さまざまなオートメーションアクションとデータベース管理タスクを実行

するためのアクセス許可を Amazon RDS Custom DB インスタンスに付与するために使用されるインスタンスプロファイルロールにのみアタッチする必要があります。RDS Custom インスタンスの作成時にインスタンスプロファイルを `custom-iam-instance-profile` パラメータとして渡すと、RDS Custom はこのインスタンスプロファイルを DB インスタンスに関連付けます。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `ssm`、`ssmmessages`、`ec2messages` - RDS Custom が Systems Manager を介して DB インスタンスで通信、自動化の実行、エージェントの管理を実行できるようにします。
- `ec2`、`s3` - RDS Custom が、ポイントインタイム復元機能を提供する DB インスタンスでバックアップオペレーションを実行できるようにします。
- `secretsmanager` - RDS Custom が RDS Custom によって作成された DB インスタンス固有のシークレットを管理できるようにします。
- `cloudwatch`、`logs`- RDS Custom が CloudWatch エージェントを介して DB インスタンスのメトリクスとログを CloudWatch にアップロードできるようにします。
- `events`、`sqs`- RDS Custom が DB インスタンスに関するステータス情報を送受信できるようにします。
- `kms` - RDS Custom がインスタンス固有の KMS キーを使用して、RDS Custom が管理するシークレットと S3 オブジェクトの暗号化を実行できるようにします。

JSON ポリシードキュメントなど、このポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「AmazonRDSCustomInstanceProfileRolePolicy」を参照してください。

AWS マネージドポリシー: AmazonRDSPreviewServiceRolePolicy

IAM エンティティに `AmazonRDSPreviewServiceRolePolicy` をアタッチしないでください。このポリシーは、Amazon RDS が RDS DB リソースに代わって AWS のサービスを呼び出すことを許可するサービスにリンクされたロールに関連付けられています。詳細については、「Amazon RDS Preview のサービスリンクロール」を参照してください。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `ec2` – プリンシパルがアベイラビリティーゾーンとネットワークリソースを記述することを許可します。

- `secretsmanager` — プリンシパルが からシークレットの値を取得するのを許可します。AWS Secrets Manager
- `cloudwatch`、`logs` – Amazon RDS が CloudWatch エージェントを介して DB インスタンスのメトリクスとログを CloudWatch にアップロードできるようにします。

このポリシーの詳細 (JSON ポリシードキュメントを含む) については、「AWS Managed Policy Reference Guide」の「AmazonRDSPreviewServiceRolePolicy」を参照してください。

AWS マネージドポリシー: AmazonRDSBetaServiceRolePolicy

IAM エンティティに `AmazonRDSBetaServiceRolePolicy` をアタッチしないでください。このポリシーは、Amazon RDS が RDS DB リソースに代わって AWS のサービスを呼び出すことを許可するサービスにリンクされたロールに関連付けられています。詳細については、「Amazon RDS Beta のサービスにリンクされたロール許可」を参照してください。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `ec2` – Amazon RDS が、ポイントインタイム復元機能を提供する DB インスタンスでバックアップ操作を実行することを許可します。
- `secretsmanager` – Amazon RDS が Amazon RDS によって作成された DB インスタンス固有のシークレットを管理できるようにします。
- `cloudwatch`、`logs` – Amazon RDS が CloudWatch エージェントを介して DB インスタンスのメトリクスとログを CloudWatch にアップロードできるようにします。

JSON ポリシードキュメントを含むこのポリシーの詳細については、「AWS マネージドポリシーリファレンスガイド」の「AmazonRDSBetaServiceRolePolicy」を参照してください。

Amazon RDS の AWS 管理ポリシーに関する更新

Amazon RDS の AWS マネージドポリシーに対する更新の詳細について、このサービスがこれらの変更の追跡を開始した以降のものを示します。このページへの変更に関する自動アラートを受け取るには、Amazon RDS のドキュメント履歴ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
AWS マネージドポリシー: AmazonRDSPreviewServiceRolePolicy – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSPreview サービスにリンクされたロールの AmazonRDSPreviewServiceRolePolicy から sns:Publish 許可を削除しました。詳細については、「AWS マネージドポリシー: AmazonRDSPreviewServiceRolePolicy」を参照してください。	2024 年 8 月 7 日
AWS マネージドポリシー: AmazonRDSBetaServiceRolePolicy – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSBeta サービスにリンクされたロールの AmazonRDSBetaServiceRolePolicy から sns:Publish 許可を削除しました。詳細については、「AWS マネージドポリシー: AmazonRDSBetaServiceRolePolicy」を参照してください。	2024 年 8 月 7 日
Amazon RDS Custom のサービスにリンクされたロール許可 – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDS	2024 年 7 月 18 日

変更	説明	日付
	`CustomServiceRolePolicy` に新しいアクセス許可を追加しました。アクセス許可により、RDS Custom は別の AWS リージョンの Amazon RDS サービスと通信し、EC2 イメージをコピーできます。詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	
AWS マネージドポリシー: AmazonRDSServiceRolePolicy – 既存のポリシーの更新	Amazon RDS は、`AWSServiceRoleForRDS` サービスにリンクされたロールの `AmazonRDSServiceRolePolicy` から `sns:Publish` 許可を削除しました。詳細については、「AWS マネージドポリシー: AmazonRDSServiceRolePolicy」を参照してください。	2024 年 7 月 2 日

変更	説明	日付
Amazon RDS Custom のサービスにリンクされたロール許可 – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。この新しいアクセス許可により、RDS Custom はサービスロールをインスタンスプロファイルとして RDS Custom インスタンスに関連付けることができます。詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2024 年 4 月 19 日

変更	説明	日付
AWSAmazon RDS の マネージドポリシー – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスリンクロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加し、RDS Custom for SQL Server で基盤となるデータベースのホストインスタンスタイプを変更できるようにしました。また、RDS は、データベースホストに関するインスタンスタイプ情報を取得する ec2:DescribeInstanceTypes アクセス許可も追加しました。詳細については、「AWSAmazon RDS のマネージドポリシー」を参照してください。	2024 年 4 月 8 日
AWSAmazon RDS の マネージドポリシー - 新しいポリシー	Amazon RDS は、RDS Custom が EC2 インスタンスプロファイルを介して自動化アクションとデータベース管理タスクを実行できるように AmazonRDSCustom InstanceProfileRolePolicy という名前の新しいマネージドポリシーを追加しました。詳細については、「AWSAmazon RDS のマネージドポリシー」を参照してください。	2024 年 2 月 27 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDS サービスリンクロールの AmazonRDSServiceRolePolicy に新しいステートメント ID を追加しました。 詳細については、「Amazon RDS のサービスにリンクされたロールのアクセス許可」を参照してください。	2024 年 1 月 19 日
AWSAmazon RDS の マネージドポリシー - 既存のポリシーの更新	AmazonRDSPerformanceInsightsReadOnly および AmazonRDSPerformanceInsightsFullAccess マネージドポリシーには、Sid (ステートメント ID) がポリシーステートメントの識別子として含まれます。 詳細については、「AWS マネージドポリシー: AmazonRDSPerformanceInsightsReadOnly」および「AWS 管理ポリシー: AmazonRDSPerformanceInsightsFullAccess」を参照してください。	2023 年 10 月 23 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。これらの新しいアクセス許可により、RDS Custom for Oracle は EventBridge マネージドルールを作成、変更、削除できるようになります。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 9 月 20 日
AWSAmazon RDS の マネージドポリシー – 既存のポリシーの更新	Amazon ECR では、新しいアクセス許可が AmazonRDSFullAccess 管理ポリシーに追加されました。これらのアクセス許可により、一定期間のパフォーマンス分析レポートを生成、表示、削除できます。 Performance Insights のアクセスポリシーの設定の詳細については、「Performance Insights 用のアクセスポリシーの設定」を参照してください。	2023 年 8 月 17 日

変更	説明	日付
AWSAmazon RDS の マネージドポリシー - 新しいポリシーと、既存のポリシーの更新	Amazon RDS では、AmazonRDSPerformanceInsightsReadOnly 管理ポリシーと AmazonRDSPerformanceInsightsFullAccess という名前の新しい管理ポリシーに新しいアクセス許可が追加されました。これらのアクセス許可により、一定期間の Performance Insights を分析したり、分析結果を推奨事項と共に表示したり、レポートを削除したりできます。 Performance Insights のアクセスポリシーの設定の詳細については、「Performance Insights 用のアクセスポリシーの設定」を参照してください。	2023 年 8 月 16 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。これらの新しいアクセス許可により、RDS Custom for Oracle は DB スナップショットを使用できるようになります。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 6 月 23 日
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。これらの新しいアクセス許可により、RDS Custom for Oracle は DB スナップショットを使用できるようになります。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 6 月 23 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。これらの新しいアクセス許可により、RDS Custom はネットワークインターフェイスを作成できるようになります。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 5 月 30 日
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。これらの新しいアクセス許可により、RDS Custom は Amazon EBS を呼び出してストレージクォータを確認できます。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 4 月 18 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS Custom は、Amazon SQS との統合のために AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。RDS Custom では、顧客アカウントで SQS キューを作成および管理するために Amazon SQS との統合が必要です。SQS キュー名は do-not-delete-rds-custom-[identifier] という形式に従い、Amazon RDS Custom のタグが付けられています。ec2:CreateSnapshot のアクセス許可も追加され、RDS Custom がインスタンスにアタッチされたボリュームのバックアップを作成できるようになりました。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2023 年 4 月 6 日

変更	説明	日付
AWSAmazon RDS の マネージドポリシー – 既存ポリシーへの更新	Amazon RDS は、AmazonRDSFullAccess と AmazonRDSReadOnlyAccess に新しい Amazon CloudWatch 名前空間 ListMetrics を追加しました。 この名前空間は、Amazon RDS が特定のリソース使用メトリクスを一覧表示するために必要です。 詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch リソースへの許可の管理の概要」を参照してください。	2023 年 4 月 4 日

変更	説明	日付
AWSAmazon RDS の マネージドポリシー – 既存ポリシーへの更新	Amazon RDS は AmazonRDSFullAccess および AmazonRDSReadOnlyAccess マネージドポリシーに新しいアクセス許可を追加して、RDS コンソールで Amazon DevOps Guru の検出結果を表示できるようにしました。 このアクセス許可は、DevOps Guru の検出結果を表示できるようにするために必要です。 詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 3 月 30 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWS Secrets Manager との統合のために AWSServiceRoleForRDS サービスにリンクされたロールの AmazonRDSServiceRolePolicy に新しいアクセス許可を追加しました。RDS では、Secrets Manager でマスターユーザーのパスワードを管理するために、Secrets Manager との統合が必要です。シークレットでは予約された命名規則を使用しており、顧客からの更新を制限します。 詳細については、「Amazon RDS および AWS Secrets Manager によるパスワード管理」を参照してください。	2022 年 12 月 22 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDSCustom サービスにリンクされたロールの AmazonRDSCustomServiceRolePolicy に新しいアクセス許可を追加しました。RDS Custom は DB クラスターをサポートします。ポリシー内のこれらの新しいアクセス許可により、RDS Custom は DB クラスターに代わって AWS のサービス を呼び出すことができます。 詳細については、「Amazon RDS Custom のサービスにリンクされたロール許可」を参照してください。	2022 年 11 月 9 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、AWSServiceRoleForRDS との統合のために AWS Secrets Manager サービスにリンクされたロールに新しいアクセス権限を追加しました。 SQL Server Reporting Services (SSRS) の E メールが RDS で機能するには Secrets Manager との統合が必要です。SSRS E メールは顧客に代わってシークレットを作成します。シークレットでは予約された命名規則を使用しており、顧客からの更新を制限します。 詳しくは、「レポートサービスの E メールを使用したレポートの送信」を参照してください。	2022 年 8 月 26 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、`PutMetricData` の `AmazonRDSPreviewServiceRolePolicy` に新しい Amazon CloudWatch 名前空間を追加しました。 この名前空間は、Amazon RDS がリソース使用メトリクスを公開するために必要です。 詳細については、Amazon CloudWatch ユーザーガイドの「条件キーを使用した CloudWatch 名前空間へのアクセスの制限」を参照してください。	2022 年 6 月 7 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、PutMetricData の AmazonRDSBetaServiceRolePolicy に新しい Amazon CloudWatch 名前空間を追加しました。 この名前空間は、Amazon RDS がリソース使用メトリクスを公開するために必要です。 詳細については、Amazon CloudWatch ユーザーガイドの「条件キーを使用した CloudWatch 名前空間へのアクセスの制限」を参照してください。	2022 年 6 月 7 日
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、PutMetricData の AWSServiceRoleForRDS に新しい Amazon CloudWatch 名前空間を追加しました。 この名前空間は、Amazon RDS がリソース使用メトリクスを公開するために必要です。 詳細については、Amazon CloudWatch ユーザーガイドの「条件キーを使用した CloudWatch 名前空間へのアクセスの制限」を参照してください。	2022 年 4 月 22 日

変更	説明	日付
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、お客様が所有する IP プールおよびローカルゲートウェイルートテーブル (LGW-RTB) の許可を管理するための新しい許可を AWSServiceRoleForRDS サービスリンクロールに追加しました。 これらの許可は、Outposts の RDS が Outposts のローカルネットワークでマルチ AZ レプリケーションを実行するために必要です。 詳細については、「AWS Outposts 上での Amazon RDS のマルチ AZ 配置の使用」を参照してください。	2022 年 4 月 19 日
アイデンティティベースポリシー – 既存ポリシーへの更新	Amazon RDS は、LGW-RTB に対する許可を記述するための新しい許可を AmazonRDS FullAccess マネージドポリシーに追加しました。 これらの許可は、Outposts の RDS が Outposts のローカルネットワークでマルチ AZ レプリケーションを実行するために必要です。 詳細については、「AWS Outposts 上での Amazon RDS のマルチ AZ 配置の使用」を参照してください。	2022 年 4 月 19 日

変更	説明	日付
AWSAmazon RDS の マネージドポリシー - 新しいポリシー	Amazon RDS では、Amazon RDS が DB インスタンスに代わって `AmazonRDSPerformanceInsightsReadOnly` サービスを呼び出せるように、AWS という名前の新しい管理ポリシーが追加されました。 Performance Insights のアクセスポリシーの設定の詳細については、「Performance Insights 用のアクセスポリシーの設定」を参照してください。	2022 年 3 月 10 日
Amazon RDS のサービスにリンクされたロールのアクセス許可 – 既存ポリシーへの更新	Amazon RDS は、`PutMetricData` の `AWSServiceRoleForRDS` に新しい Amazon CloudWatch 名前空間を追加しました。 これらの名前空間は、Amazon DocumentDB (MongoDB と互換) と Amazon Neptune が CloudWatch メトリクスを公開するために必要です。 詳細については、Amazon CloudWatch ユーザーガイドの「条件キーを使用した CloudWatch 名前空間へのアクセスの制限」を参照してください。	2022 年 3 月 4 日

変更	説明	日付
Amazon RDS Custom のサービスにリンクされたロール許可 - 新しいポリシー	Amazon RDS は、AWSServiceRoleForRDSCustom という名前の新しいサービスにリンクされたロールを追加して、RDS カスタムが DB インスタンスの代わりに AWS のサービス を呼び出せるようにしました。	2021 年 10 月 26 日
Amazon RDS が変更の追跡を開始しました。	Amazon RDS が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 10 月 26 日

サービス間での混乱した代理問題の防止

「混乱した代理」問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。AWS では、サービス間でのなりすましによって、混乱した代理問題が発生する場合があります。

サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐために AWS では、お客様のすべてのサービスのデータを保護するのに役立つツールを提供しています。これらのツールでは、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルを使用します。詳細については、IAM ユーザーガイド の 混乱した代理問題 を参照してください。

特定のリソースへのアクセスについて、Amazon RDS が別のサービスに付与する許可を制限する場合は、リソースポリシー内で `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用することをお勧めします。

例えば、Amazon S3 バケットの Amazon リソースネーム (ARN) を使用する場合など、`aws:SourceArn` 値にアカウント ID が含まれていないことがあります。このような場合は、前出のグローバル条件コンテキストキーの両方を使用して、パーミッションを制限する必要があります。場合によっては、両方のグローバル条件コンテキストキーと、アカウント ID を含む `aws:SourceArn` 値を併用します。これらを同じポリシーステートメントで使用する場合は、`aws:SourceAccount` の値には、`aws:SourceArn` 内のアカウントと同じアカウント ID を使用します。クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、`aws:SourceArn` を使用します。クロスサービスによる使用のために、AWS アカウント内の任意のリソースを関連づけたい場合は、`aws:SourceAccount` を使用します。

`aws:SourceArn` の値には、Amazon RDS リソースタイプの ARN を指定する必要があります。詳細については、「Amazon RDS の Amazon リソースネーム (ARN)」を参照してください。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して `aws:SourceArn` グローバル条件コンテキストキーを使用することです。この時、リソースの完全な ARN が分からない場合や、複数のリソースを指定しているという場合があり得ます。このような場合、ARN の未知の部分については、ワイルドカード (*) を指定しながら `aws:SourceArn` グローバルコンテキスト条件キーを使用します。例は `arn:aws:rds:*:123456789012:*` です。

次の例では、Amazon RDS で aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して、「混乱した代理」問題を回避する方法を示します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

グローバル条件コンテキストキーの aws:SourceArn と aws:SourceAccount を使用する他のポリシー例については、以下の各セクションを参照してください。

- Amazon SNS トピックに通知を発行するアクセス許可を付与する
- ネイティブバックアップおよび復元用の IAM ロールの手動作成
- SQL Server DB インスタンスの Windows 認証のセットアップ
- RDS for SQL Server を S3 と統合するための前提条件
- SQL Server Audit の IAM ロールを手動で作成する
- Amazon S3 と RDS for Oracle を統合する IAM アクセス許可の設定
- Amazon S3 バケットへのアクセスを設定する (PostgreSQL のインポート)
- Amazon S3 バケットへのアクセスを設定する (PostgreSQL エクスポート)

MariaDB、MySQL、および PostgreSQL の IAM データベース認証

AWS Identity and Access Management (IAM) データベース認証を使用して、DB インスタンスを認証できます。IAM データベース認証には、MariaDB、MySQL、および PostgreSQL を使用します。この認証方法では、DB インスタンスに接続するときにパスワードを使用する必要はありません。代わりに、認証トークンを使用します。

認証トークンは、Amazon RDS がリクエストに応じて生成する一意の文字列です。認証トークンは、AWS 署名バージョン 4 を使用して生成されます。各トークンには 15 分の有効期間があります。認証は IAM を使用して外部的に管理されるため、ユーザー認証情報をデータベースに保存する必要はありません。引き続きスタンダードのデータベース認証を使用することもできます。トークンは認証にのみ使用され、確立後のセッションには影響しません。

IAM データベース認証には次の利点があります。

- データベースとの間で送受信されるネットワークトラフィックは、Secure Socket Layer (SSL) または Transport Layer Security (TLS) を使用して暗号化されます。Amazon RDS で SSL/TLS を使用する方法については、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。
- IAM を使用して各 DB インスタンスで個別に管理するのではなく、データベースリソースへのアクセスを一元的に管理できます。
- Amazon EC2 で実行するアプリケーションの場合、セキュリティを高めるため、EC2 インスタンスに固有のプロファイル認証情報を使用して、パスワードの代わりにデータベースにアクセスできます。

一般に、アプリケーションコードでユーザー名とパスワードを直接管理したくない場合は、IAM データベース認証の使用を検討してください。

Amazon Web Services (AWS) JDBC ドライバーは IAM データベース認証をサポートしています。詳細については、「Amazon Web Services (AWS) JDBC ドライバー GitHub リポジトリ」の「AWS IAM Authentication Plugin」を参照してください。https://github.com/aws/aws-advanced-jdbc-wrapper

Amazon Web Services (AWS) Python ドライバーは IAM データベース認証をサポートしています。詳細については、「Amazon Web Services (AWS) Python ドライバー GitHub リポジトリ」の「AWS IAM Authentication Plugin」を参照してください。https://github.com/aws/aws-advanced-python-wrapper

DB 認証に IAM を設定するプロセスについては、以下のトピックを参照してください。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成
- IAM 認証を使用した DB インスタンスへの接続

利用可能なリージョンとバージョン

機能の可用性とサポートは、各データベースエンジンの特定のバージョンによって異なります。Amazon RDS と IAM データベース認証を使用したエンジン、バージョン、リージョンの可用性の詳細については、「Amazon RDS での IAM データベース認証でサポートされているリージョンと DB エンジン」を参照してください。

CLI および SDK のサポート

IAM データベース認証は、AWS CLI と以下の各言語固有の AWS SDK について使用できます。

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby

IAM データベース認証の制限

IAM データベース認証を使用する場合、以下の制限が適用されます。

- 現在、IAM データベース認証はすべてのグローバル条件コンテキストキーをサポートしていません。

  グローバル条件コンテキストキーの詳細については、「IAM ユーザーガイド」の「AWS グローバル条件コンテキストキー」を参照してください。

- PostgreSQL の場合、IAM ロール (`rds_iam`) がマスターユーザーに追加される (マスターユーザーである RDS を含む) と、IAM 認証はパスワード認証よりも優先されるため、ユーザーは IAM ユーザーとしてログインする必要があります。
- PostgreSQL の場合、Amazon RDS は IAM 認証方法と Kerberos 認証方法両方の同時有効化をサポートしていません。
- PostgreSQL では、IAM 認証を使用してレプリケーション接続を確立することはできません。
- DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。
- CloudWatch と CloudTrail は IAM 認証のログ記録を行いません。これらのサービスは、IAM ロールにデータベース接続の有効化を許可する `generate-db-auth-token` API コールを追跡しません。
- IAM DB 認証には、データベースインスタンスのコンピューティングリソースが必要です。信頼性の高い接続を実現するには、データベースに 300 ~ 1,000 MiB 以上の追加メモリが必要です。ワークロードに必要なメモリを確認するには、IAM DB 認証を有効にする前と後に、拡張モニタリングプロセスリストの RDS プロセスの RES 列を比較します。「RDS コンソールでの OS メトリクスの表示」を参照してください。

  バースト可能なクラスインスタンスを使用している場合は、バッファやキャッシュなどの他のパラメータで使用されるメモリを同じ量だけ減らして、メモリが不足しないようにします。
- IAM DB 認証は、どのエンジンの RDS on Outposts でもサポートされていません。

IAM データベース認証に関する推奨事項

IAM データベース認証を使用する場合には、以下のことをお勧めします。

- IAM データベース認証トークンのサイズは、IAM タグの数、IAM サービスポリシー、ARN の長さ、その他の IAM やデータベースのプロパティなど、さまざまな要素によって異なります。このトークンの最小サイズは、通常、約 1 KB ですが、それ以上になることもあります。このトークンは IAM 認証を使用するデータベースへの接続文字列のパスワードとして使用されるため、データベースドライバー (ODBC など) やツールが、サイズを理由にこのトークンを制限したり、切り詰めたりしないようにする必要があります。トークンが切り詰められると、データベースと IAM による認証検証は失敗します。
- IAM データベース認証トークンの作成時に一時的な認証情報を使用している場合でも、IAM データベース認証トークンを使用して接続リクエストを行うときには、その一時的な認証情報が引き続き有効なものである必要があります。

サポートされていない AWS グローバル条件コンテキストキー

IAMデータベース認証は AWS グローバル条件コンテキストキーのうち次のサブセットをサポートしていません。

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

条件キーの詳細については、IAM ユーザーガイド の「AWS グローバル条件コンテキストキー」 を参照してください。

IAM データベース認証の有効化と無効化

デフォルトでは、IAM データベース認証は DB インスタンスで無効になります。AWS マネジメントコンソール、AWS CLI、API のいずれかを使用して、IAM データベース認証を有効または無効にすることができます。

次のいずれかのアクションを実行する際に、IAM データベース認証を有効にすることができます。

- IAM データベース認証を有効にして新しい DB インスタンスを作成するには、「Amazon RDS DB インスタンスの作成」を参照してください。
- DB インスタンスを変更して IAM データベース認証を有効にするには、「Amazon RDS DB インスタンスを変更する」を参照してください。
- IAM データベース認証を有効にしてスナップショットから DB インスタンスを復元するには、「DB インスタンスへの復元」を参照してください。
- IAM データベース認証を有効化しながら DB インスタンスを特定の時点に復元するには、「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

コンソール

作成または変更の各ワークフローには、[データベース認証] セクションがあり、IAM データベース認証を有効または無効にすることができます。そのセクションで、[パスワードと IAM データベース認証] を選択して、IAM データベース認証を有効にします。

既存の DB インスタンスに対して IAM データベース認証を有効または無効にするには

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 変更する DB インスタンスを選択します。

> **Note**
>
> DB インスタンスが IAM 認証と互換性があることを確認します。利用可能なリージョンとバージョン の互換性要件を確認する。

4. [Modify] を選択します。
5. [データベース認証] セクションで、[パスワードと IAM データベース認証] を選択して、 を有効にします。IAM 認証を無効にするには、[パスワード認証] または [パスワードと Kerberos 認証] を選択します。
6. CloudWatch Logs への IAM DB 認証ログの発行を有効にすることもできます。[ログのエクスポート] で、[iam-db-auth-error ログ] オプションを選択します。ログを CloudWatch Logs に発行するとストレージが消費され、その分の料金が発生します。不要になった CloudWatch Logs は削除してください。
7. [Continue] を選択します。
8. 変更をすぐに適用するには、[変更のスケジューリング] セクションで [今すぐ] を選択します。
9. [DB インスタンスを変更] を選択します。

AWS CLI

AWS CLI を使用して、IAM 認証で新しい DB インスタンスを作成するには、create-db-instance コマンドを使用します。次の例のように、`--enable-iam-database-authentication` オプションを指定します。

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m3.medium \
    --engine MySQL \
    --allocated-storage 20 \
    --master-username masterawsuser \
    --manage-master-user-password \
    --enable-iam-database-authentication
```

既存の DB インスタンスを更新して、IAM 認証を使用するかどうかを指定するには、AWS CLI コマンド modify-db-instance を使用します。必要に応じて --enable-iam-database-authentication または --no-enable-iam-database-authentication オプションを指定します。

> **Note**
>
> DB インスタンスが IAM 認証と互換性があることを確認します。利用可能なリージョンとバージョン の互換性要件を確認する。

デフォルトでは、Amazon RDS は次のメンテナンスウィンドウ中に変更を実行します。これを上書きし、IAM DB 認証をできるだけ早く有効にする場合は、--apply-immediately パラメータを使用します。

次の例は、既存の DB インスタンスの IAM 認証をすぐに有効にする方法を示しています。

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --apply-immediately \
    --enable-iam-database-authentication
```

DB インスタンスを復元する場合は、次のいずれかの AWS CLI コマンドを使用します。

- restore-db-instance-to-point-in-time
- restore-db-instance-from-db-snapshot

IAM データベース認証設定は、デフォルトで元のスナップショットの設定になります。この設定を変更するには、必要に応じて --enable-iam-database-authentication または --no-enable-iam-database-authentication オプションを設定します。

RDS API

API を使用して、IAM 認証で新しい DB インスタンスを作成するには、API オペレーション CreateDBInstance を使用します。EnableIAMDatabaseAuthentication パラメータを true に設定します。

既存の DB インスタンスを更新して、IAM 認証を持つ、または持たないようにするには、API オペレーション ModifyDBInstance を使用します。EnableIAMDatabaseAuthentication パラメータを true に設定して IAM 認証を有効にするか、false に設定して無効にします。

> **Note**
>
> DB インスタンスが IAM 認証と互換性があることを確認します。利用可能なリージョンとバージョン の互換性要件を確認する。

DB インスタンスを復元する場合は、次のいずれかの API オペレーションを使用します。

- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceToPointInTime

IAM データベース認証設定は、デフォルトで元のスナップショットの設定になります。この設定を変更するには、EnableIAMDatabaseAuthentication パラメータを true に設定して IAM 認証を有効にするか、false に設定して無効にします。

IAM データベースアクセス用の IAM ポリシーの作成と使用

ユーザーまたはロールに DB インスタンスへの接続を許可するには、IAM ポリシーを作成する必要があります。その後、ポリシーをアクセス許可セットまたはロールにアタッチします。

> **Note**
>
> IAM キーポリシーの詳細については、「Amazon RDS での Identity and Access Management」を参照してください。

次のポリシー例では、ユーザーは IAM データベース認証を使用して、DB インスタンスに接続できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:db-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

Important

管理者権限を持つユーザーは、IAM ポリシーで明示的なアクセス許可が設定されていない場合でも、DB インスタンスのにアクセスできます。管理者アクセスを DB インスタンスのに制限するには、最低限のアクセス許可が適切に設定された IAM ロールを作成し、それを管理者に設定します。

Note

`rds-db:` プレフィックスと、`rds:` で始まる他の RDS API オペレーションのプレフィックスを混同しないでください。IAM データベース認証に対してのみ、`rds-db:` プレフィックスと `rds-db:connect` アクションを使用します。これらは、その他のコンテキストでは有効ではありません。

このポリシーには、次の要素を持つ 1 つのステートメントが含まれています。

- `Effect` - DB インスタンスへのアクセスを許可するには、`Allow` を指定します。アクセスを明示的に許可しない場合、デフォルトでアクセスは拒否されます。
- `Action` - DB インスタンスへの接続を許可するには、`rds-db:connect` を指定します。
- `Resource` - 1 つの DB インスタンスで 1 つのデータベースアカウントを示す Amazon リソースネーム (ARN) を指定します。ARN 形式は次のとおりです。

```
arn:aws:rds-db:region:account-id:dbuser:DbiResourceId/db-user-name
```

この形式では、以下のように置き換えます。

- *region* は、DB インスタンスの AWS リージョンです。このポリシー例での AWS リージョンは us-east-2 です。
- *account-id* は DB インスタンスの AWS アカウント番号です。このポリシー例でのアカウント番号は 1234567890 です。ユーザーは DB インスタンスのアカウントと同じアカウントでなければなりません。

  クロスアカウントアクセスを実行するには、DB インスタンスのアカウントに上記のポリシーで IAM ロールを作成し、他のアカウントがそのロールを引き継ぐことを許可します。
- *DbiResourceId* は、DB インスタンスの識別子です。この識別子は AWS リージョンに固有であり、変更されることはありません。このポリシー例での識別子は db-ABCDEFGHIJKL01234 です。

  Amazon RDS 用の DB インスタンスのリソース ID を AWS マネジメントコンソール で検索するには、DB インスタンスを選択して、その詳細を表示します。そして、[Configuration (設定)] タブを選択します。[設定] セクションに [リソース ID] が表示されます。

  または、次のように AWS CLI コマンドを使用して、以下に示されているように、現在の AWS リージョンのすべての DB インスタンスの識別子とリソース ID をリストできます。

```
aws rds describe-db-instances --query "DBInstances[*].
[DBInstanceIdentifier,DbiResourceId]"
```

  Amazon Aurora を使用している場合は、DbiResourceId の代わりに DbClusterResourceId を指定してください。詳細については、Amazon Aurora ユーザーガイドの「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。

> **Note**
>
> RDS Proxy 経由でデータベースに接続する場合は、prx-ABCDEFGHIJKL01234 などのプロキシリソース ID を指定します。RDS Proxy で IAM データベース認証を使用する方法については、「IAM 認証を使用したデータベースへの接続」を参照してください。

- *db-user-name* は、IAM 認証に関連付けるデータベースアカウントの名前です。このポリシー例で、データベースアカウントは db_user です。

多様なアクセスパターンをサポートするため、他の ARN を構築できます。次のポリシーでは、DB インスタンスで 2 つの異なるデータベースアカウントにアクセスできます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
mary_roe"
         ]
      }
   ]
}
```

次のポリシーでは、特定の AWS アカウントと AWS リージョンのすべての DB インスタンスとデータベースアカウントに一致させるために「*」文字を使用します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
```

```
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:*/*"
            ]
        }
    ]
}
```

次のポリシーは、特定の AWS アカウントと AWS リージョンの DB インスタンスすべてに一致します。ただし、jane_doe データベースアカウントを持つ DB インスタンス または DB クラスターにのみにアクセスが許可されます。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

ユーザーまたはロールは、データベースユーザーがアクセスするデータベースにのみアクセスできます。例えば、DB インスタンスに dev という名前のデータベースと、test という名前の別のデータベースがあるとします。データベースユーザー jane_doe が dev のみにアクセスできる場合、jane_doe ユーザーでその DB インスタンスにアクセスできるユーザーまたはロールも、dev にのみアクセスできます。このアクセス制限は、テーブル、ビューなどその他のデータベースオブジェクトにも当てはまります。

管理者は、エンティティに必要な、指定されたリソースに対して特定の API オペレーションを実行するアクセス許可を付与する IAM ポリシーを作成する必要があります。続いて、管理者は、それら

のアクセス許可を必要とするアクセス許可セットまたはロールに、そのポリシーをアタッチします。ポリシーの例については、「Amazon RDS のアイデンティティベースのポリシーの例」を参照してください。

IAM ポリシーをアクセス許可セットまたはロールにアタッチする

データベース認証を許可する IAM ポリシーを作成した後、そのポリシーをアクセス許可セットまたはロールにアタッチする必要があります。このトピックに関するチュートリアルについては、IAM ユーザーガイドの「はじめてのカスタマー管理ポリシーの作成とアタッチ」を参照してください。

チュートリアルを進める際に、このセクションに記載されているいずれかのポリシー例をスタート点として使用し、ニーズに合わせて調整することができます。チュートリアルを完了すると、`rds-db:connect` アクションを利用できる、ポリシーがアタッチされたアクセス許可セットが作成されます。

> **Note**
>
> 複数のアクセス許可セットまたはロールを同じデータベースユーザーアカウントにマップできます。例えば、IAM ポリシーで以下のリソース ARN を指定したとします。
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> ポリシーを Jane、Bob、Diego にアタッチした場合、これらの各ユーザーは、`jane_doe` データベースアカウントを使用して、指定された DB インスタンスに接続できます。

IAM 認証を使用したデータベースアカウントの作成

IAM データベース認証では、作成するユーザーアカウントにデータベースのパスワードを割り当てる必要はありません。データベースアカウントにマッピングされているユーザーを削除した場合は、`DROP USER` ステートメントでデータベースアカウントも削除する必要があります。

> **Note**
>
> IAM 認証に使用されるユーザー名は、データベース内のユーザー名の大文字および小文字と一致する必要があります。

トピック

- MariaDB および MySQL での IAM 認証の使用
- PostgreSQL での IAM 認証の使用

MariaDB および MySQL での IAM 認証の使用

MariaDB および MySQL では、認証は、AWSAuthenticationPlugin (IAM とシームレスに連携してユーザーを認証する AWS 提供のプラグイン) によって処理されます。DB インスタンスに、マスターユーザーまたはユーザーを作成して権限を付与できる別のユーザーとして接続します。接続後、次の例に示すように、CREATE USER ステートメントを発行します。

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

IDENTIFIED WITH 句により、MariaDB および MySQL は AWSAuthenticationPlugin を使用して、データベースアカウント (jane_doe) を認証できます。AS 'RDS' 句は、認証方式を参照します。指定したデータベースユーザー名は、IAM データベースアクセスの IAM ポリシー内のリソースと同じであることを確認します。詳細については、「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。

> **Note**
>
> 次のメッセージが表示された場合、AWS が提供するプラグインが、現在の DB インスタンスに使用できないことを意味します。
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> このエラーをトラブルシューティングするには、サポートされている設定を使用していること、および DB インスタンスで IAM データベース認証を有効にしていることを確認します。詳細については、「利用可能なリージョンとバージョン」および「IAM データベース認証の有効化と無効化」を参照してください。

AWSAuthenticationPlugin を使用してアカウントを作成したら、他のデータベースのアカウントと同様に管理します。例えば、GRANT および REVOKE ステートメントでアカウント特権を変更したり、ALTER USER ステートメントでさまざまなアカウント属性を変更したりできます。

IAM を使用する場合、データベースネットワークトラフィックは SSL/TLS を使用して暗号化されます。SSL 接続を許可するには、以下のコマンドでユーザーアカウントを変更します。

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

PostgreSQL での IAM 認証の使用

PostgreSQL で IAM 認証を使用するには、マスターユーザーまたはユーザーを作成して権限を付与できる別のユーザーとして DB インスタンスに接続します。接続後、データベースユーザーを作成して、次の例に示すように、ユーザーに rds_iam ロールを付与します。

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

指定したデータベースユーザー名は、IAM データベースアクセスの IAM ポリシー内のリソースと同じであることを確認します。詳細については、「IAM データベースアクセス用の IAM ポリシーの作成と使用」を参照してください。IAM 認証を使用するには、rds_iam ロールを付与する必要があります。ロールのネストされたメンバーシップまたは間接的な付与を使用することもできます。

IAM 認証を使用した DB インスタンスへの接続

IAM データベース認証では、DB インスタンスに接続するときに認証トークンを使用します。認証トークンは、パスワードの代わりに使用する文字列です。認証トークンを生成した後、期限切れになるまで 15 分間有効です。期限切れのトークンを使用して接続を試みると、接続リクエストは拒否されます。

すべての認証トークンは、AWS 署名バージョン 4 を使用した有効な署名が添付されている必要があります (詳細については、AWS 全般のリファレンスの「Signature Version 4 の署名プロセス」を参照してください)。AWS CLI や AWS など、AWS SDK for Java とAWS SDK for Python (Boto3) SDK は、作成した各トークンに自動的に署名できます。

別の AWS のサービス (AWS Lambda など) から Amazon RDS に接続するときに、認証トークンを使用できます。トークンを使用することで、コードにパスワードを含めないで済みます。あるいは、AWS SDK を使用して、認証トークンをプログラムで作成して、プログラムで署名することもできます。

IAM 認証トークンに署名した後、Amazon RDS DB インスタンス に接続できます。以下では、コマンドラインツールまたは AWS や AWS SDK for Java などの AWS SDK for Python (Boto3) SDK を使用して、これを行う方法を示しています。

詳細については、以下のブログ投稿を参照してください。

- IAM 認証を使用して SQL Workbench/J により Aurora MySQL または Amazon RDS for MySQL に接続する
- pgAdmin Amazon Aurora PostgreSQL または Amazon RDS for PostgreSQL と接続するための IAM 認証の使用

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

トピック

- IAM 認証と AWS ドライバーを使用した DB インスタンスへの接続
- コマンドラインから IAM 認証を使用して、DB インスタンスに接続する: AWS CLI および mysql クライアント
- コマンドラインから IAM 認証を使用して DB インスタンスに接続する: AWS CLI および psql クライアント
- IAM 認証および AWS SDK for .NET を使用した DB インスタンスへの接続
- IAM 認証および AWS SDK for Go を使用した DB インスタンスへの接続
- IAM 認証および AWS SDK for Java を使用した DB インスタンスへの接続
- IAM 認証および AWS SDK for Python (Boto3) を使用した DB インスタンスへの接続

IAM 認証と AWS ドライバーを使用した DB インスタンスへの接続

AWS のドライバースイートは、スイッチオーバーとフェイルオーバーの時間の短縮、AWS Secrets Manager、AWS Identity and Access Management (IAM)、フェデレーティッド ID での認証をサポートするように設計されています。AWS ドライバーは、DB インスタンスステータスをモニタリングし、インスタンストポロジを認識して新しいライターを決定することを前提としています。このアプローチにより、スイッチオーバーとフェイルオーバーの時間が 1 桁秒に短縮されます (オープンソースドライバーの場合は数十秒)。

AWS ドライバーの詳細については、使用している RDS for MariaDB、RDS for MySQL、または RDS for PostgreSQL DB インスタンスに対応する言語ドライバーを参照してください。

> **Note**
>
> RDS for MariaDB でサポートされている機能は、AWS Secrets Manager、AWS Identity and Access Management (IAM)、およびフェデレーティッド ID による認証のみです。

コマンドラインから IAM 認証を使用して、DB インスタンスに接続する: AWS CLI および mysql クライアント

以下に示すように、AWS CLI および `mysql` コマンドラインツールを使用して、コマンドラインから Amazon RDS DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

> **Note**
>
> IAM 認証を使用して SQLWorkbench/J を使用してデータベースに接続する方法については、ブログ記事「IAM 認証を使用して SQL Workbench/J で Aurora MySQL または Amazon RDS for MySQL に接続する」を参照してください。

トピック

- IAM 認証トークンの生成
- DB インスタンスへの接続

IAM 認証トークンの生成

次の例では、AWS CLI を使用して署名された認証トークンを取得する方法を示します。

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
```

```
    --port 3306 \
    --region us-west-2 \
    --username jane_doe
```

この例で、パラメータは次のとおりです。

- `--hostname` - アクセス先の DB インスタンスのホスト名。
- `--port` - DB インスタンスへの接続に使用するポート番号
- `--region` - DB インスタンスが実行中の AWS リージョン
- `--username` - アクセス先のデータベースアカウント

トークンの初期の複数の文字は次のようになります。

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

DB インスタンスへの接続

接続の全般的な形式を次に示します。

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

パラメータは次のとおりです。

- `--host` - アクセス先の DB インスタンスのホスト名。
- `--port` - DB インスタンスへの接続に使用するポート番号
- `--ssl-ca` - 公開キーを含む SSL 証明書ファイルへのフルパス

  MariaDB での SSL/TLS サポートについては、「Amazon RDS 上の MariaDB DB インスタンスの SSL/TLS サポート」を参照してください。

MySQL での SSL/TLS サポートについては、「Amazon RDS 上の MySQL DB インスタンスの SSL/TLS サポート」を参照してください。

SSL 証明書をダウンロードするには SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照ください。

- `--enable-cleartext-plugin` - この接続で AWSAuthenticationPlugin を使用する必要があることを示す値

  MariaDB クライアントを使用している場合、`--enable-cleartext-plugin` オプションは必須ではありません。
- `--user` - アクセス先のデータベースアカウント
- `--password` - 署名済みの IAM 認証トークン

認証トークンは数百の文字で構成されます。これは、コマンドラインでは手に負えなくなる可能性があります。この問題を回避する 1 つの方法は、環境可変にトークンを保存し、接続時にその可変を使用することです。次の例は、この回避策を実行する 1 つの方法を示しています。この例では、*/sample_dir/* が公開キーを含む SSL 証明書ファイルへのフルパスです。

```
RDSHOST="mysqldb.123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

AWSAuthenticationPlugin を使って接続した場合、接続は SSL を使用して保護されます。これを確認するには、mysql> コマンドプロンプトで以下を入力します。

```
show status like 'Ssl%';
```

出力の次の行に詳細情報が示されます。

```
+---------------+-------------+
| Variable_name | Value
                                                                |
```

```
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                  |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                  |
| ...           | ...
+-----------------------------+
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

コマンドラインから IAM 認証を使用して DB インスタンスに接続する: AWS CLI および psql クライアント

以下に示すように、AWS CLI および psql コマンドラインツールを使用して、コマンドラインから Amazon RDS for PostgreSQL DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

> **Note**
>
> pgAdminを使用してIAM認証でデータベースに接続する方法については、ブログ記事IAM認証を使用してpgAdmin Amazon Aurora PostgreSQLまたはAmazon RDS for PostgreSQLで接続するを参照してください。

トピック

- IAM 認証トークンの生成
- Amazon RDS PostgreSQL インスタンスへの接続

IAM 認証トークンの生成

認証トークンは数百の文字で構成されるため、コマンドラインでは手に負えなくなる可能性があります。この問題を回避する 1 つの方法は、環境可変にトークンを保存し、接続時にその可変を使用することです。次の例では、AWS CLI コマンドを使用して署名された認証トークンを取得するために generate-db-auth-token を使用し、PGPASSWORD 環境可変に格納する方法を示しています。

```
export RDSHOST="rdspostgres.123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

例では、generate-db-auth-token コマンドへのパラメータは次のとおりです。

- --hostname - アクセス先の DB インスタンス のホスト名
- --port - DB インスタンスへの接続に使用するポート番号
- --region - DB インスタンスが実行中の AWS リージョン
- --username - アクセス先のデータベースアカウント

生成されたトークンの初期の複数の文字は次のようになります。

```
rdspostgres.123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

Amazon RDS PostgreSQL インスタンスへの接続

psql を使用して接続する全般的な形式を次に示します。

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

パラメータは次のとおりです。

- host - アクセス先の DB インスタンス のホスト名
- port - DB インスタンスへの接続に使用するポート番号
- sslmode - 使用する SSL モード

  sslmode=verify-full を使用すると、SSL 接続で DB インスタンスのエンドポイントを SSL 証明書のエンドポイントと照合します。
- sslrootcert - 公開キーを含む SSL 証明書ファイルへのフルパス

  詳細については、「PostgreSQL DB インスタンスで SSL を使用する」を参照してください。

  SSL 証明書をダウンロードするには SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照ください。
- dbname - アクセス先のデータベース
- user - アクセス先のデータベースアカウント
- password - 署名済みの IAM 認証トークン

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

次の例は、psql を使用して接続する方法を示しています。この例の psql では、環境変数 RDSHOST をホスト用に、また、環境変数 PGPASSWORD を生成されたトークン用に使用しています。また、*/sample_dir/* は公開キーを含む SSL 証明書ファイルへの完全なパスを示します。

```
export RDSHOST="rdspostgres.123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM 認証および AWS SDK for .NET を使用した DB インスタンスへの接続

次に説明するように、AWS SDK for .NET を使用して、RDS for MariaDB、MySQL、または PostgreSQL DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

例

以下のコード例で、認証トークンを生成し、それを使用して DB インスタンスに接続する方法を示します。

このコードサンプルを実行するには、AWS SDK for .NET サイトにある AWS が必要です。`AWSSDK.CORE` および `AWSSDK.RDS` パッケージが必要です。DB インスタンスに接続するには、MariaDB または MySQL 用の MySqlConnector や PostgreSQL 用の Npgsql など、DB エンジン用の .NET データベースコネクタを使用します。

このコードで MariaDB インスタンスまたは MySQL DB インスタンスに接続します。必要に応じて以下の可変の値を変更します。

- `server` - アクセス先の DB インスタンスのエンドポイント
- `user` - アクセス先のデータベースアカウント
- `database` - アクセス先のデータベース
- `port` - DB インスタンスへの接続に使用するポート番号
- `SslMode` - 使用する SSL モード

  `SslMode=Required` を使用すると、SSL 接続で DB インスタンスのエンドポイントを SSL 証明書のエンドポイントと照合します。
- `SslCa` - Amazon RDS のSSL 証明書へのフルパス

  証明書をダウンロードするには、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqldb.123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new MySqlConnection($"server=mysqldb.123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);

      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
```

```
  }
}
```

このコードで PostgreSQL DB インスタンスに接続します。

必要に応じて以下の可変の値を変更します。

- `Server` - アクセス先の DB インスタンスのエンドポイント
- `User ID` - アクセス先のデータベースアカウント
- `Database` - アクセス先のデータベース
- `Port` - DB インスタンスへの接続に使用するポート番号
- `SSL Mode` - 使用する SSL モード

  `SSL Mode=Required` を使用すると、SSL 接続で DB インスタンスのエンドポイントを SSL 証明書のエンドポイントと照合します。
- `Root Certificate` - Amazon RDS のSSL 証明書へのフルパス

  証明書をダウンロードするには、「SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化」を参照してください。

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

```
using System;
using Npgsql;
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmydb.123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
```

```
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmydb.123456789012.us-east-1.rds.amazonaws.com;User
 Id=jane_doe;Password={pwd};Database=mydb;SSL Mode=Require;Root
 Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM 認証および AWS SDK for Go を使用した DB インスタンスへの接続

次に説明するように、AWS SDK for Go を使用して、RDS for MariaDB、MySQL、または PostgreSQL DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

例

これらのコードサンプルを実行するには、AWS SDK for Go サイトにある AWS が必要です。

必要に応じて以下の可変の値を変更します。

- dbName - アクセス先のデータベース
- dbUser - アクセス先のデータベースアカウント
- dbHost - アクセス先の DB インスタンスのエンドポイント

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

- dbPort - DB インスタンスへの接続に使用するポート番号
- region - DB インスタンスが実行中の AWS リージョン

さらに、サンプルコード内のインポートされるライブラリがシステムに存在することを確認してください。

> **Important**
>
> このセクションの例では、次のコードを使用して、ローカル環境からデータベースにアクセスする認証情報を提供します。
> `creds := credentials.NewEnvCredentials()`
> Amazon EC2 や Amazon ECS などの AWS のサービスからデータベースにアクセスする場合は、コードを次のコードに置き換えることができます。
> `sess := session.Must(session.NewSession())`
> `creds := sess.Config.Credentials`
> この変更を行う場合は、次のインポートを追加してください。
> `"github.com/aws/aws-sdk-go/aws/session"`

トピック

- IAM 認証と AWS SDK for Go V2 を使用した接続
- IAM 認証と AWS SDK for Go V1 を使用した接続。

IAM 認証と AWS SDK for Go V2 を使用した接続

IAM 認証と AWS SDK for Go V2 を使用して DB インスタンスに接続できます

以下のコード例で、認証トークンを生成し、それを使用して DB インスタンスに接続する方法を示します。

このコードで MariaDB インスタンスまたは MySQL DB インスタンスに接続します。

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqldb.123456789012.us-east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )
```

```
    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

このコードで PostgreSQL DB インスタンスに接続します。

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
```

```
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM 認証と AWS SDK for Go V1 を使用した接続。

IAM 認証と AWS SDK for Go V1 を使用して DB インスタンスに接続できます

以下のコード例で、認証トークンを生成し、それを使用して DB インスタンスに接続する方法を示します。

このコードで MariaDB インスタンスまたは MySQL DB インスタンスに接続します。

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
```

```
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqldb.123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

このコードで PostgreSQL DB インスタンスに接続します。

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
```

```
    dbUser := "jane_doe"
    dbHost := "postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM 認証および AWS SDK for Java を使用した DB インスタンスへの接続

次に説明するように、AWS SDK for Java を使用して、RDS for MariaDB、MySQL、または PostgreSQL DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

- AWS SDK for Java をセットアップする

SDK for Java 2.x の使用方法の例については、「SDK for Java 2.x を使用した Amazon RDS の例」を参照してください。AWS Advanced JDBC Wrapper を使用することもできます。「AWS Advanced JDBC Wrapper ドキュメント」を参照してください。

トピック

- IAM 認証トークンの生成
- IAM 認証トークンを手動で構築する
- DB インスタンスへの接続

IAM 認証トークンの生成

AWS SDK for Java を使用してプログラムを作成する場合、RdsIamAuthTokenGenerator クラスを使用して署名付き認証トークンを取得できます。このクラスを使用するには、AWS 認証情報を提供する必要があります。これを行うには、DefaultAWSCredentialsProviderChain クラスのインスタンスを作成します。DefaultAWSCredentialsProviderChain は、デフォルトの認証情報プロバイダチェーンで見つける初期の AWS のアクセスキーとシークレットキーを使用します。AWS アクセスキーの詳細については、「ユーザーのアクセスキーの管理」を参照してください。

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

RdsIamAuthTokenGenerator のインスタンスを作成した後、getAuthToken メソッドを呼び出して、署名済みトークンを取得できます。AWS リージョン、ホスト名、ポート番号、およびユーザー名を指定します。次のコード例はこれを行う方法を示しています。

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {
```

```
    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

IAM 認証トークンを手動で構築する

Java では、認証トークンを生成するための最も簡単な方法は、RdsIamAuthTokenGenerator を使用することです。このクラスは、認証トークンを作成した後、AWS 署名バージョン 4 を使用してサインインします。詳細については、AWS 全般のリファレンスの「Signature Version 4 の署名プロセス」を参照してください。

ただし、次のコード例に示すように、認証トークンを手動で構築して署名できます。

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
```

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
 SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
        DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
     String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
     String awsSecretKey = creds.getCredentials().getAWSSecretKey();
        String expiryMinutes = "900";
```

```
        System.out.println("Step 1:  Create a canonical request:");
        String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
        System.out.println(canonicalString);
        System.out.println();

        System.out.println("Step 2:  Create a string to sign:");
        String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
        System.out.println(stringToSign);
        System.out.println();

        System.out.println("Step 3:  Calculate the signature:");
        String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
        System.out.println(signature);
        System.out.println();

        System.out.println("Step 4:  Add the signing info to the request");

        System.out.println(appendSignature(signature));
        System.out.println();

    }

    //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
    public static String createCanonicalString(String user, String accessKey, String
date, String dateTime, String region, String expiryPeriod, String hostName, String
port) throws Exception {
        canonicalQueryParameters.put("Action", action);
        canonicalQueryParameters.put("DBUser", user);
        canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
        canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
        canonicalQueryParameters.put("X-Amz-Date", dateTime);
        canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
        canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
```

```
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
 currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                              SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
```

```
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String
serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                      SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
```

```
        }
    }
}
```

DB インスタンスへの接続

次のコード例では、認証トークンを生成し、それを使用して MariaDB または MySQL を実行しているインスタンスに接続する方法を示しています。

このコードサンプルを実行するには、AWS SDK for Java サイトにある AWS が必要です。また、以下が必要になります。

- MySQL Connector/J。 このコードの例は `mysql-connector-java-5.1.33-bin.jar` でテストされています。
- AWS リージョンに固有の、Amazon RDS の中間証明書。(詳細については、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください)。クラスローダーは、実行時にこの Java コード例と同じディレクトリで証明書を探し、クラスローダーがその証明書を見つけられるようにします。
- 必要に応じて以下の可変の値を変更します。
  - `RDS_INSTANCE_HOSTNAME` - アクセス先の DB インスタンスのホスト名。
  - `RDS_INSTANCE_PORT` - PostgreSQL DB インスタンスへの接続に使用されるポート番号。
  - `REGION_NAME` - DB インスタンスが実行中の AWS リージョン
  - `DB_USER` - アクセス先のデータベースアカウント。
  - `SSL_CERTIFICATE` - AWS リージョンに固有の、Amazon RDS の SSL 証明書。

    AWS リージョンの証明書をダウンロードするには、SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照してください。この Java プログラムファイルと同じディレクトリに SSL 証明書を配置し、クラスローダーが実行時にその証明書を見つけられるようにします。

このコード例では、デフォルトの認証情報プロバイダチェーンから AWS 認証情報を取得します。

> **Note**
>
> セキュリティ上のベストプラクティスとして、ここに示されているプロンプト以外の `DEFAULT_KEY_STORE_PASSWORD` のパスワードを指定してください。

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;
```

```
    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
```

```
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
 and password:
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
 DEFAULT_KEY_STORE_PASSWORD);
    }
```

```
   /**
    * This method returns the path of the Key Store File needed for the SSL
verification during the IAM Database Authentication to
    * the db instance.
    * @return
    * @throws Exception
    */
   private static String createKeyStoreFile() throws Exception {
       return createKeyStoreFile(createCertificate()).getPath();
   }

   /**
    *  This method generates the SSL certificate
    * @return
    * @throws Exception
    */
   private static X509Certificate createCertificate() throws Exception {
       CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
       URL url = new File(SSL_CERTIFICATE).toURI().toURL();
       if (url == null) {
           throw new Exception();
       }
       try (InputStream certInputStream = url.openStream()) {
           return (X509Certificate) certFactory.generateCertificate(certInputStream);
       }
   }

   /**
    * This method creates the Key Store File
    * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
    * @return
    * @throws Exception
    */
   private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
       File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
KEY_STORE_FILE_SUFFIX);
       try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
           KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
           ks.load(null);
           ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
           ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
       }
```

```
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM 認証および AWS SDK for Python (Boto3) を使用した DB インスタンスへの接続

次に説明するように、AWS SDK for Python (Boto3) を使用して、RDS for MariaDB、MySQL、または PostgreSQL DB インスタンスに接続できます。

前提条件

IAM 認証を使用して DB インスタンスに接続するための前提条件は以下のとおりです。

- IAM データベース認証の有効化と無効化
- IAM データベースアクセス用の IAM ポリシーの作成と使用
- IAM 認証を使用したデータベースアカウントの作成

さらに、サンプルコード内のインポートされるライブラリがシステムに存在することを確認してください。

例

コード例では、共有認証情報のプロファイルを使用します。認証情報の指定については、AWS SDK for Python (Boto3) ドキュメントの「認証情報」を参照してください。

以下のコード例で、認証トークンを生成し、それを使用して DB インスタンスに接続する方法を示します。

このコードサンプルを実行するには、AWS SDK for Python (Boto3) サイトにある AWS が必要です。

必要に応じて以下の可変の値を変更します。

- ENDPOINT - アクセス先の DB インスタンスのエンドポイント
- PORT - DB インスタンスへの接続に使用するポート番号
- USER - アクセス先のデータベースアカウント
- REGION - DB インスタンスが実行中の AWS リージョン
- DBNAME - アクセス先のデータベース
- SSLCERTIFICATE - Amazon RDS の SSL 証明書へのフルパス

  ssl_ca を使用する場合、SSL 証明書を指定します。SSL 証明書をダウンロードするには SSL/TLS を使用した DB インスタンスまたはクラスターへの接続の暗号化 を参照ください。

> **Note**
>
> DB インスタンス エンドポイントの代わりに、カスタム Route 53 DNS レコードを使用して認証トークンを生成することはできません。

このコードで MariaDB インスタンスまたは MySQL DB インスタンスに接続します。

このコードを実行する前に、Python Package Index の手順に従って PyMySQL ドライバーをインストールしてください。

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqldb.123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
```

```
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

このコードで PostgreSQL DB インスタンスに接続します。

このコードを実行する前に、Psycopg documentation の手順に従って psycopg2 をインストールしてください。

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmydb.123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)
```

```
try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

プロキシ経由で DB インスタンスに接続する場合は、「IAM 認証を使用したデータベースへの接続」を参照してください。

IAM DB 認証のトラブルシューティング

以下に、いくつかの一般的な IAM DB 認証に関する問題のトラブルシューティングのヒントと、IAM DB 認証の CloudWatch ログおよびメトリクスに関する情報を示します。

CloudWatch Logs への IAM DB 認証エラーログのエクスポート

IAM DB 認証エラーログはデータベースホストに保存され、CloudWatch Logs アカウントでこれらのログをエクスポートできます。このページのログと修復方法を使用して、IAM DB 認証の問題をトラブルシューティングします。

コンソール、AWS CLI、および RDS API から CloudWatch Logs へのログエクスポートを有効にできます。コンソールの手順については、「Amazon CloudWatch Logs へのデータベースログの発行」を参照してください。

AWS CLI から DB インスタンスを作成するときに IAM DB 認証エラーログを CloudWatch Logs にエクスポートするには、次のコマンドを使用します。

```
aws rds create-db-instance --db-instance-identifier mydbinstance \
--region us-east-1 \
--db-instance-class db.t3.large \
--allocated-storage 50 \
--engine postgres \
--engine-version 16 \
--port 5432 \
--master-username master \
--master-user-password password \
--publicly-accessible \
```

```
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

AWS CLI から DB インスタンスを変更するときに IAM DB 認証エラーログを CloudWatch Logs にエクスポートするには、次のコマンドを使用します。

```
aws rds modify-db-instance --db-instance-identifier mydbinstance \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

DB インスタンスが IAM DB 認証ログを CloudWatch Logs にエクスポートしているかどうかを確認するには、`describe-db-instances` コマンドの出力で `EnabledCloudwatchLogsExports` パラメータが `iam-db-auth-error` に設定されているかどうかを確認します。

```
aws rds describe-db-instances --region us-east-1 --db-instance-identifier mydbinstance
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

IAM DB 認証 CloudWatch メトリクス

Amazon RDS は、IAM DB 認証に関するほぼリアルタイムのメトリクスを Amazon CloudWatch アカウントに配信します。次の表に、CloudWatch で使用可能な IAM DB 認証メトリクスを示します。

メトリクス	説明
`IamDbAuthConnectionRequests`	IAM DB 認証で行われた接続リクエストの合計数。
`IamDbAuthConnectionSuccess`	成功した IAM DB 認証リクエストの合計数。
`IamDbAuthConnectionFailure`	失敗した IAM DB 認証リクエストの合計数。
`IamDbAuthConnectionFailureInvalidToken`	トークンが無効であるために失敗した IAM DB 認証リクエストの合計数。

メトリクス	説明
IamDbAuthConnectionFailureInsufficientPermissions	ポリシーまたはアクセス許可が正しくないために失敗した IAM DB 認証リクエストの合計数。
IamDbAuthConnectionFailureThrottling	IAM DB 認証スロットリングにより失敗した IAM DB 認証リクエストの合計数。
IamDbAuthConnectionFailureServerError	IAM DB 認証機能の内部サーバーエラーにより失敗した IAM DB 認証リクエストの合計数。

一般的な問題と解決策

IAM DB 認証の使用時に、次の問題が発生することがあります。問題を解決するには、表の修復ステップを使用します。

エラー	メトリクス	原因	ソリューション
[ERROR] Failed to authenticate the connection request for user *db_user* because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureInvalidToken	接続リクエストの IAM DB 認証トークンが有効な SigV4a トークンではないか、正しくフォーマットされていません。	アプリケーションでトークン生成戦略を確認します。場合によっては、トークンを有効なフォーマットで渡してください。トークンを切り捨てる (または文字列形式が正しくない) と、トークンが無効になります。
[ERROR] Failed to authenticate the connection request for user *db_user* because	IamDbAuthConnectionFailure	IAM DB 認証トークンの有効期限が切れています。トークンは 15 分間のみ有効です。	アプリケーションでトークンキャッシュやトークン再利用ロジックを確認します。15 分より古いトー

エラー	メトリクス	原因	ソリューション
the token age is longer than 15 minutes. (Status Code: 400, Error Code:ExpiredToken)	IamDbAuthConnectionFailureInvalidToken		クンは再利用しないでください。
[ERROR] Failed to authorize the connection request for user *db_user* because the IAM policy assumed by the caller 'arn:aws:sts::123456789012:assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureInsufficientPermissions	このエラーの原因としては、以下が考えられます。 • アプリケーションが引き受ける IAM ポリシーは、rds-db:connect アクションを許可しません。 • *db_user* がデータベースに接続するための正しくないロール/ポリシーを引き受けています。 • *db_user* の正しいポリシーを引き受けていますが、正しいデータベースに接続していません。	アプリケーションで引き受ける IAM ロールやポリシーを確認します。DB に接続する場合と同じポリシーを引き受けてトークンを生成します。

エラー	メトリクス	原因	ソリューション
[ERROR] Failed to authorize the connection request for user *db_user* due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureThrottling	短時間に DB への接続リクエストが多すぎます。	IAM 認証を使用して新しい接続を確立する速度を下げます。確立された接続をアプリケーションで再利用するには、RDS Proxy を使用して接続プールを実装することを検討してください。
[ERROR] Failed to authorize the connection request for user *db_user* due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureThrottling	IAM DB 認証で DB 接続を許可中に内部エラーが発生しました。	https://aws.amazon.com/premiumsupport/ に連絡して問題を調査してください。

Amazon RDS のアイデンティティおよびアクセスのトラブルシューティング

次の情報は、Amazon RDS と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- Amazon RDS でアクションを実行する権限がありません。
- iam:PassRole を実行する権限がない

- 自分の AWS アカウントの外部のユーザーに Amazon RDS リソースへのアクセスを許可したい

Amazon RDS でアクションを実行する権限がありません。

AWS マネジメントコンソール から、アクションを実行する権限がないと通知された場合は、管理者に問い合わせてサポートを依頼する必要があります。管理者は、サインイン認証情報を提供した担当者です。

以下の例のエラーは、mateojackson ユーザーがコンソールを使用して、######の詳細を表示しようとしましたが、rds:*GetWidget* アクセス許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

この場合、Mateo は、*my-example-widget* アクションを使用して rds:*GetWidget* リソースにアクセスできるように、ポリシーの更新を管理者に依頼します。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合、管理者にお問い合わせ、サポートを依頼する必要があります。管理者は、サインイン認証情報を提供した担当者です。Amazon RDS にロールを渡すことができるようにポリシーを更新するよう、管理者に依頼します。

一部の AWS サービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

以下の例のエラーは、marymajor という名前のユーザーがコンソールを使用して Amazon RDS でアクションを実行しようとした場合に発生します。ただし、アクションには、サービスロールによってサービスに許可が付与されている必要があります。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary は iam:PassRole アクションの実行が許可されるように、担当の管理者にポリシーの更新を依頼します。

自分の AWS アカウントの外部のユーザーに Amazon RDS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- Amazon RDS でこれらの機能がサポートされるかどうかを確認するには、「Amazon RDS と IAM の連携」を参照してください。
- 所有している AWS アカウント全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの「所有している別の AWS アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- リソースへのアクセスをサードパーティーの AWS アカウントに提供する方法については、 IAM ユーザーガイドのサードパーティーが所有する AWS アカウントへのアクセスの提供を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、IAM ユーザーガイドのIAM ロールとリソースベースのポリシーとの相違点を参照してください。

Amazon RDS でのログ記録とモニタリング

モニタリングは、Amazon RDS と AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。AWS には、Amazon RDS リソースをモニタリングし、潜在的なインシデントに対応するための複数のツールが用意されています。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間中、1 つのメトリクスをモニタリングします。メトリクスが特定の閾値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポ

リシーに通知が送信されます。CloudWatch アラームは、特定の状態にあるという理由ではアクションを呼び出しません。状態が変わり、それが指定した期間だけ維持される必要があります。

AWS CloudTrail ログ

CloudTrail では、Amazon RDS のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を確認できます。CloudTrail は、コンソールからの呼び出しと Amazon RDS API オペレーションへのコード呼び出しを含む、Amazon RDS のすべての API コールをイベントとしてキャプチャします。CloudTrail によって収集された情報を使用して、リクエストの作成元の IP アドレス、リクエストの実行者、リクエストの実行日時などの詳細を調べて、Amazon RDS に対してどのようなリクエストが行われたかを判断できます。詳細については、「AWS CloudTrail での Amazon RDS API コールのモニタリング」を参照してください。

拡張モニタリング

Amazon RDS には、DB インスタンスが実行されているオペレーティングシステム (OS) のリアルタイムのメトリクスが用意されています。コンソールで DB インスタンスのメトリクスを表示したり、選択したモニタリングシステムで Amazon CloudWatch Logs からの拡張モニタリング JSON 出力を使用したりできます。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。

Amazon RDS Performance Insights

Performance Insights は、既存の Amazon RDS モニタリング機能を拡張して、データベースのパフォーマンスを明確にし、これに影響を与えるあらゆる問題を分析しやすくします。Performance Insights ダッシュボードを使用してデータベースロードを視覚化したり、ロードを待機、SQL ステートメント、ホスト、ユーザー別にフィルタリングしたりできます。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。

データベースのログ

データベースログを表示、ダウンロード、モニタリングするには、AWS マネジメントコンソール、AWS CLI、または RDS API を使用します。詳細については、「Amazon RDS ログファイルのモニタリング」を参照してください。

Amazon RDS の推奨事項

Amazon RDS は、データベースリソースに対して自動化された推奨事項を示します。これらの推奨事項では、DB インスタンス設定、使用状況、パフォーマンスデータを分析して、ベストプラクティスガイダンスを提供します。詳細については、「Amazon RDS の推奨事項」を参照してください。

Amazon RDS イベントの通知

Amazon RDS では、Amazon RDS のイベントが発生したときに、Amazon Simple Notification Service (Amazon SNS) を使用して通知を送信します。これらの通知は、AWS リージョンの Amazon SNS でサポートされているすべての通知の形式を使うことができます (E メール、テキストメッセージ、HTTP エンドポイントの呼び出しなど)。詳細については、「Amazon RDS イベント通知の操作」を参照してください。

AWS Trusted Advisor

Trusted Advisor は、AWS の数十万のお客様にサービスを提供することにより得られた、運用実績から学んだベストプラクティスを活用しています。Trusted Advisor はお客様の AWS 環境を検査し、システムの可用性とパフォーマンスを向上させたりセキュリティギャップを埋めたりする機会がある場合には、推奨事項を作成します。すべての AWS のお客様は、Trusted Advisor の 5 つのチェックにアクセスできます。ビジネスまたはエンタープライズサポートプランをご利用のお客様は、すべての Trusted Advisor チェックを表示できます。

Trusted Advisor には、以下を対象とした Amazon RDS 関連のチェックがあります。

- Amazon RDS アイドル DB インスタンス
- Amazon RDS セキュリティグループのアクセスリスク
- Amazon RDS バックアップ
- Amazon RDS マルチ AZ

これらのチェックの詳細については、「Trusted Advisor のベストプラクティス (チェック)」を参照してください。

Amazon RDS のモニタリングの詳細については、「Amazon RDS インスタンスでのメトリクスのモニタリング」を参照してください。

Amazon RDS のコンプライアンス検証

サードパーティーの監査者は、複数の AWS コンプライアンスプログラムの一環として Amazon RDS のセキュリティとコンプライアンスを評価します。このプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの範囲内の AWS サービスのリストについては、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS Artifact のレポートのダウンロード」を参照してください。

Amazon RDS を使用する際のお客様のコンプライアンス責任は、データの機密性、組織のコンプライアンス目的、適用法規によって決まります。AWS は、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド — これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに焦点を当てたベースライン環境を AWS にデプロイするためのステップを示します。
- 「Architecting for HIPAA Security and Compliance on Amazon Web Services」(Amazon Web Services での HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ) – このホワイトペーパーは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法を説明しています。
- AWS コンプライアンスのリソース - お客様の業界や地域に当てはまる可能性のあるワークブックやガイドのコレクションです。
- AWS Config - この AWS サービスでは、自社プラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- AWS Security Hub CSPM – この AWS のサービス は、AWS 内のセキュリティ状態の包括的なビューを提供します。Security Hub CSPM では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、「Security Hub CSPM のコントロールリファレンス」を参照してください。

Amazon RDS の耐障害性

AWS のグローバルインフラストラクチャは、AWS リージョンとアベイラビリティーゾーンを中心として構築されています。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立し隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon RDS では、AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズに対応するための機能を提供しています。

バックアップと復元

Amazon RDS は、DB インスタンスの自動バックアップを作成および保存します。Amazon RDS は DB インスタンスのストレージボリュームのスナップショットを作成し、個々のデータベースだけではなく、その DB インスタンス全体をバックアップします。

Amazon RDS は、DB インスタンスのバックアップ期間中に DB インスタンスの自動バックアップを作成します。Amazon RDS は、指定したバックアップ保持期間に従って DB インスタンスの自動バックアップを保存します。必要に応じて、バックアップ保持期間内の任意の時点でデータベースを復旧できます。また、DB スナップショットを手動で作成して、DB インスタンスを手動でバックアップすることもできます。

DB インスタンスを作成するには、ソース DB インスタンスに障害が発生した場合に、災害対策ソリューションとして、この DB スナップショットから復元します。

詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

レプリケーション

Amazon RDS では、MariaDB、MySQL、Oracle、PostgreSQL の DB エンジンの組み込みのレプリケーション機能を使用して、リードレプリカと呼ばれる特殊なタイプの DB インスタンスをソースの DB インスタンスから作成することができます。ソース DB インスタンスに加えられた更新は、リードレプリカに非同期的にコピーされます。読み取りクエリをアプリケーションからリードレプリカに

ルーティングすることにより、ソース DB インスタンスへの負荷を減らすことができます。リードレプリカを使うと、単一 DB インスタンスの容量制約にとらわれることなく伸縮自在にスケールアウトし、読み取り負荷の高いデータベースワークロードに対応できます。ソース DB インスタンスで障害が発生した場合は、災害対策ソリューションとして、リードレプリカをスタンドアロンインスタンスに昇格させることができます。DB エンジンによっては、Amazon RDS は他にもレプリケーションオプションをサポートしています。

詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。

フェイルオーバー

Amazon RDS は、マルチ AZ 配置を使用して DB インスタンスの高可用性およびフェイルオーバーサポートを提供します。Amazon RDS は複数の異なるテクノロジーを使用してフェイルオーバーサポートを提供します。Oracle、PostgreSQL、MySQL、MariaDB DB インスタンスのマルチ AZ 配置では、Amazon のフェイルオーバーテクノロジーが使用されます。SQL Server DB インスタンスでは SQL Server データベースのミラーリング (DBM) が使用されます。

詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。

Amazon RDS でのインフラストラクチャセキュリティ

マネージドサービスとして、Amazon Relational Database Service は AWS グローバルネットワークセキュリティによって保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには「セキュリティの柱 - AWS 適切なアーキテクチャを備えたフレームワーク」の「インフラストラクチャの保護」を参照してください。

ネットワーク経由で Amazon RDS にアクセスするには、AWS が発行した API コールを使用します。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

また、Amazon RDS には、インフラストラクチャのセキュリティをサポートする機能もあります。

セキュリティグループ

セキュリティグループにより DB インスタンスに対する送受信トラフィックへのアクセスを制御します。デフォルトでは、ネットワークアクセスは DB インスタンスに対してオフになっています。セキュリティグループで IP アドレス範囲、ポート、またはセキュリティグループからのアクセスを許可するルールを指定できます。Ingress ルールを設定したら、そのセキュリティグループに関連付けられているすべての DB インスタンスに、同じルールが適用されます。

詳しくは、「セキュリティグループによるアクセス制御」を参照してください。

パブリックアクセシビリティ

Amazon VPC サービスに基づき、仮想プライベートクラウド (VPC) 内で DB インスタンスを起動する場合は、そのインスタンスのパブリックアクセシビリティをオンまたはオフにすることができます。作成した DB インスタンスにパブリック IP アドレスに解決される DNS 名を含むかどうかを指定するには、Public accessibility パラメータを使用します。このパラメータを使用することで、DB インスタンスに対するパブリックアクセスがあるかどうかを指定することができます。Public accessibility パラメータを変更することによって、DB インスタンスのパブリックアクセス可能性をオンまたはオフにすることができます。

詳細については、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。

> **Note**
>
> DB インスタンスが VPC 内にあるがパブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続や Direct Connect 接続を使用してプライベートネットワークからアクセスすることもできます。詳しくは、「ネットワーク間のトラフィックのプライバシー」を参照してください。

Amazon RDS API とインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、VPC と Amazon RDS API エンドポイント間にプライベート接続を確立できます。インターフェイスエンドポイントは を使用しますAWS PrivateLink

AWS PrivateLink を使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続なしで、Amazon RDS API オペレーションにプライベートにアクセスできます。VPC 内の DB インスタンスは、DB インスタンス を起動、変更、または終了するための Amazon RDS API エンドポイントとの通信に、パブリック IP アドレスを必要としません。また、RDS API オペレーションの使用にも、パブリック IP アドレスを必要としません。VPC と Amazon RDS 間のトラフィックは、Amazon ネットワークを離れません。

各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。Elastic Network Interface の詳細については、Amazon EC2 ユーザーガイドの「Elastic Network Interface」を参照してください。

VPC エンドポイントの詳細については、Amazon VPC ユーザーガイドの「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。RDS API オペレーションの詳細については、Amazon RDS API リファレンスを参照してください。

DB インスタンスへの接続には、インターフェイス VPC エンドポイントは必要ありません。詳細については、「VPC の DB インスタンスにアクセスするシナリオ」を参照してください。

VPC エンドポイントに関する考慮事項

Amazon RDS API エンドポイントのインターフェイス VPC エンドポイントを設定する前に、Amazon VPC ユーザーガイド の「インターフェイスエンドポイントのプロパティと制限」を確認してください。

Amazon RDS リソースの管理に関連するすべての RDS API オペレーションは、AWS PrivateLink を使用して VPC から利用することができます。

VPC エンドポイントポリシーは RDS API エンドポイントでサポートされます。デフォルトでは、エンドポイント経由で RDS API オペレーションへのフルアクセスが許可されます。詳細については、Amazon VPC ユーザーガイドの「VPC エンドポイントによるサービスのアクセス制御」を参照してください。

利用可能な状況

現在、Amazon RDS API は、次の AWS リージョンで VPC エンドポイントをサポートしています。

- 米国東部 (オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ニュージーランド)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (台北)
- アジアパシフィック (タイ)
- アジアパシフィック (東京)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (チューリッヒ)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- イスラエル (テルアビブ)

- メキシコ (中部)
- 中東 (バーレーン)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

Amazon RDS API 用のインターフェイス VPC エンドポイントの作成

Amazon RDS API 用の VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) で作成できます。詳細については、Amazon VPC ユーザーガイドのインターフェイスエンドポイントの作成を参照してください。

サービス名 `com.amazonaws.region.rds` を使用して、Amazon RDS API の VPC エンドポイントを作成します。

中国の AWS リージョンを除き、エンドポイントでプライベート DNS を有効にすると、AWS リージョンのデフォルト DNS 名 (`rds.us-east-1.amazonaws.com` など) を使用して、VPC エンドポイントで Amazon RDS に API リクエストを行うことができます。中国 (北京) および 中国 (寧夏) AWS リージョンの場合、それぞれ `rds-api.cn-north-1.amazonaws.com.cn` および `rds-api.cn-northwest-1.amazonaws.com.cn` を使用して VPC エンドポイントで API リクエストを行うことができます。

詳細については、Amazon VPC ユーザーガイド の「インターフェイスエンドポイントを介したサービスへのアクセス」を参照してください。

Amazon RDS API 用の VPC エンドポイントポリシーの作成

VPC エンドポイントに Amazon RDS API へのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、Amazon VPC ユーザーガイドの「VPC エンドポイントによるサービスのアクセス制御」を参照してください。

例: Amazon RDS API アクションの VPC エンドポイントポリシー

Amazon RDS API のエンドポイントポリシーの例を次に示します。このポリシーは、エンドポイントにアタッチされると、すべてのリソースのすべてのプリンシパルに対して、登録されている Amazon RDS API アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

例: 指定した AWS アカウントからのすべてのアクセスを拒否する VPC エンドポイントポリシー

以下の VPC エンドポイントポリシーは、AWS アカウント 123456789012 からリソースへのエンドポイントを使用したすべてのアクセスを拒否します。このポリシーは、他のアカウントからのすべてのアクションを許可します。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
```

```
}
```

Amazon RDS のセキュリティのベストプラクティス

AWS Identity and Access Management (IAM) アカウントを使用して、Amazon RDS API オペレーション、特に Amazon RDS リソースの作成、変更、削除を行うオペレーションへのアクセスを制御します。そのようなリソースには、DB インスタンス、セキュリティグループ、およびパラメータグループなどがあります。また、IAM を使用して、DB インスタンスのバックアップや復元など、一般的な管理アクションを実行するアクションも制御します。

- Amazon RDS リソースを管理するユーザー (本人を含む) ごとに個別のユーザーを作成します。Amazon RDS リソースの管理には、AWS ルート認証情報を使用しないでください。
- それぞれの職務の実行に最低限必要になる一連のアクセス許可を各ユーザーに付与します。
- IAM グループを使用して、複数のユーザーのアクセス許可を効果的に管理します。
- IAM 認証情報のローテーションを定期的に行います。
- Amazon RDS のシークレットが自動的にローテーションされるように、AWS Secrets Manager を設定します。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager シークレットのローテーション」を参照してください。認証情報は、AWS Secrets Manager プログラムから取得することもできます。詳細については、AWS Secrets Manager ユーザーガイドの「シークレット値の取得」を参照してください。

Amazon RDS でのセキュリティの詳細については、「Amazon RDS でのセキュリティ」を参照してください。IAM の詳細については、「AWS Identity and Access Management」を参照してください。IAM のベストプラクティスについては、「IAM のベストプラクティス」を参照してください。

AWS Security Hub CSPM は、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub CSPM を使用して RDS リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「Amazon リレーショナルデータベースサービスコントロール」を参照してください。

Security Hub CSPM を使用して、セキュリティのベストプラクティスに関連する RDS の使用状況をモニタリングできます。詳細については、「What is AWS Security Hub CSPM?」を参照してください。

AWS マネジメントコンソール、AWS CLI、RDS API を使用して、マスターユーザーのパスワードを変更します。SQL クライアントなどの別のツールを使用する場合、マスターユーザーのパスワードを変更すると、ユーザーの権限が意図せずに取り消される可能性があります。

セキュリティグループによるアクセス制御

VPC セキュリティグループにより DB インスタンスに対する送受信トラフィックへのアクセスを制御します。デフォルトでは、ネットワークアクセスは DB インスタンスに対してオフになっています。セキュリティグループで IP アドレス範囲、ポート、またはセキュリティグループからのアクセスを許可するルールを指定できます。Ingress ルールを設定したら、そのセキュリティグループに関連付けられているすべての DB インスタンスに、同じルールが適用されます。セキュリティグループでは最大 20 のルールを指定できます。

VPC セキュリティグループの概要

VPC セキュリティグループの各ルールにより、特定のソースがその VPC セキュリティグループに関連付けられている VPC 内の DB インスタンスにアクセスできるようになります。ソースとしては、アドレスの範囲 (203.0.113.0/24 など) または別の VPC セキュリティグループを指定できます。VPC セキュリティグループをソースとして指定すると、ソース VPC セキュリティグループを使用するすべてのインスタンス (通常はアプリケーションサーバー) からの受信トラフィックを許可することになります。VPC セキュリティグループには、インバウンドトラフィックとアウトバウンドトラフィック両方を制御するルールを追加できます。ただし、アウトバウンドトラフィックのルールは通常、DB インスタンスには適用されません。送信トラフィックのルールは、DB インスタンスがクライアントである場合にのみ適用されます。例えば、送信トラフィックのルールは送信データベースリンクがある Oracle DB インスタンスに適用されます。VPC セキュリティグループを作成するには、「Amazon EC2 API」を使用するか、VPC コンソールの [Security Group] (セキュリティグループ) オプションを使用する必要があります。

VPC 内のインスタンスへのアクセスを許可する VPC セキュリティグループのルールを作成するときは、そのルールがアクセスを許可するアドレスの範囲ごとにポートを指定する必要があります。例えば、VPC 内のインスタンスへの Secure Shell (SSH) アクセスをオンにするには、指定したアドレスの範囲の TCP ポート 22 に対するアクセスを許可するルールを作成します。

VPC 内の異なるインスタンスに対する異なるポートへのアクセスを許可する複数の VPC セキュリティグループを設定できます。例えば、VPC のウェブサーバーに TCP ポート 80 へのアクセスを許可する VPC セキュリティグループを作成できます。次に、VPC の RDS for MySQL DB インスタンスの TCP ポート 3306 へのアクセスを許可する別の VPC セキュリティグループを作成できます。

VPC セキュリティグループの詳細については、Amazon Virtual Private Cloud ユーザーガイドの「セキュリティグループ」を参照してください。

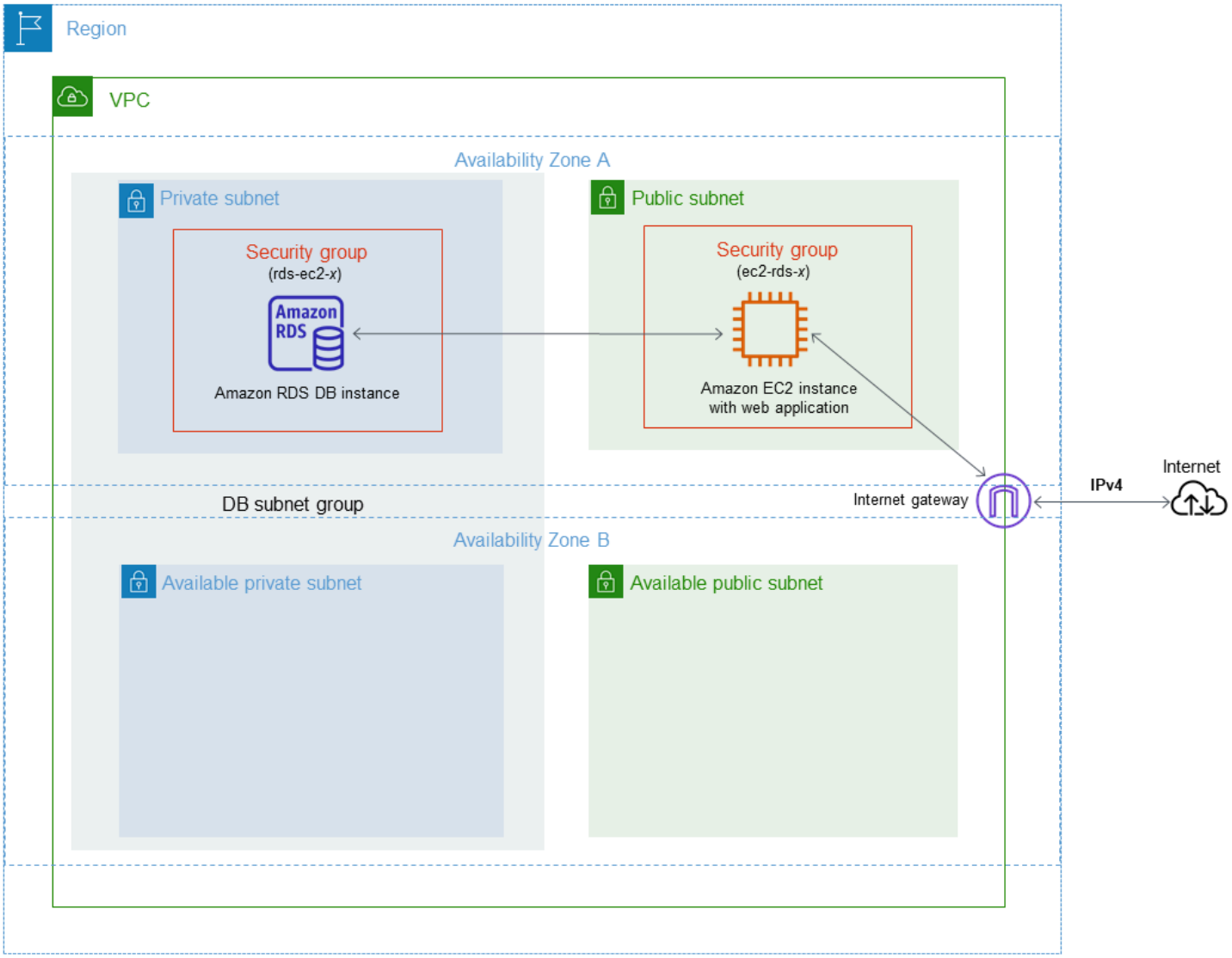
> **Note**
>
> DB インスタンスが VPC 内にあるが、パブリックアクセス可能でない場合は、AWS Site-to-Site VPN 接続または Direct Connect 接続を使用してプライベートネットワークからアクセスすることもできます。詳細については、「ネットワーク間のトラフィックのプライバシー」を参照してください。

セキュリティグループのシナリオ

VPC 内の DB インスタンスの一般的な用途は、同じ VPC 内の Amazon EC2 インスタンスで実行され、VPC の外にあるクライアントアプリケーションによってアクセスされるアプリケーションサーバーとデータを共有することです。このシナリオでは、AWS マネジメントコンソールの RDS および VPC ページ、または RDS および EC2 API オペレーションを使用して、必要なインスタンスおよびセキュリティグループを作成します。

1. VPC セキュリティグループ (「sg-0123ec2example」など) を作成し、ソースとしてクライアントアプリケーションの IP アドレスを使用するという受信ルールを定義します。このセキュリティグループにより、クライアントアプリケーションは、このセキュリティグループを使用する VPC 内の EC2 インスタンスに接続できるようになります。
2. アプリケーションの EC2 インスタンスを作成し、前のステップで作成した VPC セキュリティグループ (「sg-0123ec2example」) に EC2 インスタンスを追加します。
3. 2 つ目の VPC セキュリティグループ (「sg-6789rdsexample」など) を作成し、ステップ 1 で作成した VPC セキュリティグループ (「sg-0123ec2example」) をソースとして指定して新しいルールを作成します。
4. 新しい DB インスタンスを作成し、その DB インスタンスを前のステップで作成した VPC セキュリティグループ (sg-6789rdsexample) に追加します。DB インスタンスを作成するとき、ステップ 3 で作成した VPC セキュリティグループ (sg-6789rdsexample) のルールに指定した同じポート番号を使用します。

以下の図に、このシナリオを示しています。



このシナリオの VPC 設定の手順の詳細については、「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。VPC の使用に関する詳細については、「Amazon VPC と Amazon RDS」を参照してください。

VPC セキュリティグループを作成する

DB インスタンスの VPC セキュリティグループは、VPC コンソールを使って作成できます。セキュリティグループを作成する方法については、Amazon Virtual Private Cloud ユーザーガイドの「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」および「セキュリティグループ」を参照してください。

セキュリティグループを DB インスタンスと関連付ける

RDS コンソールの [変更]、`ModifyDBInstance` Amazon RDS API、または `modify-db-instance` AWS CLI コマンドを使用して、セキュリティグループを DB インスタンスに関連付けることができます。

次の CLI の例では、特定の VPC セキュリティグループを関連付け、DB インスタンスから DB セキュリティグループを削除します。

```
aws rds modify-db-instance --db-instance-identifier dbName --vpc-security-group-ids sg-ID
```

DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。DB スナップショットから DB インスタンスを復元するときのセキュリティグループの考慮事項については、「セキュリティグループに関する考慮事項」を参照してください。

> **Note**
>
> ポート値がデフォルト以外の値に設定されている場合、RDS コンソールにはデータベースのさまざまなセキュリティグループルール名が表示されます。

RDS for Oracle DB インスタンスでは、Oracle Enterprise Manager Database Express (OEM)、Oracle Management Agent for Enterprise Manager Cloud Control (OEM Agent)、および Oracle Secure Sockets Layer オプションのセキュリティグループオプション設定を入力することで、追加のセキュリティグループを関連付けることができます。この場合、DB インスタンスに関連付けられているセキュリティグループとオプション設定の両方が DB インスタンスに適用されます。これらのオプショングループの詳細については、「Oracle Enterprise Manager」、「Enterprise Manager Cloud Control 向け Oracle Management Agent」、「Oracle Secure Sockets Layer」を参照してください。

マスターユーザーアカウント権限

新しい DB インスタンスを作成すると、使用するデフォルトマスターユーザーがその DB インスタンスの特定の権限を取得します。DB インスタンスの作成後にマスターユーザー名を変更することはできません。

> **Important**
>
> アプリケーションではマスターユーザーを直接使用しないことを強くお勧めします。代わりに、アプリケーションに必要な最小の特権で作成されたデータベースユーザーを使用するというベストプラクティスに従ってください。

> **Note**
>
> マスターユーザーのアクセス許可を誤って削除した場合は、DB インスタンスを変更して新しいマスターユーザーパスワードを設定することで復元できます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

次の表は、各データベースエンジンに対してマスターユーザーが取得する権限とデータベースロールを示しています。

データベースエンジン	システム権限	データベースロール
RDS for Db2	マスターユーザーは `masterdba` グループに割り当てられ、`master_user_role` が割り当てられます。 SYSMON、DATAACCESS 付き DBADM、および ACCCESSCTRL 、BINDADD、CONNECT、CREATETAB 、CREATE_SECURE_OBJECT 、EXPLAIN、IMPLICIT_SCHEMA 、LOAD、SQLADM、WLMADM	DBA,DBA_RESTRICTED , DEVELOPER ,ROLE_NULLID_PACKAGES , ROLE_PROCEDURES ,ROLE_TABLESPACES 詳細については、「Amazon RDS for Db2 のデフォルトロール」を参照してください。
RDS for MariaDB	SELECT,INSERT,UPDATE,DELETE, CREATE,DROP,RELOAD, PROCESS,REFERENCES ,INDEX, ALTER,SHOW DATABASES	—

データベースエンジン	システム権限	データベースロール
	,CREATE TEMPORARY TABLES ,LOCK TABLES, EXECUTE,REPLICATION CLIENT,CREATE VIEW ,SHOW VIEW,CREATE ROUTINE, ALTER ROUTINE,CREATE USER, EVENT,TRIGGER,REPLICATION SLAVE RDS for MariaDB バージョン 11.4 以降、マスターユーザーには SHOW CREATE ROUTINE 権限も付与されます。	
RDS for MySQL バージョン 8.0.36 以上	SELECT,INSERT,UPDATE, DELETE,CREATE,DROP, RELOAD,PROCESS, REFERENCES ,INDEX,ALTER, SHOW DATABASES ,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE, REPLICATION SLAVE ,REPLICATION CLIENT , CREATE VIEW,SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,EVENT,TRIGGER, CREATE ROLE,DROP ROLE, APPLICATION_PASSWORD_ADMIN , ROLE_ADMIN ,SET_USER_ID , XA_RECOVER_ADMIN	rds_superuser_role rds_superuser_role の詳細については、「RDS for MySQL のロールベースの権限モデル」を参照してください。
RDS for MySQL バージョン 8.0.36 未満	SELECT,INSERT,UPDATE, DELETE,CREATE,DROP, RELOAD,PROCESS, REFERENCES ,INDEX,ALTER, SHOW DATABASES ,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE, REPLICATION CLIENT ,CREATE VIEW, SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE USER,EVENT, TRIGGER,REPLICATION SLAVE	—

データベースエンジン	システム権限	データベースロール
RDS for PostgreSQL	CREATE ROLE,CREATE DB, PASSWORD VALID UNTIL INFINITY,CREATE EXTENSION ,ALTER EXTENSION ,DROP EXTENSION ,CREATE TABLESPACE ,ALTER <OBJECT> OWNER,CHECKPOINT , PG_CANCEL_BACKEND( ) , PG_TERMINATE_BACKEND() ,SELECT PG_STAT_REPLICATION ,EXECUTE PG_STAT_STATEMENTS_RESET() ,OWN POSTGRES_FDW_HANDLER() ,OWN POSTGRES_FDW_VALIDATOR() ,OWN POSTGRES_FDW , EXECUTE PG_BUFFERCACHE_PAGES() ,SELECT PG_BUFFERCACHE	RDS_SUPERUSER RDS_SUPERUSER の詳細については、「PostgreSQL のロールとアクセス権限について」を参照してください。

データベースエンジン	システム権限	データベースロール
RDS for Oracle	ADMINISTER DATABASE TRIGGER ,ALTER DATABASE LINK,ALTER PUBLIC DATABASE LINK, AUDIT SYSTEM,CHANGE NOTIFICATION , DROP ANY DIRECTORY ,EXEMPT ACCESS POLICY,EXEMPT IDENTITY POLICY,EXEMPT REDACTION POLICY,FLASHBACK ANY TABLE, GRANT ANY OBJECT PRIVILEGE ,RESTRICTED SESSION ,SELECT ANY TABLE,UNLIMITED TABLESPACE	DBA **Note** DBA ロールには、以下の権限が適用されません。 ALTER DATABASE,ALTER SYSTEM, CREATE ANY DIRECTORY ,CREATE EXTERNAL JOB,CREATE PLUGGABLE DATABASE, GRANT ANY PRIVILEGE ,GRANT ANY ROLE,READ ANY FILE GROUP

データベースエンジン	システム権限	データベースロール
Amazon RDS for Microsoft SQL Server	ADMINISTER BULK OPERATIONS ,ALTER ANY CONNECTION ,ALTER ANY CREDENTIAL , ALTER ANY EVENT SESSION,ALTER ANY LINKED SERVER,ALTER ANY LOGIN,ALTER ANY SERVER AUDIT,ALTER ANY SERVER ROLE, ALTER SERVER STATE,ALTER TRACE, CONNECT SQL,CREATE ANY DATABASE, VIEW ANY DATABASE,VIEW ANY DEFINITION ,VIEW SERVER STATE,ALTER ON ROLE SQLAgentOperatorRole	DB_OWNER (データベースレベルのロール)、PROCESSADMIN (サーバーレベルのロール)、SETUPADMIN (サーバーレベルのロール)、SQLAgentUserRole (データベースレベルのロール)、SQLAgentReaderRole (データベースレベルのロール)、SQLAgentOperatorRole (データベースレベルのロール)

Amazon RDS のサービスにリンクされたロールの使用

Amazon RDS は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用します。サービスにリンクされたロールは、Amazon RDS に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、Amazon RDS による事前定義済みのロールであり、ユーザーに代わってサービスから AWS の他のサービスを呼び出すために必要なすべてのアクセス許可を備えています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Amazon RDS の使用が簡単になります。サービスにリンクされたロールのアクセス許可は、Amazon RDS により定義されます。特に指定されている場合を除き、Amazon RDS のみがそのロールを引き受けることができます。定義される許可は、信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

ロールを削除するには、まず関連リソースを削除します。これにより、リソースへの意図しないアクセスによるアクセス許可の削除が防止され、Amazon RDS リソースは保護されます。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS サービス」を参照のうえ、[サービスにリンクされたロール] の欄が [はい] になっているサービスを検索してください。そのサービスに関するサービスにリンクされたロールのドキュメントを表示するには、リンクが設定されている [はい] を選択します。

Amazon RDS のサービスにリンクされたロールのアクセス許可

Amazon RDSでは、AWSServiceRoleForRDS と呼ばれるサービスにリンクされたロールを使用します。これにより Amazon RDS は DB インスタンスに代わって AWS サービスを呼び出せるようになります。

サービスにリンクされたロール AWSServiceRoleForRDS では、以下のサービスを信頼してロールを引き受けます。

- `rds.amazonaws.com`

このサービスにリンクされたロールには、アカウントで操作するためのアクセス許可を付与する `AmazonRDSServiceRolePolicy` というアクセス許可ポリシーがアタッチされています。

JSON ポリシードキュメントを含むこのポリシーの詳細については、AWS マネージドポリシーリファレンスガイドの「AmazonRDSServiceRolePolicy」を参照してください。

> **Note**
>
> サービスリンクロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するにはアクセス許可を設定する必要があります。次のエラーメッセージが表示された場合は、以下のように対応します。
> リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。
> 次のアクセス許可が有効であることを確認します。
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> 詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon RDS のサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。DB インスタンスを作成する際、Amazon RDS がサービスにリンクされたロールを作成します。

> **Important**
>
> サービスにリンクされたロールのサポートが開始された 2017 年 12 月 1 日より前に Amazon RDS サービスを使用している場合、その時点で Amazon RDS が、ご使用のアカウント内に AWSServiceRoleForRDS ロールを作成しています。詳細については、「AWS アカウントに新しいロールが表示される」を参照してください。

このサービスにリンクされたロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。DB インスタンスを作成する際、Amazon RDS がサービスにリンクされたロールを再度作成します。

Amazon RDS のサービスにリンクされたロールの編集

Amazon RDS では、サービスにリンクされたロール AWSServiceRoleForRDS を編集できません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの編集」を参照してください。

Amazon RDS のサービスにリンクされたロールの削除

サービスにリンクされたロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。ただし、サービスにリンクされたロールを削除する前に、すべての DB インスタンス を削除する必要があります。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除するには、まずそのロールにアクティブなセッションがないことを確認し、そのロールで使用されているリソースをすべて削除する必要があります。

サービスにリンクされたロールにアクティブなセッションがあるかどうかを、IAM コンソールで確認するには

1. AWS マネジメントコンソール にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールのナビゲーションペインで [ロール] を選択します。次に、AWSServiceRoleForRDS ロールの名前 (チェックボックスではありません) を選択します。
3. 選択したロールの [概要] ページで、[最終アクセス] タブを選択します。
4. [最終アクセス] タブで、サービスにリンクされたロールの最新のアクティビティを確認します。

> **Note**
>
> Amazon RDS が AWSServiceRoleForRDS ロールを使用しているかどうか不明な場合は、ロールを削除してみてください。サービスでロールが使用されている場合、削除は

失敗し、ロールが使用されている AWS リージョンが表示されます。ロールが使用されている場合は、ロールを削除する前にセッションが終了するのを待つ必要があります。サービスにリンクされたロールのセッションを取り消すことはできません。

AWSServiceRoleForRDS ロールを削除する場合、初期にすべての DB インスタンスを削除する必要があります。

すべてのインスタンスの削除

以下のいずれかの手順を使用して、インスタンスをそれぞれ削除します。

インスタンスを削除するには (コンソール)

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. ナビゲーションペインで、[データベース] を選択します。
3. 削除するインスタンスを選択します。
4. [アクション] について [削除] を選択します。
5. [最終スナップショットを作成しますか?] で、[はい] または [いいえ] を選択します。
6. 前のステップで [はい] を選択した場合は、[最終スナップショット名] に最終スナップショットの名前を入力します。
7. [削除] を選択します。

インスタンスを削除するには (CLI)

AWS CLI コマンドリファレンス の「`delete-db-instance`」を参照してください。

インスタンスを削除するには (API)

Amazon RDS API Reference の「`DeleteDBInstance`」を参照してください。

IAM コンソール、IAM CLI、または IAM API を使用して、AWSServiceRoleForRDS サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの削除」を参照してください。

Amazon RDS Custom のサービスにリンクされたロール許可

Amazon RDS Custom は、AWSServiceRoleForRDSCustom と呼ばれるサービスにリンクされたロールを使用します。これにより RDS Custom が RDS DB リソースに代わって AWS のサービスを呼び出せるようになります。

サービスにリンクされたロール AWSServiceRoleForRDSCustom は、次のサービスを信頼してロールを引き受けます。

- custom.rds.amazonaws.com

このサービスにリンクされたロールには、アカウントで操作するためのアクセス許可を付与する AmazonRDSCustomServiceRolePolicy というアクセス許可ポリシーがアタッチされています。

RDS Custom のサービスにリンクされたロールの作成、編集、および削除は、Amazon RDS の場合と同じ方法で行います。詳細については、「AWS マネージドポリシー: AmazonRDSCustomServiceRolePolicy」を参照してください。

> **Note**
>
> サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。次のエラーメッセージが表示された場合は、以下のように対応します。
> リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。
> 次のアクセス許可が有効であることを確認します。

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
AmazonRDSCustomServiceRolePolicy",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"custom.rds.amazonaws.com"
        }
    }
}
```

詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon RDS Beta のサービスにリンクされたロール許可

Amazon RDS は、AWSServiceRoleForRDSBeta と呼ばれるサービスにリンクされたロールを使用します。これにより Amazon RDS が RDS DB リソースに代わって AWS のサービスを呼び出せるようになります。

サービスにリンクされたロール AWSServiceRoleForRDSBeta では、以下のサービスを信頼してロールを引き受けます。

- rds.amazonaws.com

このサービスにリンクされたロールには、アカウントで操作するためのアクセス許可を付与する AmazonRDSBetaServiceRolePolicy というアクセス許可ポリシーがアタッチされています。詳細については、「AWS マネージドポリシー: AmazonRDSBetaServiceRolePolicy」を参照してください。

> **Note**
>
> サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。次のエラーメッセージが表示された場合は、以下のように対応します。
> リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。
> 次のアクセス許可が有効であることを確認します。
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
> ```

```
    }
}
```

詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon RDS Preview のサービスリンクロール

Amazon RDS は、AWSServiceRoleForRDSPreview と呼ばれるサービスにリンクされたロールを使用します。これにより Amazon RDS が RDS DB リソースに代わって AWS のサービスを呼び出せるようになります。

サービスリンクロール AWSServiceRoleForRDSPreview は、次のサービスを信頼してそのロールを引き受けます。

- rds.amazonaws.com

このサービスにリンクされたロールには、アカウントで操作するためのアクセス許可を付与する AmazonRDSPreviewServiceRolePolicy というアクセス許可ポリシーがアタッチされています。詳細については、「AWS マネージドポリシー: AmazonRDSPreviewServiceRolePolicy」を参照してください。

> **Note**
>
> サービスにリンクされたロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。次のエラーメッセージが表示された場合は、以下のように対応します。
> リソースを作成できません。サービスにリンクされたロールを作成するために必要なアクセス許可があることを確認します。それ以外の場合は、時間をおいてからもう一度お試しください。
> 次のアクセス許可が有効であることを確認します。
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
> ```

```
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"custom.rds.amazonaws.com"
        }
    }
}
```

詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon VPC と Amazon RDS

Amazon Virtual Private Cloud (Amazon VPC) を使用すると、Amazon RDS DB インスタンスなどの AWS リソースを仮想プライベートクラウド (VPC) で起動できます。

VPC を使用する場合、仮想ネットワーキング環境を制御できます。独自の IP アドレスの範囲を選択し、サブネットを作成してルーティングおよびアクセス制御リストを設定できます。VPC で DB インスタンスを実行するために、追加料金はかかりません。

アカウントにはデフォルト VPC があります。新しいすべての DB インスタンスは、特に指定がない限り、デフォルトの VPC 内に作成されます。

トピック

- VPC 内の DB インスタンスの使用
- DB インスタンスの VPC の更新
- VPC の DB インスタンスにアクセスするシナリオ
- チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)
- チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード)
- VPC 外の DB インスタンスを VPC 内に移行する

以下に、Amazon RDS DB インスタンス に関連する VPC の機能について説明します。Amazon VPC の詳細については、Amazon VPC 入門ガイドおよびAmazon VPC ユーザーガイドを参照してください。

VPC 内の DB インスタンスの使用

DB インスタンスは仮想プライベートクラウド (VPC) 内にあります。VPC は、AWS クラウドの他の仮想ネットワークから論理的に切り離された仮想ネットワークです。Amazon VPC では、Amazon RDS DB インスタンスや Amazon EC2 インスタンスなど、AWS リソースを VPC で起動できます。VPC は、自分のアカウントに属するデフォルト VPC を使用するか、独自に作成することもできます。すべての VPC は、AWS アカウントに関連付けられます。

デフォルト VPC には、VPC 内でリソースを隔離するために使用できる 3 つのサブネットがあります。デフォルト VPC には、VPC 外から VPC 内のリソースへのアクセスを可能にするインターネットゲートウェイもあります。

VPC 内と VPC 外の Amazon Aurora DB クラスターが関係するシナリオのリストについては、「VPC の DB インスタンスにアクセスするシナリオ」を参照してください。

トピック

- VPC 内の DB インスタンスの使用
- VPC 暗号化コントロール
- DB サブネットグループの使用
- 共有サブネット
- Amazon RDS IP アドレス指定
- VPC 内の DB インスタンスをインターネットから隠す
- VPC に DB インスタンスを作成する

以下のチュートリアルでは、一般的な Amazon RDS 状況で使用できる VPC の作成方法を学ぶことができます。

- チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)
- チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード)

VPC 内の DB インスタンスの使用

次に、VPC の DB インスタンスの使用に関するヒントを紹介します。

- VPC では、少なくとも 2 つのサブネットを指定する必要があります。これらのサブネットは、DB インスタンスをデプロイする AWS リージョン 内の 2 つの異なるアベイラビリティーゾーンに存在している必要があります。サブネットは、VPC の IP アドレス範囲の指定可能なセグメントで、セキュリティや運用上のニーズに基づいて DB インスタンスをグループ化することができます。

  マルチ AZ 配置の場合、AWS リージョン 内の 2 つ以上のアベイラビリティーゾーンにサブネットを定義すると、Amazon RDS は必要に応じて別のアベイラビリティーゾーンに新しいスタンバイを作成できるようになります。シングル AZ 配置の場合も、どこかの時点でマルチ AZ 配置に変換する場合に備えてこのように定義してください。

  > **Note**
  >
  > ローカルゾーンの DB サブネットグループは、サブネットを 1 つだけ持つことができます。

- VPC の DB インスタンスをパブリックにアクセス可能にする場合は、VPC 属性の DNS hostnames と DNS resolution を有効にしてください。

- ご利用の VPC では、DB サブネットグループを作成する必要があります。DB サブネットグループを作成するには、作成したサブネットを指定します。Amazon RDS は、サブネットとそのサブネットグループ内の IP アドレスを選択し、DB インスタンスに関連付けます。DB インスタンスは、そのサブネットを含むアベイラビリティーゾーンを使用します。
- VPC には、DB インスタンスへのアクセスを許可する VPC セキュリティグループが必要です。

  詳細については、「VPC の DB インスタンスにアクセスするシナリオ」を参照してください。
- 各サブネットの CIDR ブロックは、フェイルオーバーやコンピュートスケーリングの見積もりなどのメンテナンス作業中に Amazon RDS が使用する予備の IP アドレスに十分対応できる大きさが必要です。例えば、通常は 10.0.0.0/24 や 10.0.1.0/24 などの範囲で十分な大きさです。
- VPC では、インスタンスのテナント属性が default または dedicated のいずれかに設定されます。デフォルト VPC では、インスタンスのテナント属性はすべて default に設定され、DB インスタンスのすべてのクラスがサポートされます。

  インスタンステナンシーを dedicated に設定した専有 VPC に DB インスタンスを保持する場合は、その DB インスタンスの DB インスタンスクラスは、Amazon EC2 で承認された 専有インスタンスタイプのいずれかである必要があります。例えば、r5.large Rc2 ハードウェア専有インスタンスは、db.r5.large DB インスタンスクラスに対応します。VPC のインスタンステナンシーについては、Amazon Elastic Compute Cloud ユーザーガイドの「ハードウェア専有インスタンス」を参照してください。

  ハードウェア専有インスタンスに対応するインスタンスタイプの詳細については、Amazon EC2 の料金ページで「Amazon EC2 のハードウェア専有インスタンス」を参照してください。

  > **Note**
  >
  > インスタンスのテナンシー属性を DB インスタンス専有に設定しても、DB インスタンスが専有ホストで実行されることは保証されません。

- オプショングループを DB インスタンスに割り当てると、その DB インスタンスの VPC に関連付けられます。このリンクは、別の VPC 内に DB インスタンスを復元しようとしても、その DB インスタンスに割り当てられているオプショングループは使用できないことを意味します。
- 別の VPC 内に DB インスタンスを復元する場合は、デフォルトのオプショングループを DB インスタンスに割り当てるか、その VPC にリンクされているオプショングループを割り当てるか、新しいオプショングループを作成して DB インスタンスに割り当てる必要があります。Oracle TDE などの永続または固定オプションを使用する場合は、別の VPC 内に DB インスタンスを復元するときに、永続または固定オプションを含む新しいオプショングループを作成する必要があります。

VPC 暗号化コントロール

VPC 暗号化コントロールを使用すると、VPC 内のすべてのネットワークトラフィックに対して転送中の暗号化を適用できます。暗号化コントロールを使用して、指定された VPC で暗号化可能な Nitro ベースのハードウェアのみをプロビジョニングできるようにすることで、規制コンプライアンス要件を満たします。暗号化コントロールは、プロビジョニング中ではなく API リクエスト時に互換性の問題も検出します。既存のワークロードは引き続き動作し、新しい互換性のないリクエストのみがブロックされます。

VPC コントロールモードを次のように設定して、VPC 暗号化コントロールを設定します。

- 無効 (デフォルト)
- モニター
- 強制

VPC の現在のコントロールモードを確認するには、AWS マネジメントコンソールまたは DescribeVpcs CLI または API コマンドを使用します。

VPC が暗号化を強制する場合は、その VPC で転送中の暗号化をサポートする Nitro ベースの DB インスタンスのみをプロビジョニングできます。詳細については、「DB インスタンスクラスタイプ」を参照してください。Nitro インスタンスの詳細については、「Amazon EC2 ユーザーガイド」の「AWS Nitro System 上に構築されたインスタンス」を参照してください。

> **Note**
>
> 暗号化が強制された VPC で互換性のない DB インスタンスをプロビジョニングしようとすると、Amazon RDS は `VpcEncryptionControlViolationException` 例外を返します。

DB サブネットグループの使用

サブネットは、VPC の IP アドレス範囲のセグメントで、セキュリティや運用上のニーズに基づいてリソースをグループ化するために指定します。DB サブネットグループは VPC に作成するサブネット (通常はプライベート) のコレクションで、DB インスタンス用に指定します。DB サブネットグループを使用することにより、AWS CLI または RDS API を使用して DB インスタンスを作成するときに、特定の VPC を指定することができます。コンソールを使用する場合は、使用する VPC とサブネットグループを選択できます。

各 DB サブネットグループには、特定の AWS リージョン 内の少なくとも 2 つのアベイラビリティーゾーンにサブネットが必要です。VPC に DB インスタンスを作成するときに、DB サブネットグループを選択する必要があります。DB サブネットグループから、Amazon RDS は DB インスタンスに関連付けるサブネットとそのサブネット内の IP アドレスを選択します。DB は、そのサブネットを含むアベイラビリティーゾーンを使用します。 Amazon RDS は、空き IP アドレス空間を持つサブネットから IP アドレスを割り当てます。

マルチ AZ 配置のプライマリ DB インスタンスに障害が発生した場合、Amazon RDS は対応するスタンバイを昇格させ、その後、他のアベイラビリティーゾーンの 1 つのサブネットの IP アドレスを使用して、新しいスタンバイを作成できます。

DB サブネットグループのサブネットはパブリックまたはプライベートのいずれかです。サブネットは、ネットワークアクセス制御リスト (ネットワーク ACL) とルーティングテーブルに定義した設定に応じて、パブリックまたはプライベートになります。DB インスタンスをパブリックにアクセス可能にするには、その DB サブネットグループ内のすべてのサブネットがパブリックである必要があります。パブリックにアクセスできる DB インスタンスに関連付けられているサブネットがパブリックからプライベートに変更された場合、DB インスタンスの可用性に影響する可能性があります。

デュアルスタックモードをサポートする DB サブネットグループを作成するには、DB サブネットグループに追加する各サブネットに Internet Protocol version 6 (IPv6) CIDR ブロックが関連付けられていることを確認してください。詳細については、Amazon RDS IP アドレス指定 と Amazon VPC ユーザーガイドの「IPv6 に移行する」を参照してください。

> **Note**
>
> ローカルゾーンの DB サブネットグループは、サブネットを 1 つだけ持つことができます。

Amazon RDS は、VPC に DB インスタンスを作成すると、DB サブネットグループから選択した IP アドレスを使用して、DB インスタンスにネットワークインターフェイスを割り当てます。ただし、DB インスタンスに接続するときにはドメインネームシステム (DNS) 名を使用することを強くお勧めします。基になる IP アドレスはフェイルオーバー中に変わるため、ドメインネームシステム (DNS) 名を使用することを強くお勧めします。

> **Note**
>
> VPC で実行する DB インスタンスごとに、Amazon RDS による復旧アクション用として、DB サブネットグループのサブネットごとに最低 1 つのアドレスを確保してください。

共有サブネット

DB インスタンスは共有 VPC に作成できます。

共有 VPC を使用する際に留意すべき点がいくつかあります。

- DB インスタンスを共有 VPC サブネットから非共有 VPC サブネットに、またはその逆に移動できます。
- 共有 VPC の参加者は、DB インスタンスを作成できるように、VPC にセキュリティグループを作成する必要があります。
- 共有 VPC の所有者と参加者は、SQL クエリを使用してデータベースにアクセスできます。ただし、リソースに対して任意の API 呼び出しを行うことができるのは、リソースの作成者だけです。

Amazon RDS IP アドレス指定

IP アドレスは、VPC 内のリソースの相互通信と、インターネット経由でのリソースとの通信を可能にします。Amazon RDS は、IPv4 と IPv6 の両方のアドレス指定プロトコルをサポートしています。デフォルトでは、Amazon RDS と Amazon VPC は IPv4 アドレス指定プロトコルを使用します。この動作をオフにすることはできません。VPC の作成時には、IPv4 CIDR ブロック (プライベート IPv4 アドレスの範囲) を指定する必要があります。必要に応じて、IPv6 CIDR ブロックを VPC とサブネットに割り当て、そのブロックからサブネットの DB インスタンスに IPv6 アドレスを割り当てることができます。

IPv6 プロトコルのサポートにより、サポートされる IP アドレスの数が増えます。IPv6 プロトコルを使用することで、インターネットの今後の成長に十分なアドレスを確保できます。新規および既存の RDS リソースは、VPC 内で IPv4 アドレスと IPv6 アドレスを使用できます。アプリケーションの異なる部分で使用される 2 つのプロトコル間のネットワークトラフィックの設定、保護、および変換を行うと、運用上のオーバーヘッドが発生する可能性があります。Amazon RDS リソースについては IPv6 プロトコルを標準化して、ネットワーク構成を簡素化できます。

トピック

- IPv4 アドレス
- IPv6 アドレス
- デュアルスタックモード

IPv4 アドレス

VPC を作成するときには、その VPC の IPv4 アドレスの範囲を CIDR ブロックの形式で指定する必要があります (`10.0.0.0/16` など)。DB サブネットグループは、DB インスタンスが使用できる、この CIDR ブロック内の IP アドレスの範囲を定義します。これらの IP アドレスはプライベートまたはパブリックです。

プライベート IPv4 アドレスは、インターネットから到達できない IP アドレスです。DB インスタンスと同じ VPC 内の他のリソース (Amazon EC2 インスタンスなど) との通信には、プライベート IPv4 アドレスを使用できます。各 DB インスタンスには、VPC 内の通信用のプライベート IP アドレスがあります。

パブリック IP アドレスは、インターネットから到達可能な IPv4 アドレスです。DB インスタンスとインターネット上のリソース (SQL クライアントなど) との通信には、パブリックアドレスを使用できます。DB インスタンス にパブリック IP アドレスが割り当てられるかどうかは、ユーザーが制御します。

Amazon RDS は、パブリックにアクセス可能なデータベースインスタンスに EC2 のパブリック IPv4 アドレスプールからパブリック Elastic IPv4 アドレスを使用します。これらの IP アドレスは、`describe-addresses` CLI、API を使用するとき、または AWS マネジメントコンソールの Elastic IP (EIP) セクションを表示するときに、AWS アカウントに表示されます。各 RDS マネージド IP アドレスは、`service_managed` 属性が `"rds"` に設定された状態でマークされます。

これらの IP はアカウントに表示されますが、Amazon RDS によって完全に管理されたままであり、変更または解放することはできません。Amazon RDS は、使用されなくなった IP をパブリック IPv4 アドレスプールに解放します。

CloudTrail は、`AllocateAddress` などの RDS の EIP に関連する API コールをログに記録します。これらの API コールは、サービスプリンシパル `rds.amazonaws.com` によって呼び出されます。

> **Note**
>
> Amazon RDS によって割り当てられた IP は、アカウントの EIP 制限にはカウントされません。

一般的な Amazon RDS 状況で使用できるプライベート IPv4 アドレスのみで VPC を作成する方法のチュートリアルについては、「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。

IPv6 アドレス

オプションで IPv6 CIDR ブロックを VPC およびサブネットと関連付けて、そのブロックから VPC 内のリソースに IPv6 アドレスを割り当てることができます。各 IPv6 アドレスは、グローバルに一意です。

VPC の IPv6 CIDR ブロックは、Amazon の IPv6 アドレスのプールから自動的に割り当てられます。範囲を自分で選択することはできません。

IPv6 アドレスに接続するときには、以下の条件が満たされていることを確認してください。

- クライアントは、クライアントから IPv6 経由でのデータベーストラフィックが許可されるように構成されています。
- DB インスタンスによって使用される RDS セキュリティグループは、クライアントからデータベースへの IPv6 経由のトラフィックが許可されるように、正しく構成されています。
- クライアントのオペレーティングシステムスタックは IPv6 アドレス上のトラフィックを許可し、オペレーティングシステムドライバーとライブラリは、正しいデフォルトの DB インスタンスエンドポイント (IPv4 または IPv6) を選択するように構成されています。

IPv6 の詳細については、Amazon VPC ユーザーガイドの「IP アドレス指定」を参照してください。

デュアルスタックモード

DB インスタンスが IPv4 と IPv6 の両方のアドレス指定プロトコルで通信できるときには、デュアルスタックモードで実行します。その後、リソースは IPv4、IPv6、または両方のプロトコルを使用して DB インスタンスと通信できます。プライベートデュアルスタックモード DB インスタンスには、RDS が VPC アクセスのみに制限する IPv6 エンドポイントがあり、IPv6 エンドポイントはプライベートのままになります。パブリックデュアルスタックモード DB インスタンスは、インターネットからアクセスできる IPv4 エンドポイントと IPv6 エンドポイントの両方を提供します。

トピック

- デュアルスタックモードと DB サブネットグループ
- デュアルスタックモードの DB インスタンスの操作
- IPv4 専用 DB インスタンスをデュアルスタックモードを使用するように変更する
- リージョンとバージョンの可用性
- デュアルスタックネットワーク DB インスタンスの制限

一般的な Amazon RDS 状況で使用できる IPv4 と IPv6 の両方のアドレスを持つ VPC を作成する方法のチュートリアルについては、「チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード)」を参照してください。

デュアルスタックモードと DB サブネットグループ

デュアルスタックモードを使用するには、DB インスタンスに関連付ける DB サブネットグループ内の各サブネットに IPv6 CIDR ブロックが関連付けられていることを確認してください。新しい DB サブネットグループを作成するか、既存の DB サブネットグループを変更して、この要件を満たすことができます。DB インスタンスがデュアルスタックモードになった後も、クライアントは通常どおり接続できます。クライアントセキュリティファイアウォールと RDS DB インスタンスのセキュリティグループが、IPv6 経由のトラフィックを許可するように正しく設定されていることを確認します。接続するために、クライアントは DB インスタンスのエンドポイントを使用します。クライアントアプリケーションは、データベースへの接続時に優先するプロトコルを指定できます。デュアルスタックモードでは、DB インスタンスは、クライアントの優先ネットワークプロトコル (IPv4 または IPv6) を検出し、そのプロトコルを使用して接続します。

サブネットの削除または CIDR の関連付け解除により、DB サブネットグループがデュアルスタックモードをサポートしなくなった場合、DB サブネットグループに関連付けられている DB インスタンスに対して互換性のないネットワーク状態が発生するリスクがあります。また、新しいデュアルスタックモードの DB インスタンスの作成時に DB サブネットグループを使用することはできません。

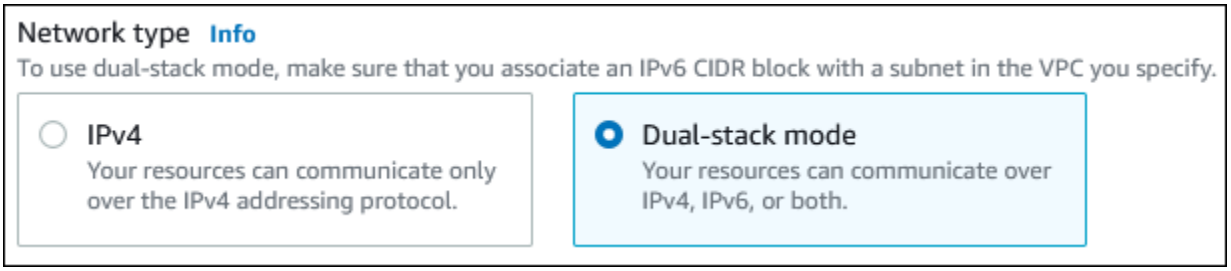
AWS マネジメントコンソール を使用して DB サブネットグループがデュアルスタックモードをサポートしているかどうかを判断するには、DB サブネットグループの詳細ページで [Network type] (ネットワークタイプ) を確認します。DB サブネットグループが AWS CLI を使用してデュアルスタックモードをサポートしているかどうかを判断するには、describe-db-subnet-groups コマンドを実行して、出力の `SupportedNetworkTypes` を確認します。

リードレプリカは独立した DB インスタンスとして扱われ、プライマリ DB インスタンスとは異なるネットワークタイプを持つことができます。リードレプリカのプライマリ DB インスタンスのネットワークタイプを変更しても、リードレプリカは影響を受けません。DB インスタンスを復元するときには、サポートされている任意のネットワークタイプに復元できます。

デュアルスタックモードの DB インスタンスの操作

DB インスタンスを作成または変更する場合は、デュアルスタックモードを指定して、リソースが DB インスタンスと IPv4、IPv6、またはその両方で通信することを許可できます。

AWS マネジメントコンソール を使用して DB インスタンスを変更するときには、[Network type] (ネットワークタイプ) セクションでデュアルスタックモードを指定できます。次の画像は、コンソールの [Network type] (ネットワークの種類) セクションを示しています。



AWS CLI を使用して DB インスタンスを作成または変更するときには、`--network-type` オプションを `DUAL` に設定して、デュアルスタックモードを使用します。RDS API を使用して DB インスタンスを作成または変更するときには、`NetworkType` パラメータを `DUAL` に設定して、デュアルスタックモードを使用します。DB インスタンスのネットワークタイプを変更すると、ダウンタイムが発生する可能性があります。指定された DB エンジンバージョンまたは DB サブネットグループでデュアルスタックモードがサポートされていない場合は、`NetworkTypeNotSupported` エラーが返されます。

DB インスタンスの作成の詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

コンソールを使用して DB インスタンスがデュアルスタックモードであるかどうかを判断するには、DB インスタンスの [Connectivity & security] (接続性とセキュリティ) タブの [Network type] (ネットワークの種類) を確認します。

IPv4 専用 DB インスタンスをデュアルスタックモードを使用するように変更する

IPv4 専用 DB インスタンスをデュアルスタックモードを使用するように変更できます。このためには、DB インスタンスのネットワークの種類を変更します。変更によってダウンタイムが発生する可能性があります。

メンテナンスウィンドウ中に Amazon RDS インスタンスのネットワークタイプを変更することをお勧めします。現在、新しいインスタンスのネットワークタイプをデュアルスタックモードに設定することはサポートされていません。ネットワークタイプは、`modify-db-instance` コマンドを使用して手動で設定できます。

DB インスタンスをデュアルスタックモードを使用するように変更する前に、その DB サブネットグループがデュアルスタックモードをサポートしていることを確認してください。DB インスタン

スに関連付けられた DB サブネットグループがデュアルスタックモードをサポートしていない場合は、DB インスタンスを変更するときに、それをサポートする別の DB サブネットグループを指定します。DB インスタンスの DB サブネットグループを変更すると、ダウンタイムが発生する可能性があります。

DB インスタンスをデュアルスタックモードを使用するように変更する前に DB インスタンスの DB サブネットグループを変更する場合は、変更の前後に DB サブネットグループが DB インスタンスに対して有効であることを確認してください。

RDS for PostgreSQL、RDS for MySQL、RDS for Oracle、RDS for MariaDB のシングル AZ インスタンスの場合、ネットワークをデュアルスタックに変更するには、`--network-type` パラメータに値 `DUAL` のみを設定して modify-db-instance コマンドを実行することをお勧めします。同じ API コールで `--network-type` パラメータと一緒に他のパラメータを追加すると、ダウンタイムが発生する可能性があります。複数のパラメータを変更する場合は、他のパラメータを使用して別の `modify-db-instance` リクエストを送信する前に、ネットワークタイプの変更が正常に完了していることを確認してください。

RDS for PostgreSQL、RDS for MySQL、RDS for Oracle、RDS for MariaDB のマルチ AZ インスタンスのネットワークタイプを変更する場合、`--network-type` パラメータのみを使用するか、modify-db-instance コマンドで複数のパラメータを組み合わせると、短いダウンタイムが発生し、フェイルオーバーがトリガーされます。

RDS for SQL Server シングル AZ インスタンスまたはマルチ AZ インスタンスでネットワークタイプを変更する場合、`--network-type` パラメータのみを使用するか、`modify-db-instance` コマンドで複数のパラメータを組み合わせると、ダウンタイムが発生します。ネットワークタイプを変更すると、SQL Server マルチ AZ インスタンスでフェイルオーバーが発生します。

変更後に DB インスタンスに接続できない場合は、選択したネットワーク (IPv4 または IPv6) でデータベースへのトラフィックを許可するように、クライアントとデータベースのセキュリティファイアウォールとルートテーブルが正確に設定されていることを確認してください。IPv6 アドレスを使用して接続するには、オペレーティングシステムのパラメータ、ライブラリ、またはドライバーを変更しなければならない場合もあります。

デュアルスタックモードを使用するように DB インスタンスを変更すると、シングル AZ 配置からマルチ AZ 配置、またはマルチ AZ 配置からシングル AZ 配置への変更を保留することはできません。

IPv4 専用の DB インスタンスをデュアルスタックモードを使用するように変更するには

1. DB サブネットグループをデュアルスタックモードをサポートするように変更するか、デュアルスタックモードをサポートする DB サブネットグループを作成します。

a. IPv6 CIDR ブロックと VPC の関連付け

詳細については、「Amazon VPC ユーザーガイド」の「IPv6 CIDR ブロックを VPC に追加する」を参照してください。

b. IPv6 CIDR ブロックを DB サブネットグループ内のすべてのサブネットにアタッチします。

詳細については、「Amazon VPC ユーザーガイド」の「IPv6 CIDR ブロックをサブネットに追加する」を参照してください。

c. DB サブネットグループがデュアルスタックモードをサポートしていることを確認します。

AWS マネジメントコンソール を使用している場合は、DB サブネットグループを選択し、[Supported network types] (サポートされているネットワークタイプ) の値が [Dual, IP v4] (デュアル、IPv4) であることを確認します。

AWS CLI を使用している場合は、describe-db-subnet-groups コマンドを実行して、DB インスタンスの `SupportedNetworkType` の値が `Dual, IPv4` であることを確認します。

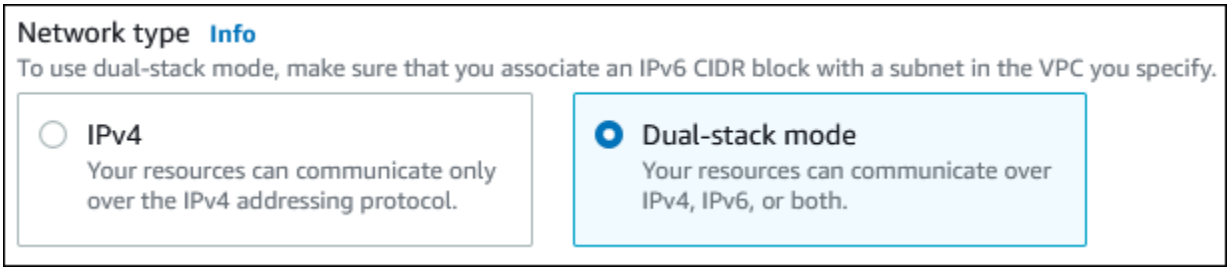
2. DB インスタンスに関連付けられたセキュリティグループを、データベースへの IPv6 接続を許可するように変更するか、IPv6 接続を許可する新しいセキュリティグループを作成します。

手順については、Amazon VPC ユーザーガイドの「セキュリティグループのルール」を参照してください。

3. DB インスタンスを変更して、デュアルスタックモードをサポートします。そのためには、ネットワークの種類をデュアルスタックモードに設定します。

コンソールを使用している場合、以下の設定が正しいことを確認します。

- [Network type] (ネットワークタイプ) – [Dual-stack mode] (デュアルスタックモード)



- [DB Subnet group] (DB サブネットグループ) — 前のステップで設定した DB サブネットグループ
- [Security group] (セキュリティグループ) - 前のステップで設定したセキュリティ

AWS CLI を使用している場合、以下の設定が正しいことを確認します。

- `--network-type` – `dual`
- `--db-subnet-group-name` - 前のステップで設定した DB サブネットグループ
- `--vpc-security-group-ids` - 前のステップで設定した VPC セキュリティグループ

例えば、次のようになります。

```
aws rds modify-db-instance --db-instance-identifier my-instance --network-type
 "DUAL"
```

4. DB インスタンスがデュアルスタックモードをサポートしていることを確認します。

   コンソールを使用している場合、DB インスタンスの [Connectivity & security] (接続とセキュリティ) (設定) タブを選択します。そのタブで、ネットワークの種類値がデュアルスタックモードであることを確認してください。

   AWS CLI を使用している場合は、describe-db-instances コマンドを実行して、DB インスタンスの `NetworkType` の値が `dual` であることを確認します。

   DB インスタンスエンドポイントで `dig` コマンドを実行して、関連付けられている IPv6 アドレスを特定します。

   ```
   dig db-instance-endpoint AAAA
   ```

   DB インスタンスに接続するには、IPv6 アドレスではなく DB インスタンスエンドポイントを使用します。

リージョンとバージョンの可用性

機能の可用性とサポートは、各データベースエンジンの特定のバージョン、および AWS リージョンによって異なります。デュアルスタックモードを利用できるバージョンとリージョンの詳細については、「Amazon RDS のデュアルスタックモードでサポートされているリージョンと DB エンジン」を参照してください。

デュアルスタックネットワーク DB インスタンスの制限

デュアルスタックネットワーク DB インスタンスには、次の制限が適用されます。

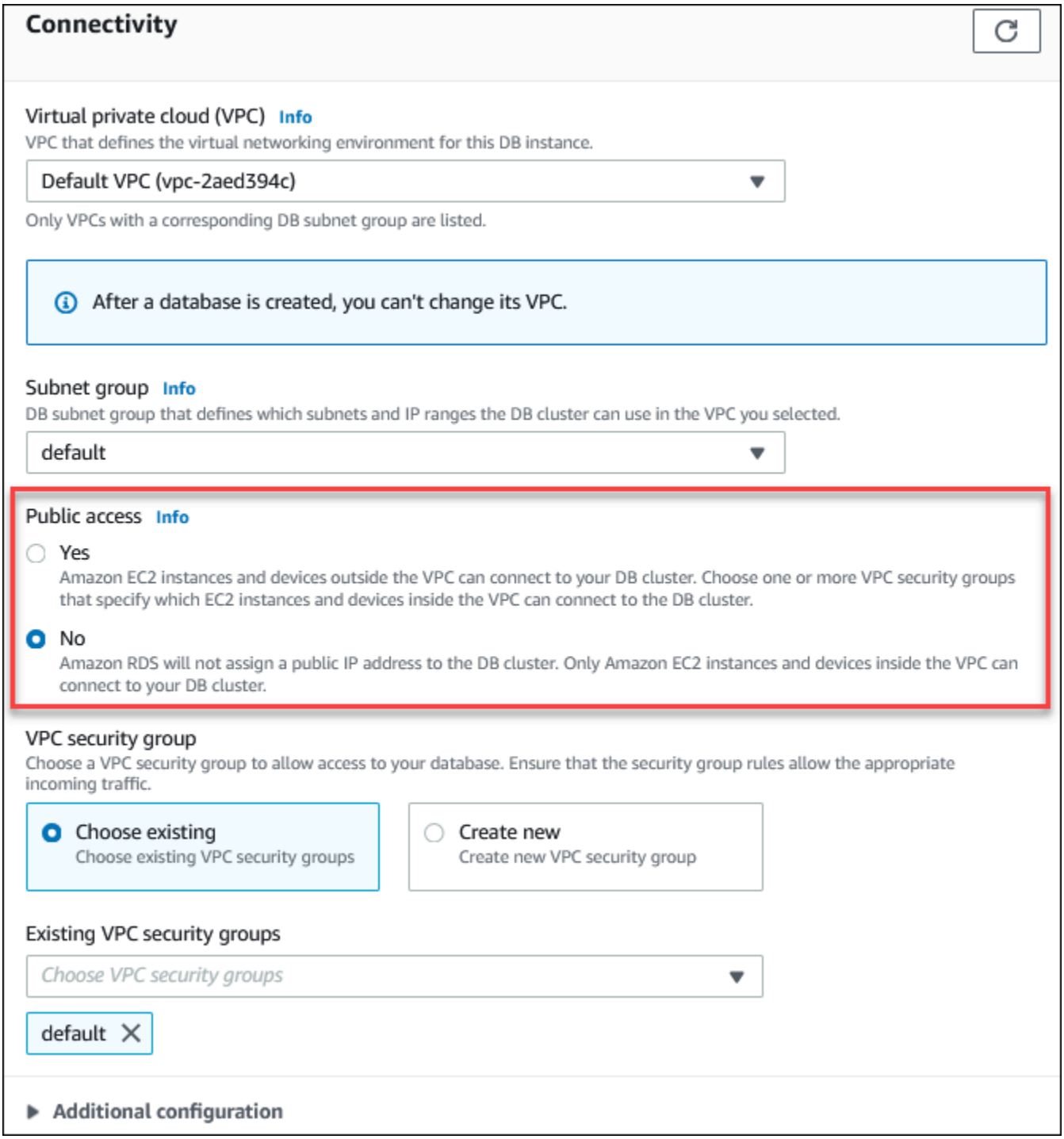
- DB インスタンスは、IPv6 プロトコルを排他的に使用することはできません。IPv4 を排他的に使用するか、IPv4 と IPv6 プロトコル を使用することができます (デュアルスタックモード)。
- Amazon RDS は、ネイティブ IPv6 サブネットをサポートしていません。
- RDS for SQL Server の場合、Always On AG アベイラビリティーグループリスナーエンドポイントを使用するデュアルスタックモード DB インスタンスは、IPv4 アドレスのみを示します。
- デュアルスタックモード DB インスタンスでは RDS Proxy を使用できません。
- AWS Outposts DB インスタンスではデュアルスタックモードを使用できません。
- ローカルゾーンでは、DB インスタンスでデュアルスタックモードを使用することはできません。

VPC 内の DB インスタンスをインターネットから隠す

Amazon RDS の一般的なシナリオの 1 つでは、一般向けウェブアプリケーションを使用する Amazon EC2 インスタンスと、パブリックアクセスが不可能なデータベースを使用する DB インスタンスがある VPC を想定しています。例えば、パブリックサブネットとプライベートサブネットを持つ VPC を作成できます。ウェブサーバーとして機能する EC2 インスタンスをパブリックサブネットにデプロイできます。DB インスタンスは、プライベートサブネットにデプロイされます。このような配置では、ウェブサーバーだけが DB インスタンスにアクセスできます。このシナリオの説明については、「同じ VPC 内の Amazon EC2 インスタンスがアクセスする、VPC 内の DB インスタンス」を参照してください。

VPC 内で DB インスタンスを起動すると、DB インスタンスには VPC 内のトラフィック用のプライベート IP アドレスが割り当てられます。このプライベート IP アドレスにはパブリックアクセスができません。パブリックアクセスオプションを使用すると、DB インスタンスがプライベート IP アドレスだけでなく、パブリック IP アドレスも保持するかどうかを指定できます。DB インスタンスがパブリックアクセスに指定されている場合、その DNS エンドポイントは VPC 内からプライベート IP アドレスに解決されます。VPC の外部からパブリック IP アドレスに解決されます。DB インスタンスへのアクセスは、最終的に使用されるセキュリティグループによって制御されます。DB インスタンスに割り当てられたセキュリティグループに、それを許可するインバウンドルールが含まれていない場合、そのパブリックアクセスは許可されません。また、内部ゲートウェイ DB インスタンスをパブリックにアクセス可能にするには、その DB サブネットグループのサブネットにインターネットゲートウェイが必要です。詳細については、「Amazon RDS DB インスタンスに接続できない」を参照してください。

パブリックアクセスオプションを変更することによって、DB インスタンスのパブリックアクセシビリティをオンまたはオフにすることができます。次の図は、[追加の接続設定] セクションの [パブリックアクセス] オプションを示しています。このオプションを設定するには、[接続] セクションの [追加の接続設定] セクションを開きます。



DB インスタンスを変更して [パブリックアクセス] オプションを設定する方法については、「Amazon RDS DB インスタンスを変更する」を参照してください。

VPC に DB インスタンスを作成する

次の手順で VPC 内に DB インスタンスを作成できます。デフォルトの VPC を使用する場合は、ステップ 2 から始めて、既に作成されている VPC と DB サブネットグループを使用することができます。VPC を追加で作成する場合は、VPC を新規に作成できます。

> **Note**
>
> VPC の DB インスタンスへのパブリックアクセスを可能にするには、VPC 属性の DNS hostnames と DNS resolution を有効化して、VPC に関する DNS 情報を更新する必要があります。VPC インスタンスの DNS 情報の更新については、「VPC の DNS サポートを更新する」を参照してください。

VPC 内に DB インスタンスを作成するには、以下のステップを実行します。

- ステップ 1: VPC を作成する
- ステップ 2: DB サブネットグループを作成する
- ステップ 3: VPC セキュリティグループを作成する
- ステップ 4: VPC に DB インスタンスを作成する

ステップ 1: VPC を作成する

最低 2 つのアベイラビリティーゾーンの中にサブネットを持つ VPC を作成します。これらのサブネットは、DB サブネットグループを作成するときに使用します。デフォルト VPC がある場合、AWS リージョン 内の各アベイラビリティーゾーンに、自動的にサブネットが作成されます。

詳細については、「プライベートサブネットおよびパブリックサブネットを持つ VPC を作成する」または Amazon VPC ユーザーガイドの「VPC を作成する」を参照してください。

ステップ 2: DB サブネットグループを作成する

DB サブネットグループは VPC 用に作成するサブネット (通常はプライベート) のコレクションで、DB インスタンスに指定します。DB サブネットグループを使用すると、AWS CLI または RDS API を使用して DB インスタンスを作成するときに、特定の VPC を指定できます。コンソールを使用する場合は、使用する VPC とサブネットを選択できます。各 DB サブネットグループには、AWS リージョン 内の少なくとも 2 つのアベイラビリティーゾーンに少なくとも 1 つのサブネットが必要

です。ベストプラクティスとして、各 DB サブネットグループには、AWS リージョン 内のアベイラビリティーゾーンごとに少なくとも 1 つのサブネットが必要です。

マルチ AZ 配置の場合、AWS リージョン のすべてのアベイラビリティーゾーンのサブネットを定義すると、Amazon RDS で必要に応じて別のアベイラビリティーゾーンに新しいスタンバイレプリカを作成できます。将来、マルチ AZ 配置に変換される可能性があるため、シングル AZ 配置でもこのベストプラクティスに従うことができます。

DB インスタンスをパブリックにアクセス可能にするには、DB サブネットグループのサブネットにインターネットゲートウェイが必要です。サブネット用のインターネットゲートウェイの詳細については、Amazon VPC ユーザーガイドの「インターネットゲートウェイを使用してサブネットをインターネットに接続する」を参照してください。

> **Note**
>
> ローカルゾーンの DB サブネットグループは、サブネットを 1 つだけ持つことができます。

VPC に DB インスタンスを作成するときに、DB サブネットグループを選択できます。Amazon RDS は、サブネットとそのサブネット内の IP アドレスを選択し、DB インスタンスに関連付けます。DB サブネットグループが存在しない場合、DB インスタンスを作成すると、Amazon RDS DB によってデフォルトのサブネットグループが作成されます。Amazon RDS では、Elastic Network Interface が作成され、その IP アドレスで DB インスタンスに関連付けられます。DB インスタンスは、そのサブネットを含むアベイラビリティーゾーンを使用します。

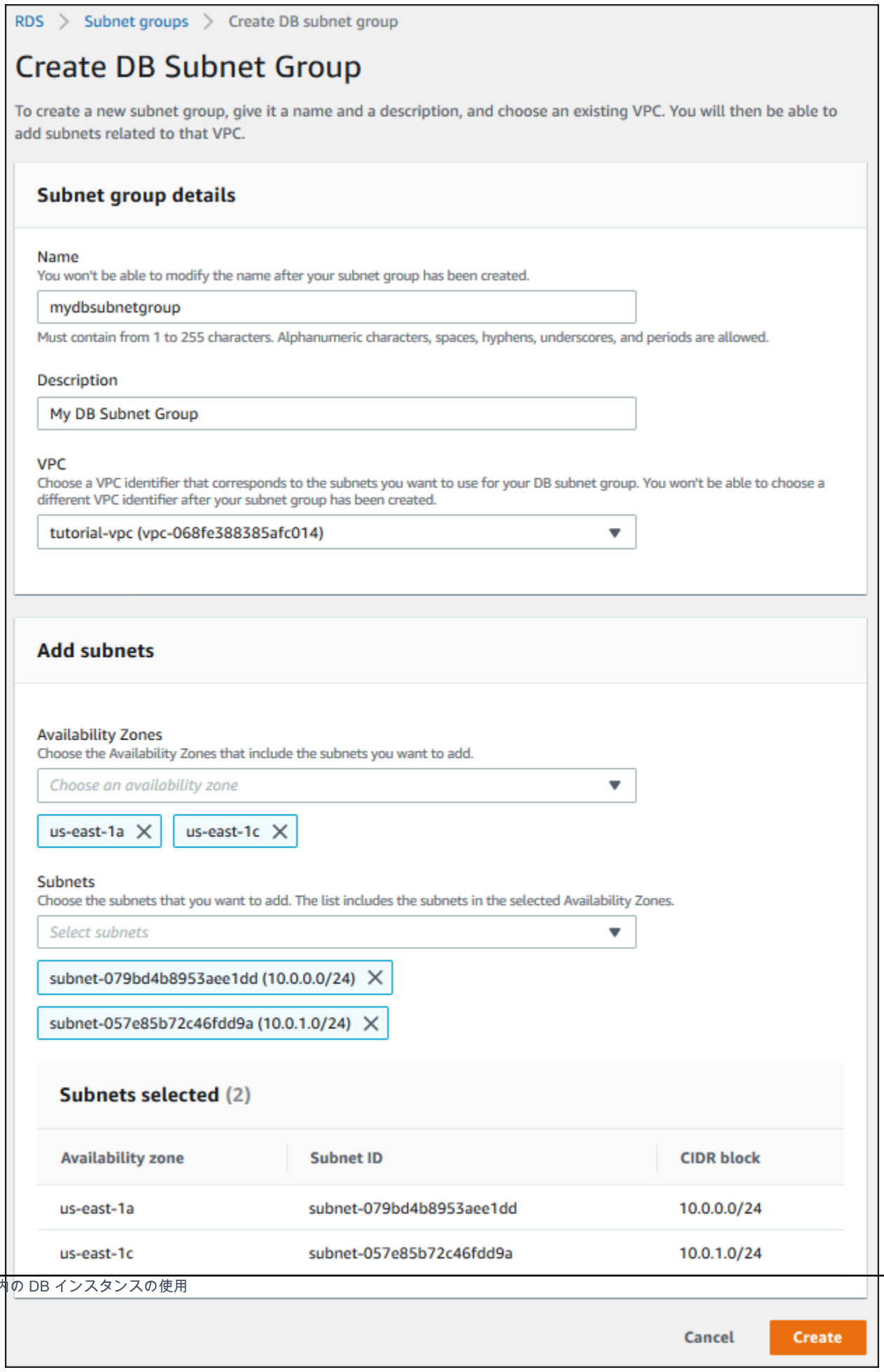
マルチ AZ 配置の場合、AWS リージョン 内の 2 つ以上のアベイラビリティーゾーンにサブネットを定義すると、Amazon RDS は必要に応じて別のアベイラビリティーゾーンに新しいスタンバイを作成できるようになります。シングル AZ 配置の場合も、どこかの時点でマルチ AZ 配置に変換する場合に備えてこのように定義する必要があります。

このステップでは、DB サブネットグループを作成し、このグループに VPC 用に作成したサブネットを追加します。

DB サブネットグループを作成する方法

1. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. [Navigation] ペインで、[Subnet groups] を選択します。
3. [Create DB Subnet Group] を選択します。

4. [Name] には、DB サブネットグループの名前を入力します。
5. [Description] に、DB サブネットグループの説明を入力します。
6. [VPC] では、デフォルトの VPC または作成した VPC を選択します。
7. [サブネットの追加] セクションで、サブネットを含むアベイラビリティーゾーンを [アベイラビリティーゾーン] から選択し、サブネットを [サブネット] から選択します。


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

> **Note**
>
> ローカルゾーンを有効にしている場合は、[DB サブネットグループの作成] ページでアベイラビリティーゾーングループを選択できます。この場合、[アベイラビリティーゾーングループ]、[アベイラビリティーゾーン]、[サブネット] の順に選択します。

8. [作成] を選択します。

   RDS コンソールの DB サブネットグループリストに新しい DB サブネットグループが表示されます。DB サブネットグループを選択すると、詳細ペインに、そのグループに関連付けられたすべてのサブネットなどの詳細を表示できます。

ステップ 3: VPC セキュリティグループを作成する

DB インスタンスを作成する前に、DB インスタンスに関連付ける VPC セキュリティグループを作成する必要があります。VPC セキュリティグループを作成しない場合、DB インスタンスを作成するときにデフォルトのセキュリティグループを使用します。DB インスタンスのセキュリティグループを作成する方法については、「プライベート DB インスタンスの VPC セキュリティグループを作成する」を参照するか、Amazon VPC ユーザーガイドの「セキュリティグループを使用してリソースへのトラフィックを制御する」を参照してください。

ステップ 4: VPC に DB インスタンスを作成する

このステップでは、DB インスタンスを作成し、前のステップで作成した VPC 名、DB サブネットグループ、および VPC セキュリティグループを使用します。

> **Note**
>
> VPC の DB インスタンスをパブリックにアクセス可能にする場合は、VPC 属性の DNS hostnames と DNS resolution を有効にする必要があります。詳細については、Amazon VPC ユーザーガイドの「DNS attributes for your VPC」(VPC の DNS 属性) を参照してください。

DB インスタンスの作成方法の詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。

[Connectivity] (接続) セクションにプロンプトが表示されたら、VPC の名前、DB サブネットグループ、および VPC セキュリティグループを入力します。

DB インスタンスの VPC の更新

AWS マネジメントコンソールを使用して DB インスタンスを別の VPC に移動できます。

DB インスタンスの変更については、Amazon RDS DB インスタンスを変更するを参照してください。次のように、変更ページの [Connectivity] (接続) セクションが表示されたら、[DB Subnet group] (DB サブネットグループ) に新しい DB サブネットグループを入力します。新しいサブネットグループは新しい VPC のサブネットグループである必要があります。

次の条件が適用される場合、DB インスタンスの VPC を変更することはできません。

- DB インスタンスが複数のアベイラビリティーゾーンに置かれている。DB インスタンスは、単一のアベイラビリティーゾーンに変換し、新しい VPC に移動した後に、マルチ AZ DB インスタンスに戻すことができます。詳細については、「Amazon RDS でのマルチ AZ 配置の設定と管理」を参照してください。
- DB インスタンスに 1 つ以上のリードレプリカがある。リードレプリカを削除し、DB インスタンスを新しい VPC に移動した後、リードレプリカを再度追加することができます。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。
- DB インスタンスがリードレプリカである。リードレプリカをスタンドアロンの DB インスタンスに昇格し、それを新しい VPC に移動することができます。詳細については、「リードレプリカをスタンドアロン DB インスタンスに昇格させる」を参照してください。
- ターゲット VPC のサブネットグループが、DB インスタンスのアベイラビリティーゾーン内にサブネットを持っていない。DB インスタンスのアベイラビリティーゾーンのサブネットを、DB サブネットグループに追加した後に、DB インスタンスを新しい VPC に移動することができます。詳細については、「DB サブネットグループの使用」を参照してください。

VPC の DB インスタンスにアクセスするシナリオ

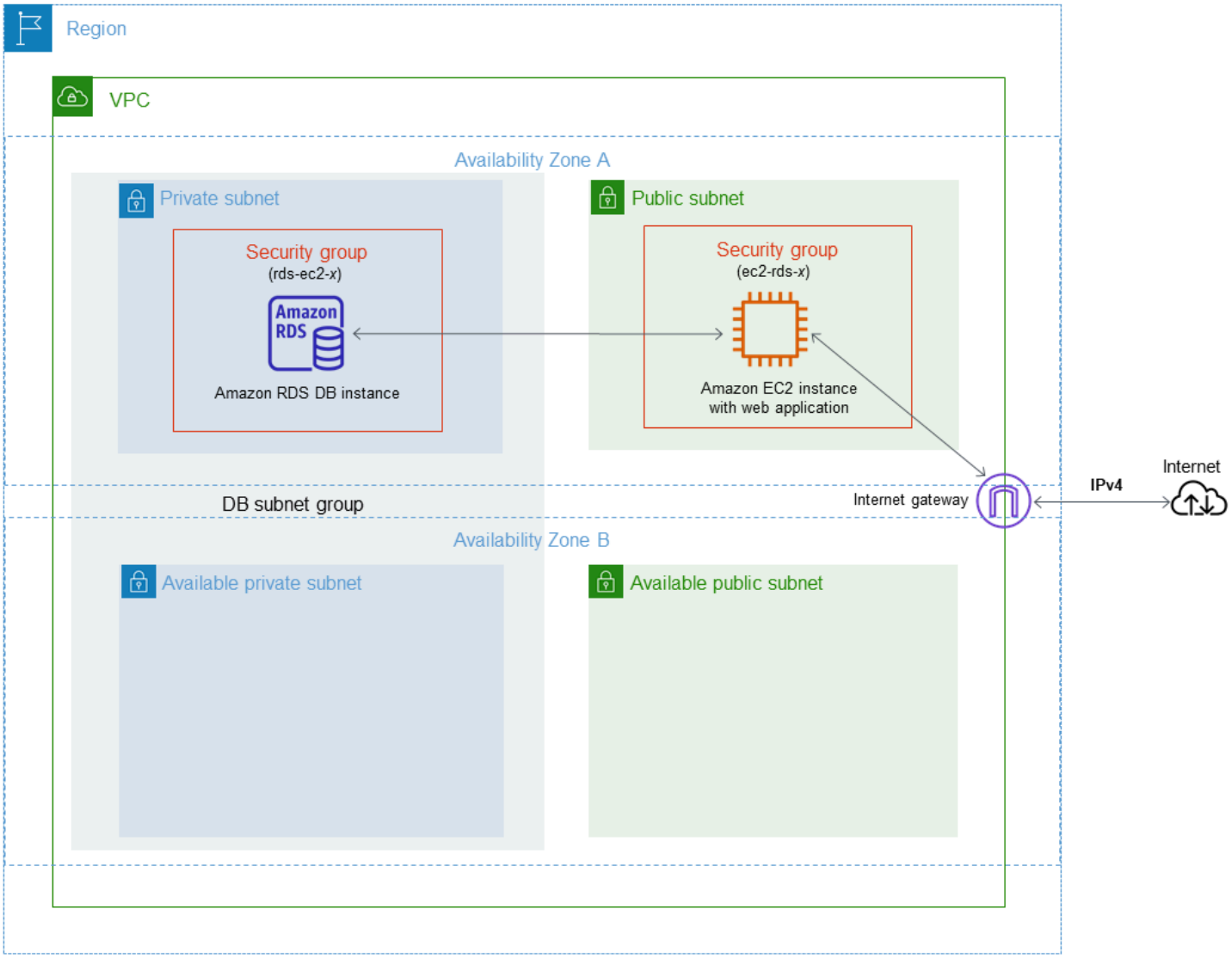
Amazon RDS は、VPC の DB インスタンスにアクセスするための以下のシナリオをサポートします。

- 同じ VPC 内の Amazon EC2 インスタンス
- 別の VPC 内の EC2 インスタンス
- インターネット経由のクライアントアプリケーション
- プライベートネットワーク

同じ VPC 内の Amazon EC2 インスタンスがアクセスする、VPC 内の DB インスタンス

VPC 内の DB インスタンスの一般的な用途は、同じ VPC 内の Amazon EC2 インスタンスで実行されるアプリケーションサーバーとデータを共有することです。

以下の図に、このシナリオを示しています。



同じ VPC 内の EC2 インスタンスと DB インスタンス間のアクセスを管理する方法として最も簡単なのは、次の方法です。

- DB インスタンスが存在する VPC セキュリティグループを作成します。このセキュリティグループは、DB インスタンスへのアクセスを制限するのに使用できます。たとえば、このセキュリティグループのカスタムルールを作成できます。これにより、DB インスタンスを作成したときに割り当てたポートと、開発またはそのほかの目的で DB インスタンスにアクセスするのに使用する IP アドレスを使用して TCP へのアクセスを許可できます。
- EC2 インスタンス (ウェブサーバーとクライアント) が属する VPC セキュリティグループを作成します。このセキュリティグループは、必要に応じて、VPC のルーティングテーブルを介したインターネットから EC2 インスタンスへのアクセスを許可できます。例えば、ポート 22 経由で

EC2 インスタンスへの TCP アクセスを許可するルールをこのセキュリティグループに設定できます。

- EC2 インスタンス用に作成したセキュリティグループからの接続を許可する DB インスタンスのセキュリティグループで、カスタムルールを作成します。このルールは、セキュリティグループのメンバーに DB インスタンスへのアクセスを許可します。

別のアベイラビリティーゾーンに、追加のパブリックサブネットとプライベートサブネットがあります。RDS DB サブネットグループには、2 つ以上のアベイラビリティーゾーンにサブネットが必要です。サブネットが追加されたことで、将来的にマルチ AZ DB インスタンス配置に簡単に切り替えることができるようになります。

このシナリオのパブリックとプライベートの両方のサブネットを使用する VPC を作成する方法のチュートリアルについては、「チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)」を参照してください。

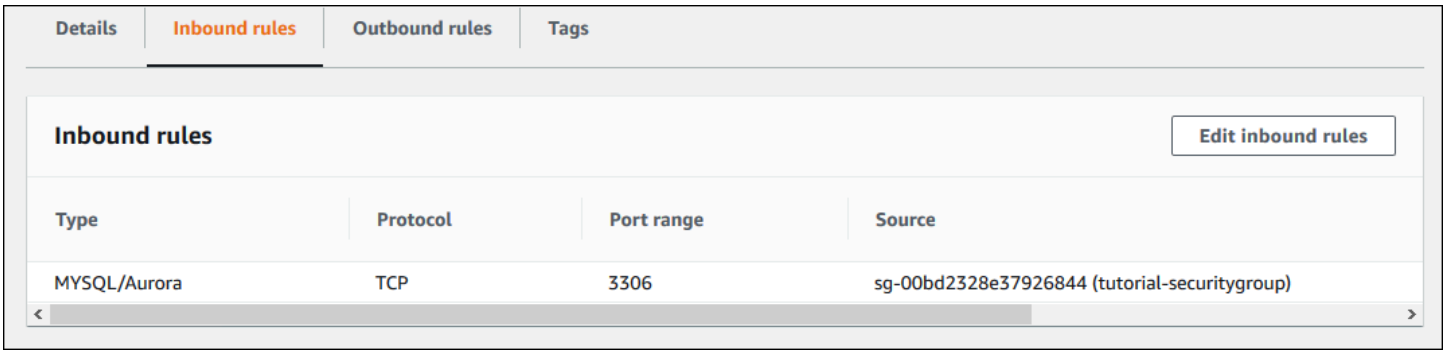
> **Tip**
>
> DB インスタンスを作成すると自動的に、Amazon EC2 インスタンスと DB インスタンス間のネットワーク接続を設定できるようになります。詳細については、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。

他のセキュリティグループからの接続を許可する VPC セキュリティグループにルールを作成するには、以下を実行します。

1. AWS マネジメントコンソール にサインインして、Amazon VPC コンソール (https://console.aws.amazon.com/vpc) を開きます。
2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. 他のセキュリティグループのメンバーからのアクセスを許可するセキュリティグループを、選択または作成します。前述のシナリオで、これは DB インスタンス向けに使用するセキュリティグループです。[インバウンドルール] タブを選択してから、[インバウンドルールの編集] を選択します。
4. [インバウンドルールの編集] ページで、[ルールの追加] を選択します。
5. [Type] (タイプ) から、DB インスタンスの作成時に使用したポートに対応するエントリ ([MYSQL/Aurora] など) を選択します。

6. [ソース] ボックスで、セキュリティグループの ID の入力をスタートすると、一致するセキュリティグループが一覧表示されます。このセキュリティグループによって保護されているリソースへのアクセスを許可するメンバーが所属しているセキュリティグループを選択します。前述のシナリオで、これは EC2 インスタンス向けに使用するセキュリティグループです。
7. 必要に応じて、[タイプ] に [すべての TCP] を、[ソース] ボックスにお客様のセキュリティグループを指定してルールを作成することで、TCP プロトコルのステップを繰り返します。UDP プロトコルを使用する場合は、[All UDP] (すべての UDP) を [Type] (タイプ) と [Source] (送信元) のセキュリティグループとして使用してルールを作成します。
8. [ルールの保存] を選択します。

次の画面には、ソース用のセキュリティグループを含むインバウンドルールが表示されます。

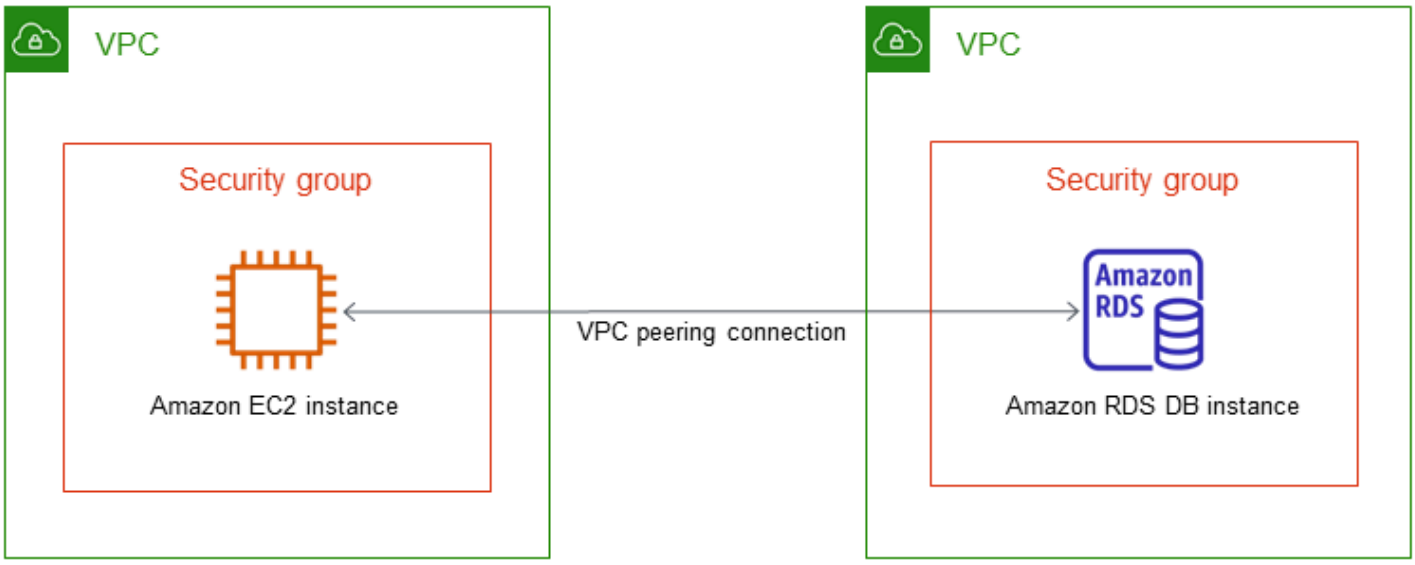


EC2 インスタンスから DB インスタンスに接続する方法の詳細については、「Amazon RDS DB インスタンスへの接続」を参照してください。

VPC 内の DB インスタンスに別の VPC 内の EC2 インスタンスからアクセスする

DB インスタンスがアクセスに使用している EC2 インスタンスとは異なる VPC にある場合、VPC ピア接続を使用してその DB インスタンスにアクセスできます。

以下の図に、このシナリオを示しています。

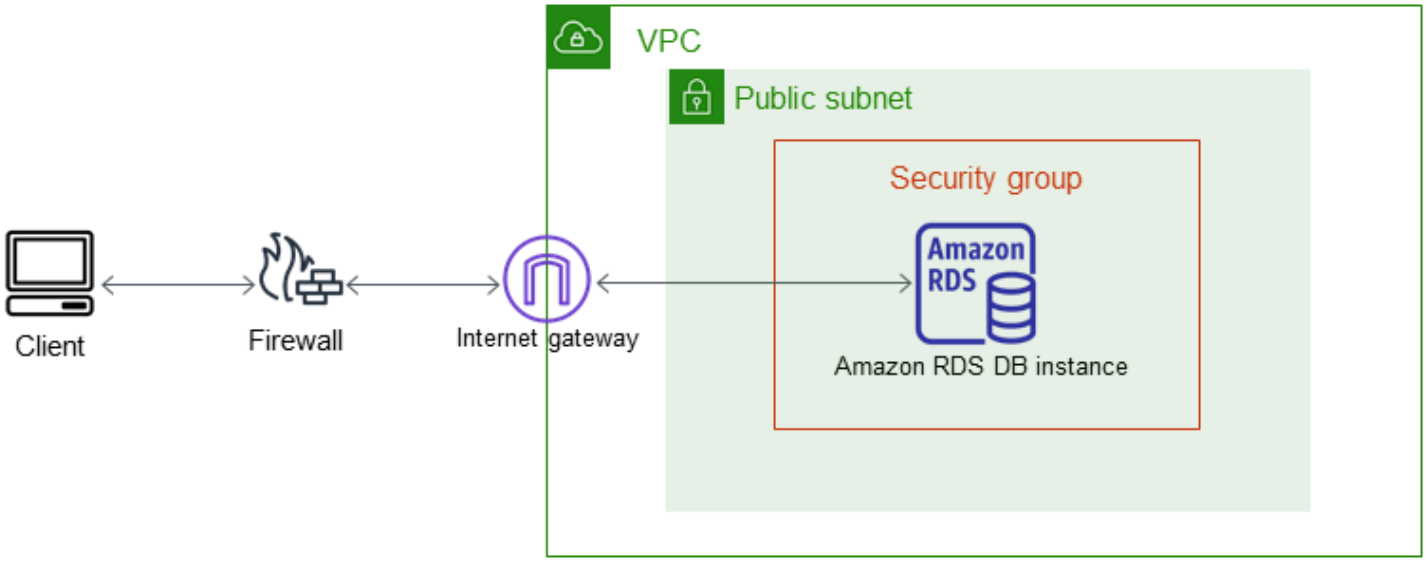


VPC ピア接続は、プライベート IP アドレスを使用して 2 つの VPC 間でトラフィックをルーティングすることを可能にするネットワーク接続です。どちらの VPC のリソースも、同じネットワーク内に存在しているかのように、相互に通信できます。VPC ピアリング接続は、自分の VPC 間、別の AWS アカウントの VPC との間、または別の AWS リージョン の VPC との間に作成できます。VPC ピア接続の詳細については、Amazon Virtual Private Cloud ユーザーガイドの「VPC ピア接続」を参照してください。

インターネット経由でクライアントアプリケーションから VPC 内の DB インスタンスにアクセスする

インターネット経由でクライアントアプリケーションから VPC 内の DB インスタンスにアクセスするには、1 つのパブリックサブネットを持つ VPC と、インターネットを介した通信を可能にするインターネットゲートウェイを設定します。

以下の図に、このシナリオを示しています。



次の構成をお勧めします。

- サイズ /16 (例えば CIDR: 10.0.0.0/16) の VPC。このサイズでは 65,536 個のプライベート IP アドレスが提供されます。
- サイズ /24 (例えば CIDR: 10.0.0.0/24) のサブネット。このサイズでは 256 個のプライベート IP アドレスが提供されます。
- VPC およびサブネットに関連付けられている Amazon RDS DB インスタンス。Amazon RDS は、サブネット内の IP アドレスを DB インスタンスに割り当てます。
- VPC をインターネットと他の AWS 製品に接続するインターネットゲートウェイ。
- DB インスタンスに関連付けられたセキュリティグループ。セキュリティグループのインバウンドルールにより、クライアントアプリケーションは DB インスタンスにアクセスできます。

VPC での DB インスタンスの作成方法に関する詳細は、「VPC に DB インスタンスを作成する」を参照してください。

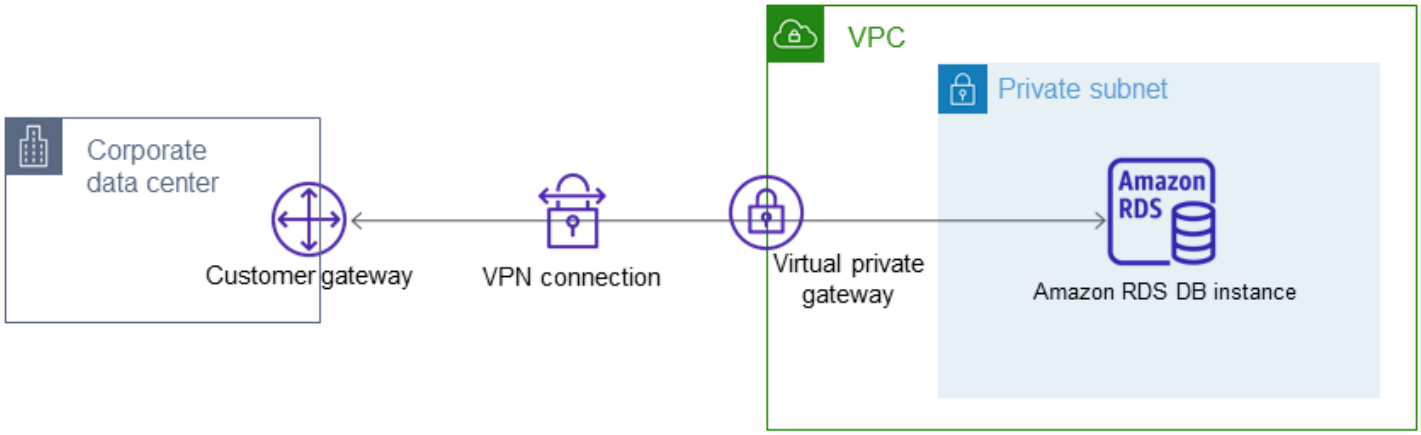
プライベートネットワークによってアクセスされる VPC 内の DB インスタンス

DB インスタンスがパブリックにアクセスできない場合は、プライベートネットワークからアクセスするための次のオプションがあります。

- AWS Site-to-Site VPN 接続。詳細については、「AWS Site-to-Site VPN とは」を参照してください。
- Direct Connect 接続。詳細については、「Direct Connect とは?」を参照してください。

- AWS Client VPN 接続。詳細については、「AWS Client VPN とは」を参照してください。

次の図は、AWS Site-to-Site VPN 接続のシナリオを示しています。

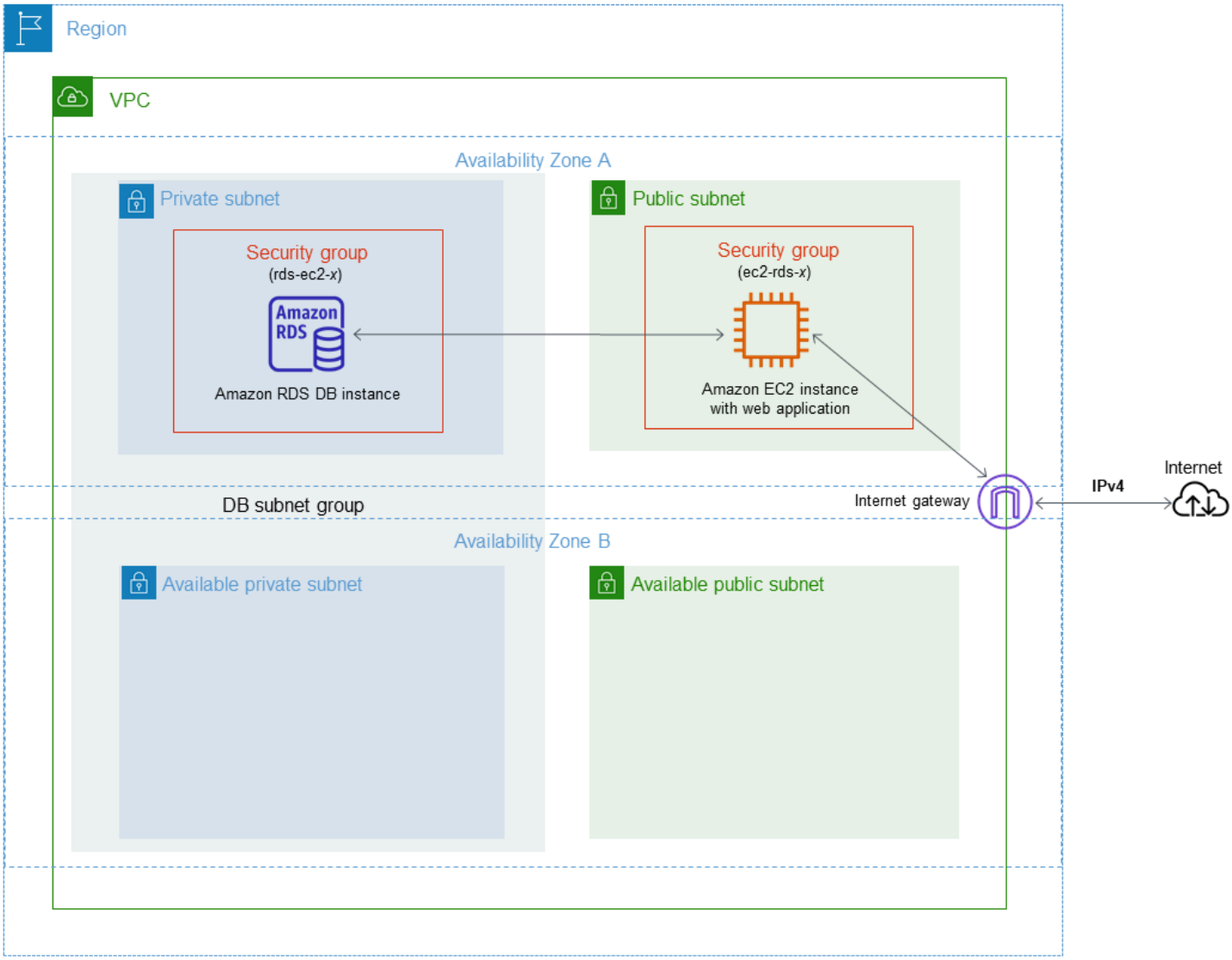


詳細については、「ネットワーク間のトラフィックのプライバシー」を参照してください。

チュートリアル: DB インスタンスで使用する VPC を作成する (IPv4 専用)

一般的なシナリオには、Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) 内の DB インスタンスが含まれます。この VPC は、同じ VPC で実行しているウェブサーバーとデータを共有します。このチュートリアルでは、このシナリオの VPC を作成します。

以下の図に、このシナリオを示しています。その他のシナリオについては、VPC の DB インスタンスにアクセスするシナリオ を参照してください。



DB インスタンスは、ウェブサーバーからのみ使用可能で、パブリックインターネットからは使用できないようにする必要があります。したがって、パブリックサブネットとプライベートサブネットを持つ VPC を作成します。ウェブサーバーはパブリックサブネットでホストされることで、パブリックインターネットにアクセスできます。DB インスタンスはプライベートサブネットでホストされます。ウェブサーバーは、同じ VPC 内でホストされているため、DB インスタンスに接続できます。

ただし、DB インスタンスはパブリックインターネットからは使用できないため、セキュリティが向上します。

このチュートリアルでは、別のアベイラビリティーゾーンに追加のパブリックサブネットとプライベートサブネットを設定します。これらのサブネットはチュートリアルでは使用されません。RDS DB サブネットグループは、少なくとも 2 つのアベイラビリティーゾーン内のサブネットを必要とします。サブネットが追加されたことで、将来的にマルチ AZ DB インスタンス配置に簡単に切り替えることができるようになります。

このチュートリアルでは、Amazon RDS DB インスタンス 用に VPC を設定する方法について説明します。この VPC シナリオ用のウェブサーバーを作成する方法を示すチュートリアルについては、「チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する」を参照してください。Amazon VPC の詳細については、Amazon VPC 入門ガイドおよびAmazon VPC ユーザーガイドを参照してください。

> **Tip**
>
> DB インスタンスを作成すると自動的に、Amazon EC2 インスタンスと DB インスタンス間のネットワーク接続を設定できるようになります。ネットワーク構成は、このチュートリアルで説明したものと似ています。詳細については、「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。

プライベートサブネットおよびパブリックサブネットを持つ VPC を作成する

以下の手順で、パブリックサブネットとプライベートサブネットを持つ VPC を作成します。

VPC とサブネットを作成するには

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. AWS マネジメントコンソールで、VPC を作成するリージョンを選択します。この例では、米国西部 (オレゴン) リージョンを使用します。
3. [VPC ダッシュボード] を選択します。VPC の作成を開始するには、[Create VPC] (VPC の作成) を選択します。
4. [VPC Settings] (VPC 設定) の [Resources to create] (作成するリソース) で、[VPC and more] (VPC など) を選択します。
5. [VPC settings] (VPC 設定) で、これらの値を設定します。

- [Name tag auto-generation] (ネームタグ自動生成) – **tutorial**
- [IPv4 CIDR block] (IPv4 CIDR ブロック) – **10.0.0.0/16**
- [IPv6 CIDR block] (IPv6 CIDR ブロック) – [No IPv6 CIDR block] (IPv6 CIDR ブロックなし)
- [Tenancy] (テナンシー) – デフォルト
- [Number of Availability Zones (AZs)] (アベイラビリティーゾーンの数 (AZ)) – 2
- [Customize AZs] (AZ をカスタマイズする) – デフォルト値を維持します。
- [Number of public subnet] (パブリックサブネット数) – 2
- [Number of private subnets] (プライベートサブネット数) – 2
- [Customize subnets CIDR blocks] (サブネット CIDR ブロックをカスタマイズ) — デフォルト値を維持します。
- [NAT gateways ($)] (NAT ゲートウェイ ($)) – なし
- [VPC endpoints] (VPC エンドポイント) – なし
- [DNS options] (DNS オプション) — デフォルト値を維持します。

> **Note**
>
> Amazon RDS では、マルチ AZ DB インスタンス配置をサポートするために、2 つの異なるアベイラビリティーゾーン内のサブネットを少なくとも 2 つ含んでいる必要があります。このチュートリアルではシングル AZ 配置を作成しますが、この要件により将来的に マルチ AZ DB インスタンス配備に簡単に変換できます。

6. [Create VPC (VPC の作成)] を選択します。

パブリックウェブサーバーの VPC セキュリティグループを作成する

次に、パブリックアクセスのためのセキュリティグループを作成します。VPC 内のパブリック EC2 インスタンスに接続するには、インバウンドルールを VPC セキュリティグループに追加します。これにより、インターネットからのトラフィックを接続できるようになります。

VPC セキュリティグループを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. [VPC ダッシュボード]、[セキュリティグループ]、[セキュリティグループの作成] の順に選択します。

3. [セキュリティグループの作成] ページで、以下の値を設定します。

   - セキュリティグループ名: **tutorial-securitygroup**
   - 説明: **Tutorial Security Group**
   - [VPC ID]: 前に作成した VPC を選択します (例: [vpc-***identifier*** (tutorial-vpc)])。

4. インバウンドルールをセキュリティグループに追加します。

   a. Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。パブリック IP アドレスを決定するには、別のブラウザウィンドウまたはタブで、https://checkip.amazonaws.com のサービスを使用できます。IP アドレスの例は `203.0.113.25/32` です。

      多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。この場合は、クライアントコンピュータが使用する IP アドレスの範囲を検索します。

      > **Warning**
      >
      > SSH アクセスに `0.0.0.0/0` を使用すると、すべての IP アドレスが SSH を使ってパブリックインスタンスにアクセスできるようになります。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、SSH を使ったインスタンスへのアクセスを限定します。

   b. [インバウンドルール] セクションで、[ルールの追加] を選択します。

   c. 新しいインバウンドルールに次の値を設定して、Amazon EC2 インスタンスへの SSH アクセスを許可します。こうすることで、Amazon EC2 インスタンスに接続して、ウェブサーバーなどのユーティリティをインストールできます。また、EC2 インスタンスに接続して、ウェブサーバー用のコンテンツをアップロードします。

      - タイプ: **SSH**
      - ソース: ステップ a で指定した IP アドレスまたはアドレス範囲 (**203.0.113.25/32** など)

   d. [ルールの追加] を選択します。

   e. 新しいインバウンドルールに次の値を設定して、ウェブサーバーに HTTP へのアクセスを許可します。

- 型: **HTTP**
- ソース: **0.0.0.0/0**

5. セキュリティグループを作成するには、[Create security group] (セキュリティグループの作成) を選択します。

   セキュリティグループ ID を書き留めます。このチュートリアルで後に必要になります。

プライベート DB インスタンスの VPC セキュリティグループを作成する

DB インスタンスをプライベートのままにするには、プライベートアクセス用の第 2 のセキュリティグループを作成します。VPC 内のプライベート DB インスタンスに接続するには、ウェブサーバーからのトラフィックのみを許可するインバウンドルールを VPC セキュリティグループに追加します。

VPC セキュリティグループを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. [VPC ダッシュボード]、[セキュリティグループ]、[セキュリティグループの作成] の順に選択します。
3. [セキュリティグループの作成] ページで、以下の値を設定します。
   - セキュリティグループ名: **tutorial-db-securitygroup**
   - 説明: **Tutorial DB Instance Security Group**
   - [VPC ID]: 前に作成した VPC を選択します (例: [vpc-***identifier*** (tutorial-vpc)])。
4. インバウンドルールをセキュリティグループに追加します。
   a. [インバウンドルール] セクションで、[ルールの追加] を選択します。
   b. 新しいインバウンドルールに次の値を設定して、Amazon EC2 インスタンスからポート 3306 への MySQL トラフィックを許可します。これを実行すると、ウェブサーバーから DB インスタンスに接続できます。そうすることで、ウェブアプリケーションからのデータをデータベースに保存および取得できるようになります。
      - 型: **MySQL/Aurora**
      - [Source] (ソース): このチュートリアルで以前に作成した tutorial-securitygroup セキュリティグループの ID (例: sg-9edd5cfb)。

5. セキュリティグループを作成するには、[Create security group] (セキュリティグループの作成) を選択します。

DB サブネットグループを作成する

DB サブネットグループは VPC に作成するサブネットのコレクションで、DB インスタンス用に指定します。DB サブネットグループでは、DB インスタンスの作成時に特定の VPC を指定することができます。

DB サブネットグループを作成するには

1. VPC 内のデータベースのプライベートサブネットを特定します。

   a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
   b. [VPC Dashboard] (VPC ダッシュボード) を選択してから、[Subnets] (サブネット) を選択します。
   c. tutorial-subnet-private1-us-west-2a と tutorial-subnet-private2-us-west-2b という名前のサブネット ID に注意してください。

      DB サブネットグループを作成するときに、サブネット ID が必要です。
2. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

   Amazon VPC コンソールではなく、Amazon RDS コンソールに接続してください。
3. [Navigation] ペインで、[Subnet groups] を選択します。
4. [Create DB subnet group] (DB サブネットグループの作成) を選択します。
5. [DB サブネットグループを作成する] ページで、[サブネットグループの詳細] に値を設定します。

   - 名前: **tutorial-db-subnet-group**
   - 説明: **Tutorial DB Subnet Group**
   - VPC: tutorial-vpc (vpc-***identifier***)
6. [サブネットの追加] セクションで、[アベイラビリティーゾーン] と [サブネット] を選択します。

   このチュートリアルでは、[Availability Zones] (アベイラビリティーゾーン) として [us-west-2a] と [us-west-2b] を選択します。[Subnets] (サブネット) では、前のステップで特定したプライベートサブネットを選択します。
7. [作成] を選択します。

RDS コンソールの DB サブネットグループリストに新しい DB サブネットグループが表示されます。DB サブネットグループを選択すると、詳細ペインに詳細が表示されます。これらの詳細には、グループに関連付けられているすべてのサブネットが含まれます。

> **Note**
>
> この VPC を作成して チュートリアル: ウェブサーバーと Amazon RDS DB インスタンスを作成する を完了した場合は、Amazon RDS DB インスタンスを作成する の手順に従って DB インスタンスを作成します 。

VPC の削除

このチュートリアルの VPC およびその他のリソースを作成後、不要になった場合は、削除できます。

> **Note**
>
> このチュートリアルで作成した VPC にリソースを追加した場合は、VPC を削除する前にこれらを削除しなければならない場合があります。例えば、これらのリソースには Amazon EC2 インスタンスや Amazon RDS DB インスタンス が含まれる場合があります。詳細については、「Amazon VPC ユーザーガイド」の「VPC の削除」を参照してください。

VPC と関連リソースを削除する方法

1. DB サブネットグループを削除する。
   a. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
   b. [ナビゲーション] ペインで、[サブネットグループ] を選択します。
   c. 削除する DB サブネットグループを選択します。(例: tutorial-db-subnet-group)
   d. [Delete] (削除) を選択してから、確認ウィンドウの [Delete] (削除) を選択します。
2. VPC ID を書き留める。
   a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
   b. [VPC ダッシュボード] を選択してから、[VPC] を選択します。

c. リストで、作成した VPC を特定します。(例: tutorial-vpc)

d. 作成した VPC の [VPC ID] をメモします。後続のステップで VPC ID が必要になります。

3. セキュリティグループを削除する。

a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。

b. [VPC Dashboard] (VPC ダッシュボード) を選択してから、[Security Groups] (セキュリティグループ) を選択します。

c. Amazon RDS DB インスタンスのセキュリティグループを選択します。(例: tutorial-db-securitygroup)

d. [Actions] (アクション) で、[Delete security groups] (セキュリティグループの削除) を選択してから、確認ページで [Delete] (削除) を選択します。

e. [Security Groups] (セキュリティグループ) ページで、Amazon EC2 インスタンスのセキュリティグループを選択します。(例: tutorial-securitygroup)

f. [Actions] (アクション) で、[Delete security groups] (セキュリティグループの削除) を選択してから、確認ページで [Delete] (削除) を選択します。

4. VPC を削除する。

a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。

b. [VPC Dashboard] (VPC ダッシュボード) を選択してから、[VPC] を選択します。

c. 削除する VPC を選択します。(例: tutorial-vpc)

d. [アクション] で、[VPC の削除] を選択します。

確認ページには、VPC に関連付けられたサブネットを含め、削除される VPC に関連付けられているその他のリソースが表示されます。

e. 確認ページで、「**`delete`**」を入力してから、[Delete] (削除) を選択します。

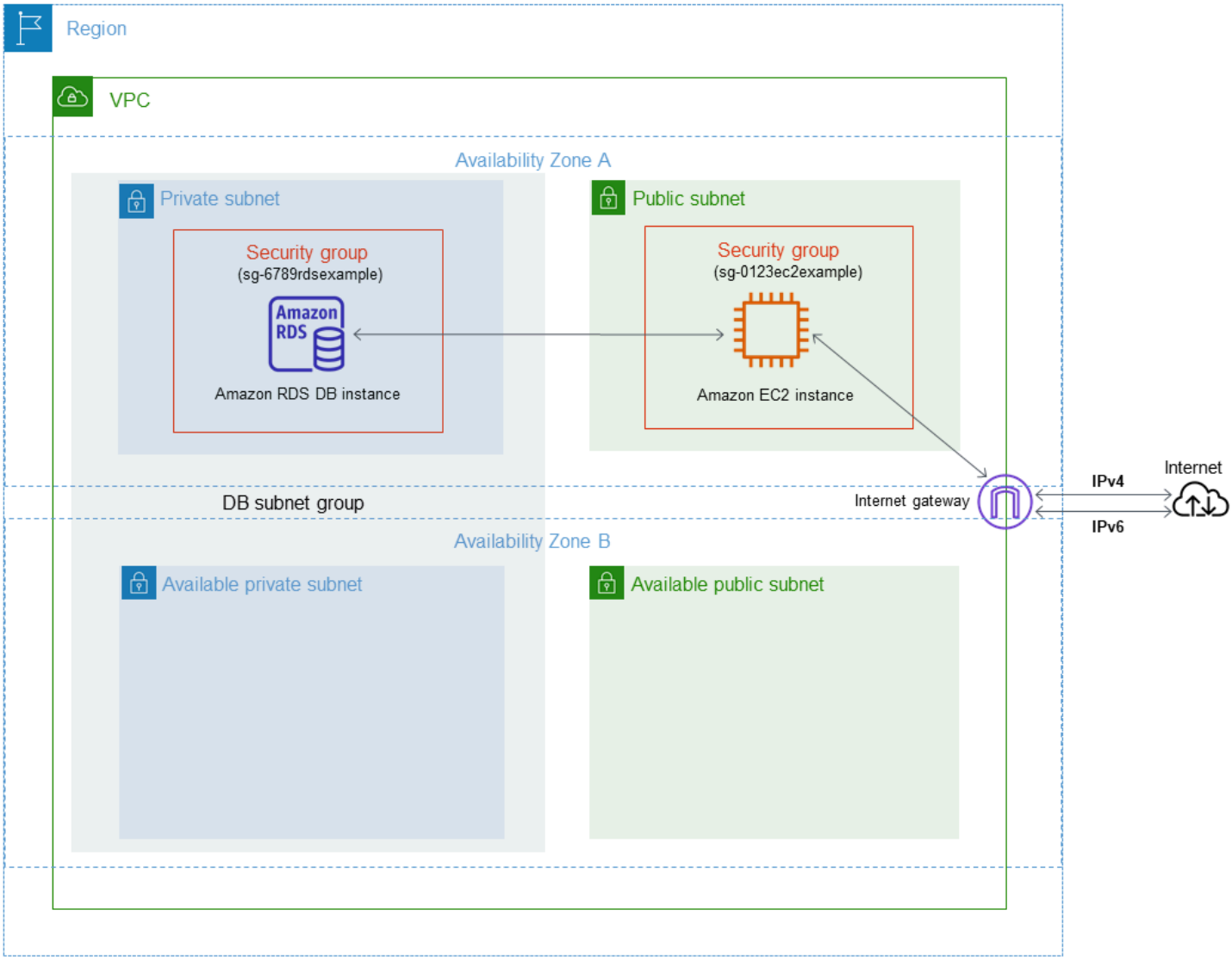
チュートリアル: DB インスタンス用の VPC を作成する (デュアルスタックモード)

一般的なシナリオには、Amazon VPC サービスに基づく仮想プライベートクラウド (VPC) 内の DB インスタンスが含まれます。この VPC は、同じ VPC で実行しているパブリック Amazon EC2 インスタンスとデータを共有します。

このチュートリアルでは、デュアルスタックモードで実行されているデータベースで動作する VPC を、このシナリオで作成します。IPv6 アドレッシングプロトコルを介した接続を可能にするデュアルスタックモード。IP アドレスの割り当てについては、「Amazon RDS IP アドレス指定」を参照してください。

デュアルスタックのネットワークインスタンスは、ほとんどのリージョンでサポートされています。詳細については、「リージョンとバージョンの可用性」を参照してください。デュアルスタックモードの制限については、「デュアルスタックネットワーク DB インスタンスの制限」を参照してください。

以下の図に、このシナリオを示しています。



その他のシナリオについては、VPC の DB インスタンスにアクセスするシナリオ を参照してください。

DB インスタンスは、Amazon EC2 インスタンスでのみ使用でき、パブリックインターネットから使用できないようにする必要があります。したがって、パブリックサブネットとプライベートサブネットを持つ VPC を作成します。Amazon EC2 インスタンスは、パブリックインターネットにアクセスできるようにパブリックサブネットでホストされます。DB インスタンスはプライベートサブネットでホストされます。Amazon EC2 インスタンスは同じ VPC 内でホストされているため、DB インスタンスに接続できます。ただし、DB インスタンスはパブリックインターネットからは使用できないため、セキュリティが向上します。

このチュートリアルでは、別のアベイラビリティーゾーンに追加のパブリックサブネットとプライベートサブネットを設定します。これらのサブネットはチュートリアルでは使用されません。RDS

DB サブネットグループは、少なくとも 2 つのアベイラビリティーゾーン内のサブネットを必要とします。サブネットが追加されたことで、将来的にマルチ AZ DB インスタンス配置に簡単に切り替えることができるようになります。

デュアルスタックモードを使用する DB インスタンスを作成するには、[Network type] (ネットワークタイプ) 設定として [Dual-stack mode] (デュアルスタックモード) を指定します。DB インスタンスを同じ設定で変更することもできます。詳細については、「Amazon RDS DB インスタンスの作成」および「Amazon RDS DB インスタンスを変更する」を参照してください。

このチュートリアルでは、Amazon RDS DB インスタンス 用に VPC を設定する方法について説明します。Amazon VPC の詳細については、「Amazon VPC ユーザーガイド」を参照してください。

プライベートサブネットおよびパブリックサブネットを持つ VPC を作成する

以下の手順で、パブリックサブネットとプライベートサブネットを持つ VPC を作成します。

VPC とサブネットを作成するには

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. AWS マネジメントコンソールで、VPC を作成するリージョンを選択します。この例では、米国東部 (オハイオ) リージョンを使用します。
3. [VPC ダッシュボード] を選択します。VPC の作成を開始するには、[Create VPC] (VPC の作成) を選択します。
4. [VPC Settings] (VPC 設定) の [Resources to create] (作成するリソース) で、[VPC and more] (VPC など) を選択します。
5. 残りの [VPC settings] (VPC 設定) で、これらの値を設定します。
   - [Name tag auto-generation] (ネームタグ自動生成) – **`tutorial-dual-stack`**
   - [IPv4 CIDR block] (IPv4 CIDR ブロック) – **`10.0.0.0/16`**
   - [IPv6 CIDR block] (IPv6 CIDR ブロック) – [Amazon-provided IPv6 CIDR block] (Amazon 提供の IPv6 CIDR ブロック)
   - [Tenancy] (テナンシー) – デフォルト
   - [Number of Availability Zones (AZs)] (アベイラビリティーゾーンの数 (AZ)) – 2
   - [Customize AZs] (AZ をカスタマイズする) – デフォルト値を維持します。
   - [Number of public subnet] (パブリックサブネット数) – 2
   - [Number of private subnets] (プライベートサブネット数) – 2

- [Customize subnets CIDR blocks] (サブネット CIDR ブロックをカスタマイズ) — デフォルト値を維持します。
- [NAT gateways ($)] (NAT ゲートウェイ ($)) – なし
- [Egress only internet gateway] (Egress-only インターネットゲートウェイ): [No] (なし)
- [VPC endpoints] (VPC エンドポイント) – なし
- [DNS options] (DNS オプション) — デフォルト値を維持します。

> **Note**
>
> Amazon RDS では、マルチ AZ DB インスタンス配置をサポートするために、2 つの異なるアベイラビリティーゾーン内のサブネットを少なくとも 2 つ含んでいる必要があります。このチュートリアルではシングル AZ 配置を作成しますが、この要件により将来的に マルチ AZ DB インスタンス配備に簡単に変換できます。

6. [Create VPC (VPC の作成)] を選択します。

パブリック Amazon EC2 インスタンスの VPC セキュリティグループを作成する

次に、パブリックアクセスのためのセキュリティグループを作成します。VPC 内のパブリック EC2 インスタンスに接続するには、インターネットから接続するトラフィックを許可するインバウンドルールを VPC セキュリティグループに追加します。

VPC セキュリティグループを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. [VPC ダッシュボード]、[セキュリティグループ]、[セキュリティグループの作成] の順に選択します。
3. [セキュリティグループの作成] ページで、以下の値を設定します。
   - セキュリティグループ名: **`tutorial-dual-stack-securitygroup`**
   - 説明: **`Tutorial Dual-Stack Security Group`**
   - [VPC ID]: 前に作成した VPC を選択します (例: [vpc-***`identifier`*** (tutorial-dual-stack-vpc)])。
4. インバウンドルールをセキュリティグループに追加します。
   a. Secure Shell (SSH) を使用して VPC の EC2 インスタンスへの接続に使用する IP アドレスを決定します。

インターネットプロトコルバージョン 4 (IPv4) アドレスの例は `203.0.113.25/32` です。インターネットプロトコルバージョン 6 (IPv6) のアドレス範囲の例は `2001:db8:1234:1a00::/64` です。

多くの場合、インターネットサービスプロバイダー (ISP) 経由、またはファイアウォールの内側から静的 IP アドレスなしで接続することがあります。この場合は、クライアントコンピュータが使用する IP アドレスの範囲を検索します。

> **Warning**
>
> IPv4 の `0.0.0.0/0` または IPv6 の `::0` を使用している場合は、すべての IP アドレスが SSH を使ってパブリックインスタンスにアクセスできるようにします。この方法は、テスト環境で短時間なら許容できますが、実稼働環境では安全ではありません。実稼働環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、インスタンスへのアクセスを許可します。

b. [インバウンドルール] セクションで、[ルールの追加] を選択します。

c. 新しいインバウンドルールに次の値を設定して、Amazon EC2 インスタンスへの Secure Shell (SSH) アクセスを許可します。このようにした場合、EC2 インスタンスに接続して SQL クライアントやその他のアプリケーションをインストールできます。EC2 インスタンスへのアクセスできるように IP アドレスを指定します。

- [Type] (タイプ): **`SSH`**
- [Source] (ソース): ステップ a で指定した IP アドレスまたは範囲。IPv4 IP アドレスの例は **`203.0.113.25/32`** です。IPv6 IP アドレスの例は **`2001:DB8::/32`** です。

5. セキュリティグループを作成するには、[Create security group] (セキュリティグループの作成) を選択します。

   セキュリティグループ ID を書き留めます。このチュートリアルで後に必要になります。

プライベート DB インスタンスの VPC セキュリティグループを作成する

DB インスタンスをプライベートのままにするには、プライベートアクセス用の第 2 のセキュリティグループを作成します。VPC 内のプライベート DB インスタンスに接続するには、VPC セキュリティグループにインバウンドルールを追加します。これにより、Amazon EC2 インスタンスからのトラフィックのみを許可します。

VPC セキュリティグループを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. [VPC ダッシュボード]、[セキュリティグループ]、[セキュリティグループの作成] の順に選択します。
3. [セキュリティグループの作成] ページで、以下の値を設定します。
   - セキュリティグループ名: **tutorial-dual-stack-db-securitygroup**
   - 説明: **Tutorial Dual-Stack DB Instance Security Group**
   - [VPC ID]: 前に作成した VPC を選択します (例: [vpc-***identifier*** (tutorial-dual-stack-vpc)])。
4. インバウンドルールをセキュリティグループに追加します。
   a. [インバウンドルール] セクションで、[ルールの追加] を選択します。
   b. 新しいインバウンドルールに次の値を設定して、Amazon EC2 インスタンスからポート 3306 への MySQL トラフィックを許可します。その場合、EC2 インスタンスから DB インスタンスに接続できます。これにより、EC2 インスタンスからデータベースにデータを送信できるようになります。
      - [Type] (タイプ): MySQL/Aurora
      - [Source] (ソース): このチュートリアルで以前に作成した tutorial-dual-stack-securitygroup セキュリティグループの ID (例: sg-9edd5cfb)。
5. セキュリティグループを作成するには、[セキュリティグループの作成] を選択します。

DB サブネットグループを作成する

DB サブネットグループは VPC に作成するサブネットのコレクションで、DB インスタンス用に指定します。DB サブネットグループを使用することにより、DB インスタンスを作成するときに、特定の VPC を指定することができます。DUAL 互換の DB サブネットグループを作成するには、すべてのサブネットが DUAL 互換である必要があります。DUAL 互換であるためには、サブネットに IPv6 CIDR が関連付けられている必要があります。

DB サブネットグループを作成するには

1. VPC 内のデータベースのプライベートサブネットを特定します。
   a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。

b. [VPC Dashboard] (VPC ダッシュボード) を選択してから、[Subnets] (サブネット) を選択します。

c. tutorial-dual-stack-subnet-private1-us-west-2a と tutorial-dual-stack-subnet-private2-us-west-2b という名前のサブネット ID に注意してください。

サブネット ID は、DB サブネットグループを作成するときに必要になります。

2. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

Amazon VPC コンソールではなく、Amazon RDS コンソールに接続してください。

3. [Navigation] ペインで、[Subnet groups] を選択します。
4. [Create DB subnet group] (DB サブネットグループの作成) を選択します。
5. [DB サブネットグループを作成する] ページで、[サブネットグループの詳細] に値を設定します。

- 名前: **`tutorial-dual-stack-db-subnet-group`**
- 説明: **`Tutorial Dual-Stack DB Subnet Group`**
- VPC: tutorial-dual-stack-vpc (vpc-***`identifier`***)

6. [Add subnets] (サブネットの追加) セクションで、[Availability Zones] (アベイラビリティーゾーン) オプションと [Subnets] (サブネット) オプションの値を選択します。

このチュートリアルでは、[Availability Zones] (アベイラビリティーゾーン) として [us-east-2a] と [us-east-2b] を選択します。[Subnets] (サブネット) では、前のステップで特定したプライベートサブネットを選択します。

7. [作成] を選択します。

RDS コンソールの DB サブネットグループリストに新しい DB サブネットグループが表示されます。DB サブネットグループを選択して詳細を表示できます。これには、サポートされているアドレス指定プロトコルと、そのグループに関連付けられたすべてのサブネット、DB サブネットグループによってサポートされるネットワークタイプが含まれます。

デュアルスタックモードの Amazon EC2 インスタンスを作成する

Amazon EC2 インスタンスを作成するには、「Amazon EC2 ユーザーガイド」の「新しいインスタンス起動ウィザードを使用してインスタンスを起動する」の指示に従います。

次に示すように、[Configure Instance Details] (インスタンスの詳細の設定) ページで次の値を設定し、他の値はデフォルトのままにします。

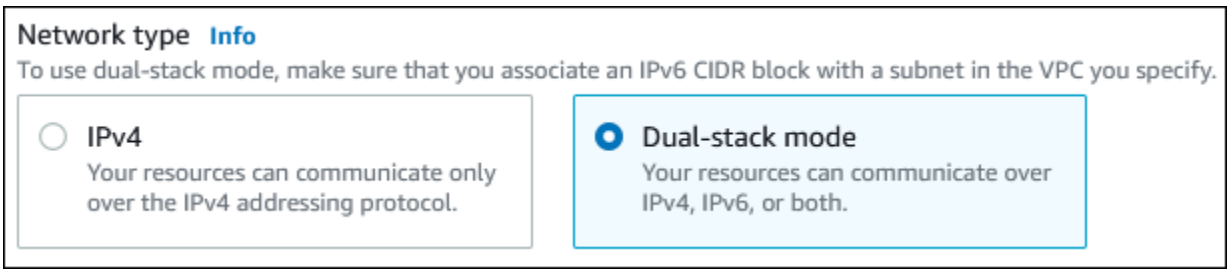
- ネットワーク – パブリックサブネットとプライベートサブネットの両方を持つ既存の VPC を選択します (プライベートサブネットおよびパブリックサブネットを持つ VPC を作成する で作成した tutorial-dual-stack-vpc (vpc-*identifier*) など)。
- [Subnet] (サブネット): 既存のパブリックサブネットを選択します (パブリック Amazon EC2 インスタンスの VPC セキュリティグループを作成する で作成した subnet-***identifier*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a など)。
- [Auto-assign Public IP] (パブリック IP の自動割り当て)]: [Enable] (有効化) を選択します。
- [Auto-assign IPv6 IP]: [Enable] (有効化) を選択します。
- [Firewall (security groups)] (ファイアウォール (セキュリティグループ)) — [Select an existing security group] (既存のセキュリティグループを選択する) を選択します。
- [Common security groups] (共通セキュリティグループ) — `tutorial-securitygroup` で作成された パブリック Amazon EC2 インスタンスの VPC セキュリティグループを作成する などの既存のセキュリティグループを選択します。選択するセキュリティグループに、Secure Shell (SSH) および HTTP アクセスのインバウンドルールが含まれていることを確認します。

デュアルスタックモードの DB インスタンスを作成する

このステップでは、デュアルスタックモードで実行する DB インスタンスを作成します。

DB インスタンスを作成するには

1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
2. コンソールで、DB インスタンスを作成する AWS リージョンを選択します。この例では、米国東部 (オハイオ) リージョンを使用します。
3. ナビゲーションペインで、[データベース] を選択します。
4. [データベースの作成] を選択します。
5. [Create database] (データベースの作成) ページで、[Standard create] (スタンダード作成) オプションがオンになっていることを確認し、 MySQL DB を選択します。
6. [接続] セクションで、次の値を設定します。
    - [Network type] (ネットワークタイプ): [Dual-stack mode] (デュアルスタックモード) を選択します。



- [Virtual private cloud (VPC)] (仮想プライベートクラウド (VPC)): パブリックサブネットとプライベートサブネットの両方を持つ既存の VPC を選択します (プライベートサブネットおよびパブリックサブネットを持つ VPC を作成する で作成した tutorial-dual-stack-vpc (vpc-*identifier*) など)。

  VPC の各サブネットは異なるアベイラビリティーゾーンに存在している必要があります。
- [DB Subnet group] (DB サブネットグループ): VPC の DB サブネットグループ (DB サブネットグループを作成する で作成した tutorial-dual-stack-db-subnet-group など)。
- [Public access] (公開アクセス) — [No] (いいえ) を選択します。
- [VPC security group (firewall)] (VPC セキュリティグループ (ファイアウォール)) — [Choose existing] (既存を選択) を選択します。
- [Existing VPC security groups] (既存の VPC セキュリティグループ) – プライベートアクセス用に設定されている既存の VPC セキュリティグループを選択します (プライベート DB インスタンスの VPC セキュリティグループを作成する で作成した tutorial-dual-stack-db-securitygroup など)。

  他のセキュリティグループ (デフォルトのセキュリティグループなど) は、それぞれの対応する [X] を選択して削除します。
- [Availability zone] (アベイラビリティーゾーン): us-west-2a を選択します。

  AZ 間のトラフィックを回避するには、DB インスタンスと EC2 インスタンスが同じアベイラビリティーゾーンにあることを確認してください。

7. 残りのセクションで、DB インスタンス設定を指定します。各設定の詳細については、「DB インスタンスの設定」を参照してください。

Amazon EC2 インスタンスと DB インスタンスに接続する

Amazon EC2 インスタンスと DB インスタンスをデュアルスタックモードで作成した後、IPv6 プロトコルを使用して各インスタンスに接続できます。IPv6 プロトコルを使用して Amazon EC2 インス

タンスに接続するには、「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」の指示に従ってください。

Amazon EC2 インスタンスから RDS for MySQL DB インスタンスに接続するには、「MySQL DB インスタンスに接続する」の手順に従ってください。

VPC の削除

このチュートリアルの VPC およびその他のリソースを作成後、不要になった場合は、削除できます。

このチュートリアルで作成した VPC にリソースを追加した場合は、VPC を削除する前にこれらを削除しなければならない場合があります。リソースの例としては、Amazon EC2 インスタンスや DB インスタンスなどがあります。詳細については、「Amazon VPC ユーザーガイド」の「VPC の削除」を参照してください。

VPC と関連リソースを削除する方法

1. DB サブネットグループを削除するには、次のようにします。
   a. Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。
   b. [ナビゲーション] ペインで、[サブネットグループ] を選択します。
   c. 削除する DB サブネットグループを選択します ([tutorial-db-subnet-group] など)。
   d. [削除] を選択してから、確認ウィンドウの [削除] を選択します。
2. 次のようにして、VPC ID をメモします。
   a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
   b. [VPC ダッシュボード] を選択してから、[VPC] を選択します。
   c. リストで、作成した VPC を特定します ([tutorial-dual-stack-vpc] など)。
   d. 作成した VPC の [VPC ID] の値をメモします。後続のステップで、この VPC ID が必要になります。
3. セキュリティグループを削除するには、次のようにします。
   a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
   b. [VPC ダッシュボード] を選択してから、[セキュリティグループ] を選択します。
   c. Amazon RDS DB インスタンスのセキュリティグループを選択します ([tutorial-dual-stack-db-securitygroup] など)。

d. [Actions] (アクション) で、[Delete security groups] (セキュリティグループの削除) を選択してから、確認ページで [Delete] (削除) を選択します。

e. [Security Groups] (セキュリティグループ) ページで、Amazon EC2 インスタンスのセキュリティグループを選択します ([tutorial-dual-stack-securitygroup] など)。

f. [Actions] (アクション) で、[Delete security groups] (セキュリティグループの削除) を選択してから、確認ページで [Delete] (削除) を選択します。

4. 次のようにして、NAT ゲートウェイを削除します。

a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。

b. [VPC ダッシュボード] を選択してから、[NAT ゲートウェイ] を選択します。

c. 作成した VPC の NAT ゲートウェイを選択します。VPC ID を使用して、適切な NAT ゲートウェイを識別します。

d. [Actions] (アクション) で、[Delete NAT gateway] (NAT ゲートウェイの削除) を選択します。

e. 確認ページで、「**delete**」を入力してから、[削除] を選択します。

5. VPC の削除

a. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。

b. [VPC ダッシュボード] を選択してから、[VPC] を選択します。

c. 削除する VPC を選択します ([tutorial-dual-stack-vpc] など)。

d. [アクション] で、[VPC の削除] を選択します。

確認ページには、VPC に関連付けられたサブネットを含め、削除される VPC に関連付けられているその他のリソースが表示されます。

e. 確認ページで、「**delete**」を入力してから、[Delete] (削除) を選択します。

6. 次のようにして、Elastic IP アドレスを解放します。

a. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。

b. [EC2 ダッシュボード] を選択してから、[Elastic IP] を選択します。

c. 解放する Elastic IP アドレスを選択します。

d. [Actions] (アクション) で、[Release Elastic IP addresses] (Elastic IP アドレスの解放) を選択します。

e. 確認ページで、[リリース] を選択します。

VPC 外の DB インスタンスを VPC 内に移行する

EC2-Classic プラットフォームの DB インスタンスのいくつかは VPC にありません。DB インスタンスが VPC 内に存在しない場合は、AWS マネジメントコンソール を使用して、VPC 内に DB インスタンスを簡単に移行できます。VPC 外の DB インスタンスを VPC に移行する前に、VPC を作成する必要があります。

EC2-Classic は 2022 年 8 月 15 日に廃止されました。EC2-Classic から VPC に移行していない場合、できるだけ早く移行することをお勧めします。詳細については、「Amazon EC2 ユーザーガイド」の「EC2-Classic から VPC へ移行」およびブログ記事「EC2-Classic ネットワーキングがリタイア — 準備方法」を参照してください。

Important

Amazon RDS を初めてご利用になる場合、以前に DB インスタンスを作成したことがない場合、または以前には使用したことがない AWS リージョンに DB インスタンスを作成する場合、使用するプラットフォームは EC2-VPC で、デフォルトの VPC があることがほとんどです。VPC での DB インスタンスの使用については、「VPC 内の DB インスタンスの使用」を参照してください。

DB インスタンスの VPC を作成するには、以下のステップを実行します。

- ステップ 1: VPC を作成する
- ステップ 2: DB サブネットグループを作成する
- ステップ 3: VPC セキュリティグループを作成する

VPC を作成した後、以下のステップに従って VPC 内に DB インスタンスを移行します。

- DB インスタンスの VPC の更新

移行の直前に DB インスタンスのバックアップを作成することを強くお勧めします。そうすることで、移行が失敗した場合でも、データを復元できます。詳細については、「データのバックアップ、復元、エクスポート」を参照してください。

VPC 内に DB インスタンスを移行する際の制限事項を以下に示します。

- 前の世代の DB インスタンスクラス - 前の世代の DB インスタンスクラスは、VPC プラットフォームではサポートされない場合があります。DB インスタンスを VPC に移動するときは、db.m3 または db.r3 DB インスタンスクラスを選択します。DB インスタンスを VPC に移動したら、後の DB インスタンスクラスを使用するように DB インスタンスをスケーリングできます。VPC でサポートされるインスタンスクラスの詳細なリストについては、Amazon RDS インスタンスタイプを参照してください。
- マルチ AZ - VPC 外のマルチ AZ DB インスタンスの VPC への移行は現在サポートされていません。DB インスタンスを VPC に移動するには、まず DB インスタンスを変更して、シングル AZ 配置にします。マルチ AZ 配置 設定を「いいえ」に変更します。DB インスタンスを VPC に移動したら、再度変更してマルチ AZ 配置にします。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。
- リードレプリカ - VPC 外の DB インスタンスとリードレプリカの VPC への移行は現在サポートされていません。DB インスタンスを VPC に移動するには、まずすべてのリードレプリカを削除します。DB インスタンスを VPC に移動したら、リードレプリカを再作成します。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。
- オプショングループ - DB インスタンスを VPC に移動し、DB インスタンスがカスタムオプショングループを使用している場合は、DB インスタンスに関連付けられているオプショングループを変更します。オプショングループはプラットフォーム固有であるため、VPC に移行するとプラットフォームも変更されます。この場合にカスタムオプショングループを使用するには、DB インスタンスにデフォルトの VPC オプショングループを割り当てる、移行する VPC 内の別の DB インスタンスで使用されているオプショングループを割り当てる、または新しいオプショングループを作成して DB インスタンスに割り当てます。詳細については、「オプショングループを使用する」を参照してください。

VPC 外の DB インスタンスを最小限のダウンタイムで VPC 内に移動する代替方法

以下の代替方法を使用して、VPC 内にない DB インスタンスを最小限のダウンタイムで VPC に移動できます。これらの代替方法により、ソース DB インスタンスの中断を最小限に抑え、移行中にユーザートラフィックを処理できるようにします。ただし、VPC への移行に必要な時間は、データベースのサイズとライブワークロードの特性によって異なります。

- AWS Database Migration Service (AWS DMS) - AWS DMS は、ソース DB インスタンスを完全に稼働させながらデータのライブ移行を可能にしますが、限定された DDL ステートメントのセットのみをレプリケートします。AWS DMS は、インデックス、ユーザー、権限、ストアドプロシージャ、テーブルデータに直接関係しないその他のデータベースの変更などの項目を伝播しません。さらに、AWS DMS は、初期 DB インスタンスの作成に RDS スナップショットを自動的に使用し

ないため、移行時間が長くなる可能性があります。詳細については、「AWS Database Migration Service」を参照してください。

- DB スナップショットの復元またはポイントインタイムリカバリ - DB インスタンスのスナップショットを復元するか、DB インスタンスを特定の時点に復元することによって、DB インスタンスを VPC に移動できます。詳細については、DB インスタンスへの復元およびAmazon RDS の DB インスタンスを特定の時点に復元するを参照してください。

Amazon RDS のクォータと制約

Amazon RDS のリソースのクォータと名前付け制約の説明は次のとおりです。

トピック

- Amazon RDS のクォータ
- Amazon RDS の命名に関する制約
- データベース接続の最大数
- Amazon RDS のファイルサイズ制限

Amazon RDS のクォータ

各 AWS アカウントには、AWS リージョン別に、作成できる Amazon RDS リソースの数に適用されるクォータがあります。リソースのクォータに達すると、そのリソースを作成するための追加の呼び出しは、失敗して例外が発生します。

次の表に、AWS リージョンごとのリソースとそのクォータを示します。

名前	デフォルト	引き上げ可能	説明
DB セキュリティグループごとの承認	サポートされている各リージョン: 20	不可	DB セキュリティグループあたりのセキュリティグループ認可数
DB クラスターごとのカスタムエンドポイント	サポートされている各リージョン：5	あり	現在のリージョンにおいて、このアカウントで Aurora DB クラスターごとに作成できるカスタムエンドポイントの最大数。この値は、アカウントにおける DB クラス

名前	デフォルト	引き上げ可能	説明
			ター内のカスタムエンドポイントの最大数を反映します。アカウントの他の DB クラスターでは、カスタムエンドポイントの数が少ない場合があります。
カスタムエンジンバージョン	サポートされている各リージョン: 40	可能	現在のリージョンでこのアカウントに許可されるカスタムエンジンバージョンの最大数
DB クラスターのパラメータグループ	サポートされている各リージョン: 50	可能	DB クラスターパラメータグループの最大数
DB クラスター	サポートされている各リージョン: 40	可能	現在のリージョンでこのアカウントに許可される Aurora クラスターの最大数
DB インスタンス	ap-south-1: 20 他のサポートされている各リージョン: 40	可能	現在のリージョンでこのアカウントに許可される DB インスタンスの最大数

名前	デフォルト	引き上げ可能	説明
DB シャードグループ	サポートされている各リージョン：5	あり	現在のリージョンにおけるこのアカウントの Aurora Limitless Database の DB シャードグループの最大数
DB サブネットグループ	ap-south-1: 20 他のサポートされている各リージョン: 50	可能	DB サブネットグループの最大数
Data API HTTP リクエスト本文のサイズ	サポートされている各リージョン: 4 MB	不可	HTTP リクエスト本文に許可される最大サイズ。
Data API 最大同時実行クラスターシークレットペア	サポートされている各リージョン: 30	不可	AWS リージョンの現在のアカウントにおける同時実行 Data API リクエストでの Aurora Serverless v1 DB クラスターとシークレットの一意のペアの最大数。

名前	デフォルト	引き上げ可能	説明
Data API 最大同時実行リクエスト	サポートされている各リージョン: 500	不可	Aurora Serverless v1 DB クラスターに対する Data API リクエストのうち、同じシークレットを使用し、同時に処理可能であるものの最大数。追加のリクエストはキューに入れられ、処理中のリクエストが完了すると処理されます。
データ API の結果セットの最大サイズ	サポートされている各リージョン: 1 MB	不可	Data API によって返されるデータベース結果セットの最大サイズ。
データ API の JSON レスポンス文字列の最大サイズ	サポートされている各リージョン: 10 MB	不可	RDS データ API によって返される簡略化された JSON レスポンス文字列の最大サイズ。
1 秒あたりのデータ API リクエスト数	サポートされている各リージョン: 1,000/秒	不可	現在の AWS リージョンにおける現在のアカウントで許可されている、Data API に対するリクエスト/秒の最大数 このクォータは、Amazon Aurora Serverless v1 クラスターにのみ適用されます。

名前	デフォルト	引き上げ可能	説明
イベントサブスクリプション	サポートされている各リージョン: 20	可能	イベントサブスクリプションの最大数
DB クラスターごとの IAM ロール	サポートされている各リージョン : 5	可能	DB クラスターに関連付けられる IAM ロールの最大数
DB インスタンスごとの IAM ロール	サポートされている各リージョン : 5	可能	DB インスタンスに関連付けられる IAM ロールの最大数
統合	サポートされている各リージョン: 100	不可	現在の AWS リージョンでこのアカウントに許可される統合の最大数。
手動 DB クラススナップショット	サポートされている各リージョン: 100	可能	手動 DB クラススナップショットの最大数
手動の DB インスタンスのスナップショット	サポートされている各リージョン: 100	可能	手動 DB インスタンススナップショットの最大数
オプショングループ	サポートされている各リージョン: 20	可能	オプショングループの最大数

名前	デフォルト	引き上げ可能	説明
パラメータグループ	ap-south-1: 20 他のサポートされている各リージョン: 50	可能	パラメータグループの最大数
プロキシ	サポートされている各リージョン: 20	可能	現在の AWS リージョンでこのアカウントに許可されるプロキシの最大数
プライマリあたりのリードレプリカ数	サポートされている各リージョン: 15	可能	プライマリ DB インスタンスあたりのリードレプリカの最大数。このクォータは、Amazon Aurora 用に調整できません。
リザーブド DB インスタンス	ap-south-1: 20 他のサポートされている各リージョン: 40	可能	現在の AWS リージョンでこのアカウントに許可される予約 DB インスタンスの最大数
セキュリティグループ	ap-south-1: 20 他のサポートされている各リージョン: 25	可能	DB セキュリティグループの最大数
DB サブネットグループあたりのサブネット	サポートされている各リージョン: 20	不可	DB サブネットグループあたりのサブネットの最大数

名前	デフォルト	引き上げ可能	説明
すべての DB インスタンスの合計ストレージ	サポートされている各リージョン: 100,000	可能	一緒に追加されたすべての Amazon RDS DB インスタンスの EBS ボリュームの最大ストレージ合計 (GB 単位)。このクォータは、各 DB クラスターの最大クラスターボリュームが 128 TiB である Amazon Aurora には適用されません。

Note

デフォルトでは、最大で合計 40 の DB インスタンスを持つことができます。RDS DB インスタンス、Aurora DB インスタンス、Amazon Neptune インスタンス、および Amazon DocumentDB インスタンスは、このクォータに該当します。
Amazon RDS DB インスタンスには、次の制限が適用されます。

- 「ライセンス込み」のモデルでは、各 SQL Server のエディション (Enterprise、Standard、Web、および Express) ごとにインスタンスをそれぞれ最大 10 使用することができます。
- 「ライセンス込み」モデルに基づく Oracle 向けの 10
- 「Bring-Your-Own-License (BYOL)」ライセンスモデルの Db2 の場合は 40
- MySQL、MariaDB、または PostgreSQL では、40 使用できます。
- 「Bring-Your-Own-License (BYOL)」 モデルの Oracle では、40 使用できます。

アプリケーションでさらに多くの DB インスタンスが必要な場合は、Service Quotas コンソールを開いて、追加の DB インスタンスをリクエストできます。ナビゲーションペイン

で、[AWS のサービス] を選択します。[Amazon Relational Database Service (Amazon RDS)] を選択してクォータを選択し、指示に従ってクォータの引き上げをリクエストします。詳細については、「Service Quotas ユーザーガイド」の「クォータの引き上げのリクエスト」を参照してください。
RDS for Oracle では、各リージョンでソース DB インスタンスごとに最大 15 個のリードレプリカを作成できますが、レプリカを 5 個に制限してレプリケーションの遅延を最小限に抑えることをお勧めします。
AWS Backup によって管理されるバックアップは手動 DB スナップショットと見なされますが、手動スナップショットクォータにはカウントされません。AWS Backup の詳細については、「AWS Backup デベロッパーガイド」を参照してください。

クロスリージョン自動バックアップのデフォルトクォータは AWS アカウントごとに 20、スナップショットの同時コピーリクエスト数のデフォルトクォータは AWS アカウントのリージョンごとに 20、同時クロスリージョンリードレプリカは AWS アカウントのリージョンごとに 20 であることに注意してください。

いずれかの RDS API オペレーションを使用して、1 秒あたりの呼び出し数のデフォルトのクォータを超えると、Amazon RDS API では次のようなエラーを発行します。

ClientError: *API_Name* オペレーションの呼び出し時にエラー (ThrottlingException) が発生しました (レート超過)。

この場合、1 秒あたりのコール回数を減らします。クォータは、ほとんどのユースケースをカバーするようにしてあります。より大きなクォータが必要な場合は、次のいずれかのオプションを使用してクォータの引き上げをリクエストできます。

- コンソールで、[Service Quotas コンソール] を開きます。
- AWS CLI で、AWS CLI コマンド request-service-quota-increase を使用します。

詳細については、Service Quotas ユーザーガイドを参照してください。

Amazon RDS の命名に関する制約

Amazon RDS の命名に関する制約を次に示します。

- DB インスタンス識別子:

  - 1 ~ 63 個の英数字またはハイフンを使用する必要があります。
  - 1 字目は文字である必要があります。
  - 文字列の最後にハイフンを使用したり、ハイフンを 2 つ続けて使用したりすることはできません。
  - 1 つの AWS アカウント、1 つの AWS リージョンにつき、すべての DB インスタンスにおいて一意である必要があります。
- 初期データベース名:
  - データベース名の制約は、データベースエンジンごとに異なります。詳細については、各 DB インスタンスの作成時に使用できる設定を参照してください。
  - SQL Server – DB インスタンスを作成した後、データベースを作成します。
- マスターユーザー名 – マスターユーザー名の制約は、データベースエンジンごとに異なります。詳細については、DB インスタンスの作成時に使用できる設定を参照してください。
- マスターパスワード:
  - データベースのマスターユーザーのパスワードには、すべての印刷可能な ASCII 文字 (/、'、"、@、またはスペースを除く) を使用できます。

    Oracle の場合、& は追加の文字制限です。
  - パスワードには、DB エンジンに応じて、次の数の印字可能な ASCII 文字を含めることができます:
    - Db2: 8 ~ 255
    - MariaDB および MySQL: 8 ~ 41 文字
    - Oracle: 8 ~ 30 文字
    - SQL Server および PostgreSQL: 8 ~ 128 文字
- DB パラメータグループ:
  - 1 ~ 255 個の英数字を使用する必要があります。
  - 1 字目は文字である必要があります。
  - この名前では、ハイフンを使用できますが、末尾に使用したり、2 つ続けて使用したりすることはできません。
- DB サブネットグループ:
  - 1 ~ 255 文字を使用する必要があります。
  - 英数字、スペース、ハイフン、アンダースコア、ピリオドを使用できます。

データベース接続の最大数

同時データベース接続の最大数は、DB エンジンのタイプと DB インスタンスクラスのメモリ割り当てによって異なります。最大接続数は、通常は DB インスタンスに関連付けられたパラメータグループで設定されます。例外は、Microsoft SQL Server Management Studio (SSMS) の DB インスタンスのサーバープロパティで設定される Microsoft SQL Server です。

データベース接続は、メモリを消費します。これらのパラメータのいずれかを高く設定しすぎると、メモリ不足が発生し、DB インスタンスが互換性のないパラメータステータスになる可能性があります。詳細については、「メモリ制限と互換性のないパラメータの状態の診断と解決」を参照してください。

アプリケーションが頻繁に接続を開いたり閉じたりする場合や、長時間の接続を多数開いたままにする場合は、Amazon RDS Proxy の使用を推奨します。RDS Proxy は、接続プーリングを使用してデータベース接続を安全かつ効率的に共有する、フルマネージドの高可用性データベースプロキシです。RDS Proxy の詳細については、Amazon RDS Proxy を参照してください。

> **Note**
>
> Oracle の場合は、ユーザープロセス、ユーザーセッションとシステムセッションの最大数を設定します。
> Db2 の場合、最大接続数を設定することはできません。上限は 64000 です。

次の表は、さまざまな DB エンジンの最大データベース接続数に関する情報を示しています。

DB エンジン	Parameter	許可される値	デフォルト値	説明
MariaDB	`max_connections`	1-100000	• MariaDB 10.5 以降のバージョンでは、デフォルトは次のようになります。 LEAST({DBInstanceClassMemory/25165760},12000)	許可されるクライアントの同時接続数

DB エンジン	Parameter	許可される値	デフォルト値	説明
			この式は実質的に MB/25 と同等です。 デフォルト値の計算結果の値が 12,000 を超える場合、Amazon RDS は制限を 12,000 に設定します。 • MariaDB バージョン 10.4 の場合: {DBInstanceClassMemory/12582880} この式は実質的に MB/12 と同等です。	
MySQL	`max_connections`	1-100000	{DBInstanceClassMemory/12582880} この式は実質的に MB/12 と同等です。	許可されるクライアントの同時接続数
Oracle	`processes`	80-20000	LEAST({DBInstanceClassMemory/9868951}, 20000)	ユーザープロセス
Oracle	`sessions`	100-65535	該当しない	ユーザーセッションとシステムセッション
PostgreSQL	`max_connections`	6–262143	LEAST({DBInstanceClassMemory/9531392}, 5000)	同時接続の最大数

DB エンジン	Parameter	許可される値	デフォルト値	説明
SQL Server	`user connections`	0-32767	0 (無制限)	同時接続の最大数 詳細については、「ユーザー接続の設定 (サーバー設定オプション)」を参照してください。

`DBInstanceClassMemory` の単位はバイトです。この数値の計算方法の詳細については、「DB パラメータの指定」を参照してください。オペレーティングシステムおよび RDS 管理プロセス用に予約されているメモリにより、このメモリサイズは、DB インスタンスクラスのハードウェア仕様 に示すギビバイト (GiB) 単位の値よりも小さくなります。

例えば、一部の DB インスタンスクラスには 8 GiB のメモリがあり、これは 8,589,934,592 バイトです。メモリが 8 GiB の DB インスタンスクラスで実行されている MySQL DB インスタンス (db.m7g.large など) の場合、合計メモリを使用する式は `8589934592/12582880=683` になります。ただし、変数 `DBInstanceClassMemory` によって、この DB インスタンスを管理するオペレーティングシステムと RDS プロセスに予約されている量が自動的に減算されます。次に、減算した残りが 12,582,880 で除算されます。この計算の結果、`max_connections` の値は 683 ではなく約 630 になります。この値は、DB インスタンスクラスと DB エンジンによって異なります。

MariaDB または MySQL DB インスタンスが db.t3.micro や db.t3.small などの小さな DB インスタンスクラスで実行されている場合、使用可能な合計メモリは少なくなります。これらの DB インスタンスクラスでは、使用可能なメモリのかなりの部分が RDS によって予約されるため、値 `max_connections` に影響します。例えば、db.t3.micro DB インスタンスクラスで実行されている MySQL DB インスタンスのデフォルトの最大接続数は約 60 です。DB MariaDB インスタンスまたは MySQL DB インスタンスの `max_connections` 値を確認するには、そのインスタンスに接続し、次の SQL コマンドを実行します。

```
SHOW GLOBAL VARIABLES LIKE 'max_connections';
```

Amazon RDS のファイルサイズ制限

ファイルサイズの制限は、特定の Amazon RDS DB インスタンスに適用されます。詳細については、次のエンジン固有の制限を参照してください。

- Amazon RDS での MariaDB のファイルサイズ制限
- Amazon RDS での MySQL のファイルサイズ制限
- Amazon RDS での Oracle のファイルサイズ制限

Amazon RDS のトラブルシューティング

以下のシナリオを使用して、Amazon RDS や Amazon Aurora の DB インスタンスで発生する問題をトラブルシューティングします。

トピック

- Amazon RDS DB インスタンスに接続できない
- Amazon RDS のセキュリティの問題
- 互換性のないネットワーク状態のトラブルシューティング
- DB インスタンス所有者のパスワードのリセット
- Amazon RDS DB インスタンスの停止または再起動
- Amazon RDS DB パラメータの変更が有効にならない
- Amazon RDS DB インスタンスのストレージ不足
- Amazon RDS で利用可能な DB インスタンスが不足している
- Amazon RDS の解放可能なメモリの問題
- MySQL および MariaDB の問題
- バックアップ保持期間を 0 に設定できない

Amazon RDS API を使用した問題のデバッグについては、「Amazon RDS のアプリケーションのトラブルシューティング」を参照してください。

Amazon RDS DB インスタンスに接続できない

DB インスタンスに接続できない場合、一般的な原因として次のようなものがあります。

- インバウンドルール - ローカルのファイアウォールによって適用されているアクセスルールと DB インスタンスへのアクセスが許可された IP アドレスが一致していない可能性があります。問題の原因として最も多いのは、セキュリティグループのインバウンドルールです。

  デフォルトでは、DB インスタンスへのアクセスは許可されていません。アクセスは、DB インスタンスとの間のトラフィックを許可する VPC に関連付けられたセキュリティグループによって許可されます。必要に応じて、特定の状況のインバウンドルールとアウトバウンドルールをセキュリティグループに追加します。IP アドレス、IP アドレスの範囲、または別の VPC セキュリティグループを指定できます。

> **Note**
>
> 新しいインバウンドルールを追加するときに [出典] に [マイ IP] を選択すると、ブラウザで検出された IP アドレスから DB インスタンスへのアクセスを許可できます。

セキュリティグループの設定の詳細については、「セキュリティグループを作成して VPC 内の DB インスタンスへのアクセスを提供する」を参照してください。

> **Note**
>
> 169.254.0.0/16 の範囲内の IP アドレスからのクライアント接続は許可されていません。これは、ローカルリンクのアドレス指定に使用される Automatic Private IP Addressing Range (APIPA) です。

- パブリックアクセス - クライアントアプリケーションを使用するなど、VPC の外部から DB インスタンスに接続するには、インスタンスにパブリック IP アドレスが割り当てられている必要があります。

  インスタンへのパブリックアクセスを許可するには、インスタンスを変更し、[Public accessibility (パブリックアクセス)] で [Yes (はい)] を選択します。詳細については、「VPC 内の DB インスタンスをインターネットから隠す」を参照してください。
- ポート - DB インスタンスの作成時に指定したポートが、ローカルのファイアウォールの制限によって通信の送受信に使用できない場合があります。指定したポートをインバウンドおよびアウトバウンドの通信に使用することがネットワークで許可されているかどうかを判断するには、ネットワーク管理者に確認します。
- 可用性 - 新しく作成された DB インスタンスでは、DB インスタンスが使用できるようになるまで、DB インスタンスのステータスは `creating` になります。ステータスが `available` になると、DB インスタンスに接続できます。DB インスタンスのサイズによっては、インスタンスが利用可能になるまでに最大 20 分かかることがあります。
- 内部ゲートウェイ - DB インスタンスをパブリックにアクセス可能にするには、その DB サブネットグループのサブネットにインターネットゲートウェイが必要です。

  サブネットのインターネットゲートウェイを設定するには

  1. AWS マネジメントコンソール にサインインし、Amazon RDS コンソール (https://console.aws.amazon.com/rds/) を開きます。

2. ナビゲーションペインで、[データベース] を選択し、DB インスタンスの名前を選択します。
3. [接続とセキュリティ] タブで、[VPC] の下の VPC ID と [サブネット] の下のサブネット ID の値を書き留めます。
4. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
5. ナビゲーションペインで、[Internet Gateways] を選択してください。ご使用の VPC にアタッチされているインターネットゲートウェイがあることを確認します。それ外の場合は、[Create Internet Gateway] を選択してインターネットゲートウェイを作成します。インターネットゲートウェイを選択し、[VPC にアタッチ] を選択して指示どおりにインターネットゲートウェイを VPC にアタッチします。
6. ナビゲーションペインで [Subnets] を選択し、サブネットを選択します。
7. [Route Table] タブで、送信先として `0.0.0.0/0`、ターゲットとして VPC のインターネットゲートウェイが指定されたルートがあることを確認します。

   IPv6 アドレスを使用してインスタンスに接続する場合はインターネットゲートウェイを指しているすべての IPv6 トラフィック (`::/0`) 用のルートがあることを確認します。それ以外の場合は以下の作業を行います。

   a. ルートテーブルの ID (rtb-xxxxxxxx) を選択して、ルートテーブルに移動します。
   b. [ルーター] タブで、[ルーター編集] を選択してください。[Add route] を選択して、`0.0.0.0/0` を送信先として追加し、インターネットゲートウェイをターゲットとして使用します。

      IPv6 の場合は、[Add route] を選択して、`::/0` を送信先として追加し、インターネットゲートウェイをターゲットとして使用します。
   c. [ルートの保存] を選択します。

   また、IPv6 エンドポイントに接続する場合は、クライアントの IPv6 アドレス範囲が DB インスタンスへの接続を認可されていることを確認してください。

詳細については、「VPC 内の DB インスタンスの使用」を参照してください。

エンジン固有の接続の問題については、以下のトピックを参照してください。

- SQL Server DB インスタンスへの接続のトラブルシューティング
- Oracle DB インスタンスへの接続のトラブルシューティング

- RDS for PostgreSQL インスタンスへの接続に関するトラブルシューティング
- MySQL および MariaDB の最大接続数

DB インスタンスへの接続のテスト

一般的な Linux または Microsoft Windows のツールを使用して DB インスタンスへの接続をテストできます。

Linux または UNIX のターミナルからは、次のように入力することで接続をテストできます。*DB-instance-endpoint* をエンドポイントに、*port* を DB インスタンスのポートに置き換えます。

```
nc -zv DB-instance-endpoint port
```

コマンドと戻り値の例を以下に示します。

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Windows ユーザーは、Telnet を使用して DB インスタンスへの接続をテストできます。Telnet での操作は、接続のテスト以外はサポートされていないことに注意してください。接続に成功した場合、このアクションはメッセージを返しません。接続に失敗した場合は、次のようなエラーメッセージが返されます。

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Telnet アクションが成功を示している場合、セキュリティグループは適切に設定されています。

> **Note**
>
> Amazon RDS は ping などの ICMP (Internet Control Message Protocol) トラフィックを受け付けません。

接続認証のトラブルシューティング

場合によっては、DB インスタンスに接続できても認証エラーが発生することがあります。このような場合、DB インスタンスのマスターユーザーパスワードのリセットが必要になることがあります。これを行うには、RDS インスタンスを変更します。

DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS のセキュリティの問題

セキュリティ問題を回避するために、AWS アカウントルートユーザー E メールアドレスとパスワードをユーザーアカウントに使用しないでください。ベストプラクティスは、ルートユーザーを使用してユーザーを作成し、DB ユーザーアカウントに割り当てることです。また、必要に応じて、ルートユーザーを使用して他のユーザーアカウントを作成することもできます。

ユーザーの作成の詳細については、「AWS アカウント での IAM ユーザーの作成」を参照してください。AWS IAM アイデンティティセンター でのユーザー作成の詳細については、「Manage identities in IAM Identity Center」(IAM Identity Center での ID の管理) を参照してください。

エラーメッセージ「アカウント属性の取得に失敗しました。コンソールの特定の機能が損なわれる可能性があります。」

このエラーが発生する原因はいくつかあります。アカウントにアクセス許可がないか、アカウントが正しく設定されていない可能性があります。新規のアカウントの場合は、アカウントの準備がまだ整っていない可能性があります。既存のアカウントの場合は、DB インスタンスの作成などの特定の操作を実行するためのアクセスポリシーでアクセス許可が設定されてない可能性があります。問題を修正するには、管理者が必要なロールをアカウントに与える必要があります。詳細については、「IAM ドキュメント」を参照してください。

互換性のないネットワーク状態のトラブルシューティング

非互換ネットワーク状態の場合、データベースレベルでは引き続きデータベースにアクセスできる可能性がありますが、変更や再起動はできません。

原因

DB インスタンスのネットワーク非互換状態は、次のいずれかのアクションの結果である可能性があります。

- DB インスタンスクラスの変更。
- マルチ AZ DB クラスターデプロイメントを使用するための DB インスタンスの変更。
- メンテナンスイベントによるホストの置き換え。
- 代替 DB インスタンスの起動。
- スナップショットバックアップからの復元。
- 以前に停止した DB インスタンスの開始。

解像度

start-db-instance コマンドを使用する

互換性のないネットワーク状態にあるデータベースを修正するには、以下の手順に従ってください。

1. https://console.aws.amazon.com/rds/ を開いて、[データベース]ナビゲーションペインを選択します。
2. 非互換ネットワーク状態の DB インスタンスを選択し、[接続とセキュリティ] タブから DB インスタンス識別子、VPC ID、およびサブネット ID を書き留めます。
3. AWS CLI を使用して `start-db-instance` コマンドを実行します。`--db-instance-identifier` 値を指定します。

> **Note**
>
> データベースが非互換モードのときにこのコマンドを実行すると、ある程度のダウンタイムが発生する可能性があります。
> `start-db-instance` コマンドを実行しても、SQL Server DB インスタンス用 RDS の問題は解決されません。

コマンドが正常に実行された場合、データベースのステータスは [使用可能] に変わります。

データベースが再起動すると、DB インスタンスは、非互換ネットワーク状態に移行する前にインスタンスで実行された最後の操作を実行する可能性があります。これにより、インスタンスは非互換ネットワーク状態に戻る可能性があります。

`start-db-instance` コマンドが失敗するか、インスタンスが互換性のないネットワーク状態に戻る場合は、RDS コンソールの [データベース] ページで、データベースを選択します。[ログとイベン

ト] セクションに移動します。[最近のイベント] セクションには、今後の解決手順が表示されます。メッセージは次のように分類されます。

- INTERNAL RESOURCE CHECK: 内部リソースに問題がある可能性があります。
- DNS CHECK: VPC コンソールで VPC の DNS 解決とホスト名を確認します。
- ENI CHECK: データベースの Elastic Network Interface (ENI) が存在しない可能性があります。
- GATEWAY CHECK: 公開されているデータベースのインターネットゲートウェイは VPC にアタッチされていません。
- IP CHECK: サブネットには空き IP アドレスがありません。
- SECURITY GROUP CHECK: データベースに関連付けられているセキュリティグループがないか、セキュリティグループが無効です。
- SUBNET CHECK: DB サブネットグループに有効なサブネットがないか、サブネットに問題があります。
- VPC CHECK: データベースに関連付けられている VPC は無効です。

ポイントインタイムリカバリを実行する

データベースが互換性のないネットワーク状態になった場合に備えて、バックアップ (スナップショットまたは論理) を作成するのがベストプラクティスです。「バックアップの概要」を参照してください。自動化バックアップを有効にした場合は、データベースへの書き込みを一時的に停止し、ポイントインタイムリカバリを実行してください。

> **Note**
>
> インスタンスが互換性のないネットワーク状態になると、DB インスタンスにアクセスして論理バックアップを実行できなくなる可能性があります。

自動バックアップを有効にしていない場合は、新しい DB インスタンスを作成します。次に、AWS Database Migration Service(AWS DMS) を使用してデータを移行するか、またはバックアップと復元ツールを使用します。

これで問題が解決しない場合は、サポート に問い合わせてサポートを受けてください。

DB インスタンス所有者のパスワードのリセット

DB インスタンスからロックアウトされた場合は、マスターユーザーとしてログインできます。その後、他の管理ユーザーまたはロールの認証情報をリセットできます。マスターユーザーとしてログインできない場合は、AWS アカウントの所有者がマスターユーザーのパスワードをリセットできます。リセットする必要がある管理アカウントまたはロールの詳細については、「マスターユーザーアカウント権限」を参照してください。

DB インスタンスのパスワードは、Amazon RDS コンソール、AWS CLI コマンドの modify-db-instance、または ModifyDBInstance API オペレーションを使用して変更できます。DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS DB インスタンスの停止または再起動

DB インスタンスの停止は、DB インスタンスが再起動されたときに発生する可能性があります。また、DB インスタンスがアクセスできない状態になったときやデータベースが再開されたときに発生する可能性があります。再起動は、DB インスタンスを手動で再起動した場合に発生することがあります。また、DB インスタンスの設定を設定を変更した場合、その設定を有効にするために再起動が必要になることがあります。

DB インスタンスの再起動は、再起動が必要な設定を変更したとき、または手動で再起動を実行したときにのみ行われます。設定を変更し、その変更を直ちに有効にすることをリクエストした場合、直ちに再起動を実行できます。または、DB インスタンスのメンテナンス期間中に発生することもあります。

以下のいずれかが発生したとき、DB インスタンスの再起動は直ちに実行されます。

- DB インスタンスのバックアップ保持期間を 0 から 0 以外の値または 0 以外の値から 0 に変更します。次に、[Apply Immediately] (すぐに適用) を `true` に設定します。
- DB インスタンスクラスを変更し、[すぐに適用] を [`true`] に設定する。
- ストレージのタイプを [Magnetic (Standard) (マグネティック (スタンダード))] から [General Purpose (SSD) (汎用 (SSD))] または [Provisioned IOPS (SSD) (プロビジョンド IOPS (SSD))] に変更する。あるいは、[Provisioned IOPS (SSD) (プロビジョンド IOPS (SSD))] または [General Purpose (SSD) (汎用 (SSD))] から [Magnetic (Standard) (マグネティック (スタンダード))] に変更する。

以下のいずれかが発生したとき、DB インスタンスの再起動はメンテナンス時間中に実行されます。

- DB インスタンスのバックアップ保持期間を 0 から 0 以外の値または 0 以外の値から 0 に変更し、[すぐに適用] を [false] に設定する。
- DB インスタンスクラスを変更し、[すぐに適用] を [false] に設定する。

DB パラメータグループの静的パラメータを変更する場合、その変更はパラメータグループに関連付けられている DB インスタンスを再起動するまで有効になりません。この変更には、手動での再起動が必要です。DB インスタンスは、メンテナンスウィンドウ中に自動では再起動されません。

DB インスタンスのアクションと [Apply Immediately] の値を設定することによる効果を示す表については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS DB パラメータの変更が有効にならない

場合によっては、DB パラメータグループのパラメータを変更しても、その変更が有効にならないことがあります。その場合は、DB パラメータグループに関連付けられている DB インスタンスの再起動が必要な可能性があります。動的パラメータを変更する場合は、その変更はすぐに有効になります。静的パラメータを変更する場合は、その変更はパラメータグループに関連付けられている DB インスタンスを再起動するまで有効になりません。

RDS コンソールを使用して DB インスタンスを再起動できます。または、`RebootDBInstance` API オペレーションを明示的に呼び出すこともできます。DB インスタンスがマルチ AZ 配置にある場合は、フェイルオーバーなしで再起動できます。静的パラメータの変更後に関連付けられている DB インスタンスの再起動を求める要件は、パラメータの誤った設定が API コールに影響を及ぼすリスクを低減するために役立ちます。例えば、`ModifyDBInstance` を呼び出して DB インスタンスクラスを変更する場合があります。詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。

Amazon RDS DB インスタンスのストレージ不足

DB インスタンスのストレージ領域が不足すると、DB インスタンスを利用できなくなる可能性があります。CloudWatch で提供される `FreeStorageSpace` メトリクスを常にモニタリングして、DB インスタンスのストレージに十分な空き容量があることを確認することを強くお勧めします。

データベースインスタンスのストレージが不足すると、ステータスが `storage-full` になります。例えば、ストレージを使い果たした DB インスタンスに対して `DescribeDBInstances` API オペレーションを呼び出すと、次のように出力されます。

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

この状態から回復するには、ModifyDBInstance API オペレーションまたは次の AWS CLI コマンドを使用してインスタンスにストレージ容量を追加します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --allocated-storage 60 \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 60 ^
    --apply-immediately
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

ここで DB インスタンスの情報を取得すると、DB インスタンスが modifying ステータスであり、ストレージのスケーリングが行われていることが示されます。

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  50  sa
modifying  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com
3306  us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

ストレージのスケーリングが完了すると、DB インスタンスのステータスは available になります。

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m5.large  mysql8.0  60  sa
available  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql8.0  in-sync
```

DescribeEvents オペレーションを使用すると、ストレージ容量を使い果たしたときに通知を受信できます。例えば、上記のオペレーションの後に DescribeEvents の呼び出しを実行すると、次のような出力が表示されます。

```
aws rds describe-events --source-type db-instance --source-identifier mydbinstance
```

```
2009-12-22T23:44:14.374Z  mydbinstance  Allocated storage has been exhausted db-
instance
2009-12-23T00:14:02.737Z  mydbinstance  Applying modification to allocated storage db-
instance
2009-12-23T00:31:54.764Z  mydbinstance  Finished applying modification to allocated
 storage
```

Amazon RDS で利用可能な DB インスタンスが不足している

InsufficientDBInstanceCapacity エラーは、DB インスタンスを作成、起動、変更を試行した際に返されます。また、DB スナップショットから DB インスタンスを復元する際に返されます。このエラーが返される場合、一般的な原因は、要求された可用性ゾーンで特定の DB インスタンスクラスが利用できないことです。次のいずれかの方法で、問題を解決できます。

- 別の DB インスタンスクラスでリクエストを再試行してください。
- 別のアベイラビリティーゾーンでリクエストを再試行します。
- 明示的なアベイラビリティーゾーンを指定せずにリクエストを再試行してください。

Amazon EC2 インスタンスの容量問題のトラブルシューティングについては、Amazon EC2 ユーザーガイドの「インスタンス容量の不足」を参照してください。

DB インスタンスの変更の詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。

Amazon RDS の解放可能なメモリの問題

解放可能なメモリは、データベースエンジンで使用できる DB インスタンスの合計ランダムアクセスメモリ (RAM) です。これは、空きオペレーティングシステム (OS) メモリと使用可能なバッファとページキャッシュメモリの合計です。データベースエンジンは、ホスト上のほとんどのメモリを使用しますが、OS プロセスも一部の RAM を使用します。データベースエンジンに現在割り当てられているメモリや OS プロセスによって使用されているメモリは、空きメモリには含まれません。データベースエンジンのメモリが不足している場合、DB インスタンスは、バッファリングとキャッシュに通常使用される一時スペースを使用できます。前述のように、この一時スペースは空きメモリに含まれます。

Amazon CloudWatch の `FreeableMemory` メトリクスを使用して、空きメモリを監視します。詳細については、「Amazon RDS のモニタリングツール」を参照してください。

DB インスタンスの解放可能なメモリが常に不足またはスワップ領域を使用する場合、DB インスタンスクラスへのスケールアップを検討する必要があります。詳細については、「 DB インスタンスクラス」を参照してください。

メモリ設定を変更することもできます。例えば、 RDS for MySQL と指定すると、`innodb_buffer_pool_size`パラメータのサイズを調整することもできます。このパラメータはデフォルトで物理メモリの 75% に設定されています。MySQL のトラブルシューティングのヒントについては、「Amazon RDS for MySQL データベースの空きメモリ不足のトラブルシューティング方法を教えてください」を参照してください。

MySQL および MariaDB の問題

MySQL および MariaDB DB インスタンスに関する問題を診断して修正できます。

トピック

- MySQL および MariaDB の最大接続数
- メモリ制限と互換性のないパラメータの状態の診断と解決
- リードレプリカ間の遅延の診断と解決
- MySQL または MariaDB のリードレプリケーションのエラーの診断と解決

- バイナリログが有効な場合に SUPER 権限が必要になるトリガーの作成
- ポイントインタイム復元のエラーの診断と解決
- レプリケーション停止エラー
- 致命的なエラー 1236 によるリードレプリカの作成の失敗またはレプリケーションの中断
- リードレプリカのレプリケーションがメタデータ構造の初期化に失敗する

MySQL および MariaDB の最大接続数

RDS for MySQL または RDS for MariaDB DB インスタンスへの許可されている接続の最大数は、その DB インスタンスクラスで使用できるメモリ量に基づきます。使用できるメモリ量が多い DB インスタンスは、使用できる接続数が多くなります。DB インスタンスクラスの詳細については、「DB インスタンスクラス」を参照してください。

DB インスタンスへの接続の制限は、デフォルトで DB インスタンスクラスの最大値に設定されます。同時接続数は、許可されている最大接続数を上限とする任意の値に制限できます。この制限には、DB インスタンスのパラメータグループの max_connections パラメータを使用します。詳細については、「データベース接続の最大数」および「Amazon RDS のパラメータグループ」を参照してください。

次のクエリを実行すると、MySQL または MariaDB DB インスタンスに許可されている最大接続数を取得できます。

```
SELECT @@max_connections;
```

次のクエリを実行すると、MySQL または MariaDB DB インスタンスへのアクティブな接続数を取得できます。

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

メモリ制限と互換性のないパラメータの状態の診断と解決

次の条件が満たされた場合、MariaDB インスタンスまたは MySQL DB インスタンスは、メモリ制限の incompatible-parameters ステータスになります。

- DB インスタンスのステータスが 利用可能 の状態で、DB インスタンスが 1 時間に 3 回以上または 1 日で 5 回以上再起動される場合。

- メンテナンスアクションまたはモニタリングプロセスが DB インスタンスを再起動できなかったため、DB インスタンスを再起動しようとして失敗した場合。
- DB インスタンスの潜在的なメモリ使用量が、DB インスタンスクラスに割り当てられたメモリの 1.2 倍を超える場合。

DB インスタンスが 1 時間で 3 回再起動するか、1 日で 5 回再起動すると、その時点でメモリ使用量のチェックが実行されます。このチェックで、DB インスタンスの潜在的なメモリ使用量を計算します。計算によって、次の値の合計が返されます。

- 値 1 - 次のパラメータの合計です。
  - innodb_additional_mem_pool_size
  - innodb_buffer_pool_size

    innodb_buffer_pool_size の値は変更することが可能です。ただし、値が入力内容と常に一致するとは限りません。この不一致は、いくつかの理由で発生します。まず、DB インスタンスがマイクロ DB インスタンスの場合は、デフォルト値を上書きして 256 MB に設定します。詳細については、「innodb_buffer_pool_size の上書き」を参照してください。

    次に、ホストマネージャー、エンジン、オペレーティングシステム、カーネル用に DB インスタンスに 500 MB のメモリが予約されていることを確認します。

    最後に、単位に分割して innodb_buffer_pool_size を最適化します。ホストマネージャーは、それらの単位のうち最も近い倍数に切り下げます。単位は、innodb_buffer_pool_chunk_size に innodb_buffer_pool_instances を乗算して計算されます。詳細については、「MySQL ドキュメント」の「Configuring InnoDB Buffer Pool Size」を参照してください。

    innodb_buffer_pool_size が 1 GB 未満でない限り、innodb_buffer_pool_instances のデフォルトは 8 です。innodb_buffer_pool_size が 1 GB 未満の場合、innodb_buffer_pool_instances のデフォルトは 1 です。innodb_buffer_pool_chunk_size のデフォルトは 128 MB です。
  - innodb_log_buffer_size
  - key_buffer_size
  - query_cache_size (MySQL バージョン 5.7 のみ)
  - tmp_table_size
- 値 2 - max_connections パラメータに、次のパラメータの合計を乗算した結果です。

- binlog_cache_size
- join_buffer_size
- read_buffer_size
- read_rnd_buffer_size
- sort_buffer_size
- thread_stack

- 値 3 - performance_schema パラメータが有効な場合は、max_connections パラメータに 429498 を乗算します。

  performance_schema パラメータが無効の場合、この値はゼロになります。

したがって、計算によって返される値は次のとおりです。

```
Value 1 + Value 2 + Value 3
```

この値が、DB インスタンスが使用する DB インスタンスクラスに割り当てられたメモリの 1.2 倍を超えると、DB インスタンスは incompatible-parameters ステータスになります。DB インスタンスクラスに割り当てられるメモリの詳細については、「DB インスタンスクラスのハードウェア仕様」を参照してください。

max_connections パラメータの値に複数のパラメータの合計を乗算して計算します。max_connections パラメータに大きな値が設定されている場合、DB インスタンスの潜在的なメモリ使用量として、非常に高い値をチェックが返す可能性があります。この場合、max_connections パラメータの値を小さくしてみてください。

この問題を解決するには、以下のステップを完了する必要があります。

1. DB インスタンスに関連付けられた DB パラメータグループのメモリパラメータを調整します。調整する際に、潜在的なメモリ使用量が DB インスタンスクラスに割り当てられたメモリの 1.2 倍未満になるようにします。

   パラメータの設定の詳細については、「Amazon RDS の DB パラメータグループのパラメータの変更」を参照してください。
2. DB インスタンスを再起動します。

   パラメータの設定の詳細については、「以前に停止した Amazon RDS DB インスタンスを開始する」を参照してください。

リードレプリカ間の遅延の診断と解決

MySQL または MariaDB のリードレプリカを作成してそのリードレプリカが使用可能になると、Amazon RDS はリードレプリカの作成オペレーションがスタートされた時点以降に出典の DB インスタンスに対して行われた変更を初期にレプリケートします。このフェーズでは、リードレプリカのレプリケーション遅延時間が 0 より大きくなります。これは、Amazon CloudWatch で Amazon RDS の `ReplicaLag` メトリクスを参照することによってモニタリングできます。

`ReplicaLag` メトリクスには、MariaDB または MySQL `Seconds_Behind_Master` コマンドの `SHOW REPLICA STATUS` フィールドの値が報告されます。詳細については、MySQL ドキュメントの「SHOW REPLICA STATUS ステートメント」を参照してください。

`ReplicaLag` メトリクスが 0 に達すると、レプリカがソース DB インスタンスに追いついています。`ReplicaLag` メトリクスが -1 を返す場合、レプリケーションがアクティブではない可能性があります。レプリケーションのエラーをトラブルシューティングする場合は、「MySQL または MariaDB のリードレプリケーションのエラーの診断と解決」を参照してください。`ReplicaLag` の値が -1 である場合は、`Seconds_Behind_Master` の値が特定できないか `NULL` であることも示しています。

> **Note**
>
> MariaDB の旧バージョンは、`SHOW SLAVE STATUS` ではなく `SHOW REPLICA STATUS` を使用していました。10.5 より前の MariaDB バージョンを使用している場合は、`SHOW SLAVE STATUS` を使用します。

`ReplicaLag` メトリクスは、ネットワークが停止しているとき、またはメンテナンス時間にパッチを適用しているときには、-1 を返します。この場合、ネットワーク接続が復元されるか、メンテナンス時間が終了するまで待ってから、`ReplicaLag` メトリクスを再度確認します。

MySQL と MariaDB のリードレプリケーションテクノロジーは非同期です。そのため、出典の DB インスタンスの `BinLogDiskUsage` メトリクスやリードレプリカの `ReplicaLag` メトリクスが増加する場合があります。例えば、出典の DB インスタンスへの大量の書き込みオペレーションがパラレルで実行される場合を例にします。このとき、リードレプリカへの書き込みオペレーションは単一の I/O スレッドでシリアルで行われます。このような場合、出典のインスタンスとリードレプリカの間に遅延が発生する可能性があります。

リードレプリカと MySQL の詳細については、MySQL ドキュメントの「レプリケーション実装の詳細」を参照してください。リードレプリカと MariaDB の詳細については、MariaDB ドキュメントの「レプリケーションの概要」を参照してください。

出典の DB インスタンスに対する更新とそれに続くリードレプリカに対する更新の間の遅延を低減するには、次のような方法があります。

- リードレプリカの DB インスタンスクラスが出典の DB インスタンスと同程度のストレージサイズを持つように設定します。
- 出典の DB インスタンスとリードレプリカによって使用される DB パラメータグループのパラメータ設定の互換性を保ちます。詳細と例については、次のセクションにある `max_allowed_packet` パラメータの説明を参照してください。
- クエリのキャッシュを無効にします。頻繁に変更されるテーブルでは、クエリのキャッシュを使用すると、キャッシュが頻繁にロックされ、更新されるため、レプリカの遅延が増加する可能性があります。このような場合、クエリのキャッシュを無効にすると、レプリカの遅延が小さくなることがあります。DB インスタンスの DB パラメータグループで `query_cache_type parameter` を 0 に設定することによって、クエリのキャッシュを無効にできます。クエリのキャッシュの詳細については、「クエリのキャッシュの設定」を参照してください。
- InnoDB for MySQL または MariaDB のリードレプリカのバッファプールをウォームします。例えば、頻繁に更新される一連の小さなテーブルがあり、InnoDB または XtraDB のテーブルスキーマを使用しているとします。その場合、それらのテーブルをリードレプリカにダンプします。そうすることで、データベースエンジンはそれらのテーブルの行をディスクからスキャンしてバッファプールにキャッシュします。これにより、レプリカの遅延を低減することができます。例を以下に示します。

  Linux、macOS、Unix の場合:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

  Windows の場合:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
```

```
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

MySQL または MariaDB のリードレプリケーションのエラーの診断と解決

Amazon RDS では、リードレプリカのレプリケーションステータスをモニタリングします。RDS では、何らかの理由でレプリケーションが停止した場合、リードレプリカのインスタンスの [Replication State] (レプリケーションステータス) フィールドを `Error` に更新します。MySQL または MariaDB エンジンによりスローされた関連するエラーの詳細は、[レプリケーションエラー] フィールドを参照することで確認できます。リードレプリカのステータスを示すイベントが生成されます (RDS-EVENT-0045、RDS-EVENT-0046、RDS-EVENT-0057 など)。イベントについてとイベントへのサブスクライブの詳細については、「Amazon RDS イベント通知の操作」を参照してください。MySQL のエラーメッセージが返された場合は、MySQL のエラーメッセージのドキュメントでエラーを確認してください。MariaDB のエラーメッセージが返された場合は、MariaDB のエラーメッセージのドキュメントでエラーを確認してください。

レプリケーションエラーを引き起こす可能性がある一般的な状況は次のとおりです。

- リードレプリカの `max_allowed_packet` パラメータの値が出典の DB インスタンスの `max_allowed_packet` パラメータの値より小さい。

  `max_allowed_packet` パラメータは、DB パラメータグループで設定できるカスタムパラメータです。`max_allowed_packet` パラメータは、データベースで実行できるデータ操作言語 (DML) の最大サイズを指定するために使用されます。ソースの DB インスタンスの `max_allowed_packet` の値がリードレプリカの `max_allowed_packet` の値より大きい場合があります。その場合、レプリケーションプロセスはエラーをスローし、レプリケーションを停止する可能性があります。最も一般的なエラーは「`packet bigger than 'max_allowed_packet' bytes`」です。出典とリードレプリカで同じ `max_allowed_packet` パラメータ値を持つ DB パラメータグループが使用されるように設定することにより、エラーを修正できます。
- リードレプリカのテーブルに書き込んでいる。リードレプリカでインデックスを作成する場合、`read_only` パラメータを 0 に設定してインデックスを作成する必要があります。リードレプリカのテーブルに書き込んだ場合、レプリケーションが中断する可能性があります。

- MyISAM などの非トランザクションストレージエンジンを使用している。リードレプリカにはトランザクションストレージエンジンが必要です。レプリケーションは、InnoDB for MySQL または MariaDB のストレージエンジンでのみサポートされています。

  次のコマンドで MyISAM テーブルを InnoDB に変換できます。

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```

- SYSDATE() など、安全でない非決定的クエリを使用している。詳細については、MySQL ドキュメントの「バイナリロギングでの安全および安全でないステートメントの判断」を参照してください。

以下のステップは、レプリケーションエラーを解決するのに役立ちます。

- 論理的なエラーが発生し、安全にエラーをスキップできる場合は、「RDS for MySQL の現在のレプリケーションエラーのスキップ」で説明されているステップに従ってください。MySQL または MariaDB DB インスタンスは、mysql_rds_skip_repl_error プロシージャを含むバージョンを実行している必要があります。詳細については、「mysql.rds_skip_repl_error」を参照してください。
- バイナリログ (binlog) の位置の問題が発生した場合は、mysql.rds_next_source_log (RDS for MySQL メジャーバージョン 8.4 以降) または mysql.rds_next_master_log (RDS for MariaDB および RDS for MySQL メジャーバージョン 8.0 以前) コマンドを使用してレプリカの再生位置を変更できます。
- DML の負荷が高いため、一時的にパフォーマンスの問題が発生する可能性があります。その場合、リードレプリカの DB パラメータグループで innodb_flush_log_at_trx_commit パラメータを 2 に設定します。これを行うことによって、一時的に ACID 特性 (不可分性、整合性、分離性、耐久性の高い) が低下しますが、リードレプリカの遅延を解消するのに役立ちます。
- リードレプリカを削除し、同じ DB インスタンス識別子を使用してインスタンスを作成できます。これを行うと、エンドポイントは前のリードレプリカと同じままになります。

レプリケーションエラーが解決すると、[Replication State] は [replicating] に変化します。詳細については、「MySQL リードレプリカに関する問題のトラブルシューティング」を参照してください。

バイナリログが有効な場合に SUPER 権限が必要になるトリガーの作成

RDS for MySQL または RDS for MriaDB の DB インスタンスにトリガーを作成しようとすると、次のエラーが表示される場合があります。

```
"You do not have the SUPER privilege and binary logging is enabled"
```

バイナリログが有効なときにトリガーを使用するには、RDS for MySQL および RDS for MriaDB の DB インスタンスに制限されている SUPER 権限が必要です。バイナリログが有効なときに SUPER 権限なしでトリガーを作成するには、log_bin_trust_function_creators パラメータを true に設定します。log_bin_trust_function_creators を true に設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

新しい DB パラメータグループを作成することで、バイナリログを有効にして RDS for MySQL または RDS for MriaDB DB インスタンスにトリガーを作成できます。そのためには、次の CLI コマンドを使用します。既存のパラメータグループを変更するには、ステップ 2 からスタートしてください。

CLI を使用して新しいパラメータグループを作成し、バイナリログが有効なトリガーを許可するには

1. 新しいパラメータグループを作成します。

   Linux、macOS、Unix の場合:

   ```
   aws rds create-db-parameter-group \
       --db-parameter-group-name allow-triggers \
       --db-parameter-group-family mysql8.0 \
       --description "parameter group allowing triggers"
   ```

   Windows の場合:

   ```
   aws rds create-db-parameter-group ^
       --db-parameter-group-name allow-triggers ^
       --db-parameter-group-family mysql8.0 ^
       --description "parameter group allowing triggers"
   ```

2. トリガーを許可するように DB パラメータグループを変更します。

   Linux、macOS、Unix の場合:

   ```
   aws rds modify-db-parameter-group \
       --db-parameter-group-name allow-triggers \
       --parameters "ParameterName=log_bin_trust_function_creators,
    ParameterValue=true, ApplyMethod=pending-reboot"
   ```

Windows の場合:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name allow-triggers ^
    --parameters "ParameterName=log_bin_trust_function_creators,
 ParameterValue=true, ApplyMethod=pending-reboot"
```

3. 新しい DB パラメータグループを使用するように DB インスタンスを変更します。

Linux、macOS、Unix の場合:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name allow-triggers \
    --apply-immediately
```

Windows の場合:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name allow-triggers ^
    --apply-immediately
```

4. 変更を有効にするために、手動で DB インスタンスを再起動します。

```
aws rds reboot-db-instance --db-instance-identifier mydbinstance
```

ポイントインタイム復元のエラーの診断と解決

一時テーブルを含む DB インスタンスの復元

MySQL または MriaDB DB インスタンスでポイントインタイム復元 (PITR) を実行しようとすると、次のエラーが発生する場合があります。

```
Database instance could not be restored because there has been incompatible database
 activity for restore
functionality. Common examples of incompatible activity include using temporary tables,
 in-memory tables,
or using MyISAM tables. In this case, use of Temporary table was detected.
```

PITR は、特定の時点への DB インスタンスの復元を MySQL または MariaDB のバックアップスナップショットとバイナリログ (binlog) の両方に依存します。binlog 内の一時テーブル情報は信頼できない場合があり、PITR の障害の原因となる可能性があります。MySQL または MriaDB DB インスタンスで一時テーブルを使用する場合は、より頻繁にバックアップを実行することによって、PITR 障害が発生する可能性を最小限に抑えることができます。PITR の障害が発生する可能性は、一時テーブルの作成から次のバックアップスナップショットまでの間で最も高くなります。

メモリ内テーブルを含む DB インスタンスの復元

メモリ内テーブルが含まれるデータベースを復元するときに問題が発生する可能性があります。メモリ内テーブルは再開時に消去されます。その結果、再起動後、メモリ内テーブルは空である可能性があります。メモリ内テーブルを使用する場合は、再開時に空のテーブルを処理するようにソリューションを設計することをお勧めします。インメモリテーブルをレプリケートされた DB インスタンスで使用している場合は、再起動後にリードレプリカを再作成する必要がある場合があります。これは、リードレプリカが再起動して空のインメモリテーブルからデータを復元できない場合に必要になる場合があります。

バックアップと PITR の詳細については、「バックアップの概要」および「Amazon RDS の DB インスタンスを特定の時点に復元する」を参照してください。

レプリケーション停止エラー

`mysql.rds_skip_repl_error` コマンドを呼び出すと、レプリケーションがダウンまたは無効であることを示すエラーメッセージが表示されることがあります。

このエラーメッセージは、レプリケーションが停止して再開できないために表示されます。

多数のエラーをスキップする必要がある場合は、レプリケーションの遅延により、バイナリログファイルがデフォルトの保持期間を超えて増大する場合があります。この場合、バイナリログファイルがレプリカで再生される前に破棄されるため、致命的なエラーが発生することがあります。この破棄によりレプリケーションが停止し、`mysql.rds_skip_repl_error` コマンドを呼び出してレプリケーションエラーをスキップすることができなくなります。

この問題は、レプリケーション出典でバイナリログファイルの保持時間を増加させることで軽減できます。バイナリログ保持時間を長くすると、レプリケーションを再開し、必要に応じて `mysql.rds_skip_repl_error` コマンドを使用できるようになります。

バイナリログの保持期間を設定するには、mysql.rds_set_configuration プロシージャを使用します。設定パラメータの「binlog retention hours」と DB クラスターでバイナリログファイルを保持する時

間数を最大 720 時間 (30 日) で指定します。以下の例では、バイナリログファイルの保持期間を 48 時間に設定しています。

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

致命的なエラー 1236 によるリードレプリカの作成の失敗またはレプリケーションの中断

MySQL または MariaDB DB インスタンスのデフォルトパラメータ値を変更した後、以下のいずれかの問題が発生する可能性があります。

- DB インスタンスのリードレプリカを作成できない。
- レプリケーションが `fatal error 1236` により失敗します。

MySQL および MariaDB DB インスタンスのいくつかのデフォルトパラメータ値は、データベースが ACID に準拠し、リードレプリカをクラッシュセーフにするために役立ちます。これは、コミットする前にトランザクションをバイナリログに書き込むことによって、各コミットが完全に同期されるようにすることで行います。パフォーマンスを上げるためにこれらのパラメータをデフォルト値から変更すると、トランザクションがバイナリログにまだ書き込まれていない場合にレプリケーションが失敗することがあります。

この問題を解決するには、次のパラメータ値を設定します。

- `sync_binlog = 1`
- `innodb_support_xa = 1`
- `innodb_flush_log_at_trx_commit = 1`

リードレプリカのレプリケーションがメタデータ構造の初期化に失敗する

レプリケーションを開始しようとして、次のエラーメッセージが表示されました。

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

このエラーは、レプリカのメタデータ構造に問題がある場合に発生します。メタデータ構造を修正するには、新しいレプリカを作成する必要があります。

このエラーが今後発生しないようにするには、次のいずれかのアクションを実行します。

- 可能であれば、レプリカのマルチスレッドを無効にします。MySQL 8.0.27 以降、マルチスレッドはデフォルトで有効になっています。
- レプリカでマルチスレッドを使用する必要がある場合は、GTID ベースのレプリケーションを使用することをお勧めします。詳細については、「GTID ベースレプリケーションを使用する」を参照してください。

バックアップ保持期間を 0 に設定できない

バックアップ保持期間を 0 に設定するのには、いくつかの理由があります。例えば、保持期間を 0 に設定すると、自動バックアップをすぐに無効にすることができます。

場合によっては、値を 0 に設定すると、保持期間は 1 から 35 の間にする必要があるというメッセージが表示されることがあります。そのような場合は、インスタンスにリードレプリカが設定されていないことを確認します。リードレプリカでは、リードレプリカのログを管理するためにバックアップが必要なため、保持期間を 0 に設定することはできません。

Amazon RDS API リファレンス

AWS マネジメントコンソール と AWS Command Line Interface (AWS CLI) の他に、Amazon RDS には API も用意されています。API を使用して、DB インスタンスと Amazon RDS の他のオブジェクトを管理するためのタスクを自動化できます。

- API オペレーションのアルファベット順リストについては、「アクション」を参照してください。
- データ型のアルファベット順リストについては、「データ型」を参照してください。
- 共通クエリパラメータのリストについては、「共通パラメータ」を参照してください。
- エラーコードの説明については、「共通エラー」を参照してください。

AWS CLI の詳細については、「Amazon RDS の AWS Command Line Interface リファレンス」を参照してください。

トピック

- クエリ API の使用
- Amazon RDS のアプリケーションのトラブルシューティング

クエリ API の使用

以下のセクションでは、Query API で使用されるパラメータおよびリクエスト認証について簡単に説明します。

Query API の動作に関する一般的な情報については、Amazon EC2 API Referenceの「クエリリクエスト」を参照してください。

クエリパラメータ

HTTP クエリベースのリクエストとは、HTTP 動詞 (GET または POST) とクエリパラメータ `Action` で記述する HTTP リクエストです。

各クエリリクエストに、アクションの認証と選択を処理するための一般的なパラメータがいくつか含まれている必要があります。

オペレーションの中にはパラメータのリストを取るものがあります。これらのリストは、`param.`*n* 表記を使用して指定されます。*n* 値は、1 から始まる整数です。

Amazon RDS のリージョンとエンドポイントの詳細については、Amazon Web Services 全般のリファレンス の「リージョンとエンドポイント」セクションの「Amazon Relational Database Service (RDS)」を参照してください。

クエリリクエストの認証

HTTPS 経由でのみリクエストを送信できます。また、各クエリリクエストには署名を含める必要があります。AWS 署名バージョン 4 または署名バージョン 2 のどちらかを使用してください。詳細については、「署名バージョン 4 の署名プロセス」および「署名バージョン 2 の署名プロセス」を参照してください。

Amazon RDS のアプリケーションのトラブルシューティング

Amazon RDS では、Amazon RDS API とのやり取りで発生する問題をトラブルシューティングする際に役立つ、具体的でわかりやすいエラーを提供します。

トピック

- エラーの取得
- トラブルシューティングのヒント

Amazon RDS DB インスタンスのトラブルシューティングの詳細については、「Amazon RDS のトラブルシューティング」を参照してください。

エラーの取得

通常、アプリケーションでは、結果を処理する前にリクエストでエラーが生成されたかどうかを必ず確認します。エラーが発生したかどうかを確認する最も簡単な方法は、Amazon RDS API からのレスポンスで `Error` ノードを検索することです。

XPath 構文を使用すると、簡単な方法で 1 つの `Error` ノードがあるかどうかを検索します。また、エラーコードとメッセージを比較的簡単に取得できます。次のコードでは、Perl および XML::XPath モジュールによって、リクエスト時のエラーの発生を判定しています。エラーが発生した場合、レスポンス内の最初のエラーコードとメッセージが表示されます。

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
```

```
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

トラブルシューティングのヒント

Amazon RDS API の問題を診断して解決するには、次の手順を実行することをお勧めします。

- http://status.aws.amazon.com を確認し、対象とする AWS リージョンで Amazon RDS が正常に動作しているかどうかを確認します。
- リクエストの構文を確認します。

  『Amazon RDS API リファレンス』には、各 Amazon RDS オペレーションについてのリファレンスページがあります。パラメータを正しく使用していることをもう一度確認してください。間違っている可能性がある部分を判断するヒントとして、同様のオペレーションを実行しているサンプルのリクエストやユーザーシナリオを調べてください。
- AWS re:Post を確認する

  Amazon RDS には開発コミュニティあり、これまでに他のデベロッパーが経験した問題に対する解決策を探すことができます。トピックを表示するには、AWS re:Post に移動してください。

ドキュメント履歴

現在の API バージョン: 2014年 10 月 31 日

以下の表に、2018 年 5 月以降の 『Amazon RDS ユーザーガイド』の各リリースにおける重要な変更点を示します。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

> **Note**
>
> [What's New with Database?] (データベースの新機能) ページで新しい Amazon RDS 機能をフィルタリングできます。[Products] (製品) フィルターで、[Amazon RDS] を選択します。その後、**`RDS Proxy`** や **`Oracle 2023`** などのキーワードを使用して検索します。

変更	説明	日付
Amazon RDS for Oracle が Oracle APEX バージョン 26.1.v1 をサポート	Oracle APEX 26.1.v1 は Oracle Database 19c 以降で使用できます。詳細については、「Oracle Application Express」を参照してください。	2026 年 8 月 14 日
Amazon RDS が MariaDB 12.3 をサポート	MariaDB バージョン 12.3 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2026 年 8 月 7 日
Amazon RDS が、データベースプレビュー環境で MySQL 9.7 をサポート	MySQL 9.7 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL	2026 年 7 月 23 日

	version 9.7 in the Database Preview environment」を参照してください。	
Amazon RDS でボリュームごとのストレージの初期化と最適化のステータスを追加	Amazon RDS は、DescribeDBInstances レスポンスの `StorageOperationStatus` および `StorageOperationPercentProgress` フィールドを通じて、ボリュームごとのストレージオペレーションのステータスと進行状況を報告するようになりました。これらのフィールドは、ポイントインタイムリカバリ、スナップショット復元、リードレプリカ作成、ブルー/グリーンデプロイ、シングル AZ からマルチ AZ への変換、ストレージスケーリングの進行状況を報告します。詳細については、「Amazon RDS DB インスタンスステータスの表示」を参照してください。	2026 年 7 月 20 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 8.0.46-RDS.20260624 および 5.7.44-RDS	RDS for MySQL 向け RDS 延長サポートバージョン 8.0.46-RDS.20260624 および 5.7.44-RDS.20260624 が利用可能になりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2026 年 7 月 15 日

Amazon RDS は MySQL 8.4.10 をサポート	MySQL バージョン 8.4.10 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2026 年 7 月 7 日
Amazon RDS for Oracle が Oracle Database 26ai をサポート	Oracle Database 26ai (26.0.0.0) を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での Oracle Database 26ai」を参照してください。	2026 年 7 月 7 日
Amazon RDS がアジアパシフィック (台北)、アジアパシフィック (ニュージーランド)、アジアパシフィック (タイ)、メキシコ (中部) のバックアップレプリケーションをサポート	バックアップレプリケーションが、アジアパシフィック (台北)、アジアパシフィック (ニュージーランド)、アジアパシフィック (タイ)、メキシコ (中部) リージョンのデータベースで利用可能になりました。詳細については、「別の AWS リージョンへの自動バックアップのレプリケーション」を参照してください。	2026 年 6 月 25 日

2026 年 7 月 1 日以降、Amazon RDS はスナップショットの復元をマグネティックストレージに制限	Amazon RDS はマグネティックストレージを廃止しました。2026 年 7 月 1 日以降、スナップショットをマグネティックストレージに復元できなくなります。マグネティックボリュームのスナップショットを復元する場合は、gp3 や io2 Block Express などの別のストレージタイプを選択する必要があります。詳細については、Amazon RDS DB インスタンスストレージを参照してください。	2026 年 6 月 25 日
RDS for SQL Server は db.x2m インスタンスクラスをサポートします	RDS for SQL Server で db.x2m インスタンスクラスを使用できるようになりました。詳細については、「SQL Server 用 DB インスタンスクラス」を参照してください。	2026 年 6 月 16 日
Amazon RDS が MariaDB 11.8.8、11.4.12、10.11.18、および 10.6.27 をサポート	MariaDB バージョン 11.8.8、11.4.12、10.11.18、10.6.27 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2026 年 6 月 11 日

RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20260521 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2026 年 6 月 4 日
Amazon RDS が MySQL 8.4.9 および 8.0.46 をサポート	MySQL バージョン 8.4.9 および 8.0.46 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2026 年 5 月 11 日
RDS for SQL Server が db.m8a および db.r8a インスタンスクラスをサポート	RDS for SQL Server の db.m8a および db.r8a インスタンスクラスを使用できるようになりました。詳細については、「SQL Server 用 DB インスタンスクラス」を参照してください。	2026 年 5 月 7 日
RDS for SQL Server は db.m8i および db.r8i インスタンスクラスをサポートします	RDS for SQL Server の db.m8i および db.r8i インスタンスクラスを使用できるようになりました。詳細については、「SQL Server 用 DB インスタンスクラス」を参照してください。	2026 年 4 月 30 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.6 をサポート	MySQL 9.6 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL version 9.6 in the Database Preview environment」を参照してください。	2026 年 4 月 16 日
RDS for Oracle は db.m8i および db.r8i インスタンスクラスをサポートします	RDS for Oracle の db.m8i および db.r8i インスタンスクラスを使用できるようになりました。詳細については、「Oracle 用 DB インスタンスクラス」を参照してください。	2026 年 4 月 9 日
RDS Custom for Oracle	サポート終了通知: AWS は、2027 年 3 月 31 日に、Amazon RDS Custom for Oracle のサポートを終了します。詳細については、「RDS Custom for Oracle のサポート終了」を参照してください。	March 31, 2026
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20260212 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2026 年 2 月 26 日

Amazon RDS が MariaDB バージョン 11.8.6、11.4.10、10.11.16、および 10.6.25 をサポート	RDS for MariaDB が MariaDB バージョン 11.8.6、11.4.10、10.11.16 および 10.6.25 をサポートするようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2026 年 2 月 10 日
Amazon RDS が MySQL 8.4.8 および 8.0.45 をサポート	MySQL バージョン 8.4.8 および 8.0.45 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2026 年 2 月 3 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20251212 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2025 年 12 月 12 日
RDS for SQL Server Developer Edition。	RDS for SQL Server は SQL Server Developer Edition をサポートするようになりました。詳細については、「RDS for SQL Server での SQL Server Developer Edition の使用」を参照してください。	2025 年 12 月 2 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.5 をサポート	MySQL 9.5 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL version 9.5 in the Database Preview environment」を参照してください。	2025 年 11 月 21 日
Amazon RDS が MySQL 8.4.7 および 8.0.44 をサポート	MySQL バージョン 8.4.7 および 8.0.44 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2025 年 11 月 18 日
Amazon RDS で MariaDB バージョン 11.4.9、10.11.15、および 10.6.24 をサポート	RDS for MariaDB が MariaDB バージョン 11.4.9、10.11.15、および 10.6.24 をサポートするようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025 年 11 月 18 日

Amazon RDS が Oracle と SQL Server の追加ストレージボリュームをサポート	RDS for Oracle または RDS for SQL Server DB インスタンスには、最大 3 つのストレージボリュームを追加できます。ワークロードの要件に応じて、ボリュームごとに gp3 または io2 ストレージを選択します。プライマリストレージボリュームと追加のストレージボリュームを組み合わせて、DB インスタンスごとに合計ストレージを最大 256 TiB までスケールします。詳細については、「追加のストレージボリューム」を参照してください。	2025 年 11 月 14 日
RDS for Oracle は db.r7i 事前設定インスタンスクラスをサポート	RDS for Oracle は、BYOL を使用する Enterprise Edition および Standard Edition 2 の db.r7i 事前設定インスタンスをサポートするようになりました。これらのインスタンスクラスは、Oracle データベースワークロードに追加のメモリを提供します。例えば、db.r7i.xlarge.tpc2.mem2x は標準の db.r7i.xlarge インスタンスの 2 倍のメモリを提供し、マルチスレッドを有効にします。詳細については、「DB インスタンスクラスのサポート」を参照してください。	2025 年 10 月 2 日

Amazon RDS for Db2 が単一の Db2 データベースのバックアップをサポート	`rdsadmin.backup_database` ストアドプロシージャを使用して、RDS for Db2 DB インスタンスの単一の Db2 データベースを Amazon S3 バケットへバックアップできるようになりました。詳細については、「rdsadmin.backup_database」を参照してください。	2025 年 9 月 30 日
Amazon RDS は、Amazon RDS for Db2 向けにリザーブド DB インスタンスをサポート	RDS for Db2 用にリザーブド DB インスタンスを購入できるようになりました。詳細については、「Amazon RDS 向けリザーブド DB インスタンス」を参照してください。	2025 年 9 月 24 日
Amazon RDS for Db2 はデータベースを復元するために Amazon S3 ストリーミングをサポート	RDS for Db2 で、Amazon S3 バケット内の複数のデータベースバックアップファイルから直接データベースを復元できるようになりました。詳細については、「rdsadmin.restore_database」を参照してください。	2025 年 9 月 22 日
Amazon RDS for Db2 はデータベースを復元するために Amazon S3 ストリーミングをサポート	RDS for Db2 で、Amazon S3 バケット内の複数のデータベースバックアップファイルから直接データベースを復元できるようになりました。詳細については、「rdsadmin.restore_database」を参照してください。	2025 年 9 月 22 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.4 をサポート	MySQL 9.4 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL version 9.4 in the Database Preview environment」を参照してください。	2025 年 9 月 19 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20250818 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2025 年 9 月 15 日
RDS for Oracle が SE2 LI の db.m7i と db.r7i をサポート	ライセンス込み (LI) モデルを使用する RDS for Oracle Standard Edition 2 (SE2) は、アジアパシフィック (タイ) およびメキシコ (中部) リージョンで db.m7i および db.r7i インスタンスクラスをサポートしています。詳細については、「Oracle DB インスタンスクラス」を参照してください。	2025 年 8 月 28 日
Amazon RDS がアジアパシフィック (ニュージーランド) リージョンで使用可能に	Amazon RDS がアジアパシフィック (ニュージーランド) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2025 年 8 月 28 日

RDS for Oracle および RDS Custom for Oracle がベアメタルインスタンスクラスをサポート	RDS for Oracle および RDS Custom for Oracle は、BYOL を使用した Enterprise Edition のベアメタルインスタンスクラスをサポートするようになりました。ベアメタルインスタンスを使用すると、アプリケーションは基盤となるサーバーのプロセッサとメモリリソースに直接アクセスできるようになります。詳細については、「DB インスタンスクラス」および「RDS Custom for Oracle での DB インスタンスクラスのサポート」を参照してください。	2025 年 8 月 27 日
Amazon RDS が MariaDB 11.8 をサポート	MariaDB バージョン 11.8 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025年08月25日

Amazon RDS for Oracle が ECDSA 暗号スイートと ECC 証明書をサポート	Amazon RDS for Oracle は、SSL および OEM エージェントオプションで楕円曲線デジタル署名アルゴリズム (ECDSA) 暗号スイートをサポートするようになりました。これらの暗号スイートは ECC 証明書と互換性があります。詳細については、SSL オプションの「暗号スイートと証明書の互換性」および OEM エージェントオプションの「暗号スイートと証明書の互換性」を参照してください。	2025年08月25日
Amazon RDS for Db2 でクロスリージョン読み取り専用レプリカをサポート	RDS for Db2 のクロスリージョンの読み取り専用レプリカを作成できるようになりました。詳細については、「Amazon RDS for Db2 のレプリカの使用」を参照してください。	2025/08/22
Amazon RDS が MariaDB 11.4.8、10.11.14、10.6.23 をサポート	MariaDB バージョン 11.4.8、10.11.14、10.6.2 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025 年 8 月 14 日

Amazon RDS Custom サービスリンクロールポリシーを更新	AmazonRDSCustomServiceRolePolicy マネージドポリシーは、EBS CopySnapshot 認可動作に準拠するように更新されました。更新では、送信元と送信先のスナップショットのアクセス許可に関する 1 つのステートメントから `ec2:CopySnapshot` を削除し、2 つの新しいステートメントを追加しています。詳細については、「Amazon RDS の AWS 管理ポリシーに関する更新」を参照してください。	2025 年 8 月 7 日
Amazon RDS が MySQL 8.4.6 および 8.0.43 をサポート	MySQL バージョン 8.4.6 および 8.0.43 を実行する Amazon RDS DB インスタンスを作成できるようになりました。MySQL 8.4.6 には、セキュリティ関連のメトリクスを収集する内部 RDS MySQL プラグインである rds_security_audit データベースプラグインが追加されています。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2025 年 8 月 1 日

RDS for Oracle が db.m6in および db.r6in DB インスタンスクラスをサポート	db.m6in および db.r6in DB インスタンスクラスは、RDS for Oracle の Bring-Your-Own-License (BYOL) モデルで使用できます。詳細については、「RDS for Oracle DB インスタンスクラス」を参照してください。	2025 年 7 月 29 日
RDS for PostgreSQL と RDS for Oracle からのゼロ ETL 統合が利用可能に	RDS for PostgreSQL および RDS for Oracle をソースとするゼロ ETL 統合を作成できるようになりました。詳細については、「Amazon RDS ゼロ ETL 統合」を参照してください。	2025 年 7 月 23 日
Amazon RDS for Db2 が Kerberos を使用したグループ認証をサポート	セルフマネージド Active Directory で Kerberos を使用してグループ認証を実行できるようになりました。詳細については、「Amazon RDS for Db2 DB インスタンスの Kerberos 認証を設定する」を参照してください。	2025 年 7 月 21 日
Amazon SageMaker AI とのゼロ ETL 統合が一般利用可能に	ゼロ ETL 統合により、トランザクションデータが RDS for MySQL DB インスタンスに書き込まれてから数秒以内に Amazon SageMaker Lakehouse で使用可能になります。詳細については、「Amazon RDS ゼロ ETL 統合」を参照してください。	2025 年 6 月 30 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.3 をサポート	MySQL 9.3 が米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MySQL バージョン 9.3」を参照してください。	2025 年 6 月 16 日
Amazon RDS for Db2 がスタンバイモードでクロスリージョンレプリカをサポート	スタンバイモードでディザスタリカバリ用の RDS for Db2 クロスリージョンレプリカを作成できるようになりました。詳細については、「RDS for Db2 のレプリカの使用」を参照してください。	2025 年 6 月 11 日
Amazon RDS がアジアパシフィック (台北) リージョンで利用可能に	Amazon RDS がアジアパシフィック (台北) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2025 年 6 月 5 日
Amazon RDS が MariaDB 10.11.13 および 11.4.7 をサポート	MariaDB バージョン 10.11.13 および 11.4.7 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025 年 6 月 4 日

Performance Insights のサポート終了に関する情報	AWS は、Performance Insights のサポート終了日を 2025 年 11 月 30 日と発表しました。この日以降、Amazon RDS は Performance Insights コンソールのエクスペリエンス、柔軟な保持期間 (1 ~ 24 か月)、および関連する料金のサポートを終了します。この日付より前に DB インスタンスを Database Insights にアップグレードするようお勧めします。詳細については、「Amazon RDS での Performance Insights を使用したDB 負荷のモニタリング」を参照してください。	2025 年 5 月 30 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20250508 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2025 年 5 月 27 日
RDS Custom for Oracle は db.m7i および db.r7i インスタンスクラスをサポート	RDS Custom for Oracle の db.m7i および db.r7i インスタンスクラスを使用できるようになりました。詳細については、「RDS Custom for Oracle の DB インスタンスクラスサポート」を参照してください。	2025 年 5 月 22 日

オープンソースエンジンのメジャーバージョンのサポート日の表示	AWS CLI と RDS API を使用して、RDS 標準サポートと RDS 延長サポートにおけるオープンソースエンジンのメジャーバージョンの開始日と終了日に関する情報を表示できるようになりました。詳細については、「Viewing support dates for engine versions in Amazon RDS Extended Support」を参照してください。	2025 年 5 月 20 日
Amazon RDS が MariaDB バージョン 10.5.29 と 10.6.22 をサポート	MariaDB バージョン 10.5.29 および 10.6.22 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025 年 5 月 20 日
Amazon RDS は MySQL 8.0.42 をサポート	MySQL バージョン 8.0.42 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2025 年 5 月 12 日
Amazon RDS は MySQL 8.4.5 をサポート	MySQL バージョン 8.4.5 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2025 年 5 月 1 日

Amazon RDS は、アジアパシフィック (香港)、アジアパシフィック (メルボルン)、アジアパシフィック (マレーシア)、カナダ西部 (カルガリー)、欧州 (チューリッヒ) のバックアップレプリケーションをサポートしています。	バックアップレプリケーションが、アジアパシフィック (香港)、アジアパシフィック (メルボルン)、アジアパシフィック (マレーシア)、カナダ西部 (カルガリー)、欧州 (チューリッヒ) の各リージョンのデータベースで利用可能になりました。詳細については、「別の AWS リージョンへの自動バックアップのレプリケーション」を参照してください。	2025 年 4 月 30 日
Amazon RDS が Oracle マルチテナントアーキテクチャでマネージドマスターユーザーパスワードをサポート	AWS Secrets Manager を使用して、RDS for Oracle コンテナデータベース (CDB) とプラガブルデータベース (PDB) のマスターユーザーパスワードを管理できます。CDB アーキテクチャのシングルテナント設定とマルチテナント設定はどちらもマネージドパスワードをサポートしています。詳細については、「CDB のユーザーアカウントと権限」を参照してください。	2025 年 4 月 24 日

Amazon RDS for Oracle は BYOL の m6id および r6id インスタンスクラスをサポート	BYOL モデルで m6id および r6id インスタンスクラスを使用して RDS for Oracle DB インスタンスを作成できるようになりました。Enterprise Edition 2 および Standard Edition 2 の両方がサポートされています。詳細については、「RDS for Oracle でサポートされているエディション、インスタンスクラス、ライセンスの組み合わせ」を参照してください。	2025 年 4 月 9 日
Amazon RDS が、データベースプレビュー環境で MariaDB 11.8 をサポート	MariaDB 11.8 が、米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MariaDB バージョン 11.8」を参照してください。	2025 年 4 月 2 日
IAM サービスリンクロール許可に対する更新	`AmazonRDSCustomServiceRolePolicy` ポリシーが、EC2 キーペアを管理し、Amazon SQS と統合するための追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2025 年 3 月 25 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.2 をサポート	MySQL 9.2 は、米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL version 9.2 in the Database Preview environment」を参照してください。	2025 年 3 月 24 日
、、 – 既存のポリシーの更新	Amazon RDS では、RDS Custom インスタンスでの RDS Custom マネージドシークレットの使用を許可する新しいアクセス許可がマネージドポリシー `AmazonRDSCustomInstanceProfileRolePolicy` に追加されました。詳細については、「AWS マネージドポリシー: AmazonRDSCustomInstanceProfileRolePolicy」を参照してください。	2025 年 3 月 20 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20250213 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2025 年 3 月 12 日

IAM サービスリンクロール許可に対する更新	AmazonRDSCustomServiceRolePolicy ポリシーが、Secrets Manager シークレットを一覧表示および復元するための追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2025 年 3 月 6 日
Amazon RDS for Db2 が db.m7i および db.r7i インスタンスクラスをサポート	RDS for Db2 の db.m7i および db.r7i インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスタイプ」を参照してください。	2025 年 2 月 27 日

Amazon RDS for Oracle が Spatial パッチバンドル (SPB) をサポート	RDS for Oracle は、Oracle Database 19c の Spatial パッチバンドル (SPB) エンジンバージョンを公開するようになりました。SPB エンジンバージョンには、リリースアップデート (RU) パッチと Oracle Spatial に固有のパッチが含まれています。通常、SPB は対応する RU の 2～3 週間後にリリースされます。DB インスタンスを標準 RU から、同じ DB エンジンバージョン以上の SPB にアップグレードできます。詳細については、「Oracle のマイナーバージョンのアップグレード」を参照してください。	2025 年 2 月 25 日
Amazon RDS が MariaDB 11.4.5、10.11.11、10.6.21、および 10.5.28 をサポート	MariaDB バージョン 11.4.5、10.11.11、10.6.21、10.5.28 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2025 年 2 月 24 日

Amazon RDS が CloudWatch Database Insights をサポート	Database Insights を使用して Amazon RDS データベースをモニタリングできるようになりました。詳細については、「Monitoring Amazon RDS databases with CloudWatch Database Insights」を参照してください。	2025 年 2 月 20 日
Amazon RDS が MySQL 8.4.4 および MySQL 8.0.41 をサポート	MySQL バージョン 8.4.4 および 8.0.41 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2025 年 2 月 19 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20250103 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2025 年 2 月 13 日
Amazon RDS がメキシコ (中部) リージョンで使用可能に	Amazon RDS がメキシコ (中部) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2025 年 1 月 14 日

Amazon RDS が、データベースプレビュー環境で MariaDB 11.7 をサポート	MariaDB 11.7 が、米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MariaDB バージョン 11.7」を参照してください。	2025 年 1 月 9 日
Amazon RDS がアジアパシフィック (タイ) リージョンで使用可能に	Amazon RDS がアジアパシフィック (タイ) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2025 年 1 月 7 日
Amazon RDS が MariaDB 11.4.4、10.11.10、10.6.20、および 10.5.27 をサポート	MariaDB バージョン 11.4.4、10.11.10、10.6.20、および 10.5.27 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2024 年 12 月 20 日
Amazon RDS for Db2 が複数のデータベースをサポート	RDS for Db2 DB インスタンスには、最大 50 個の Db2 データベースを追加できます。詳細については、「Multiple databases on an Amazon RDS for Db2 DB instance」を参照してください。	2024 年 12 月 20 日

Amazon RDS が、データベースプレビュー環境で MySQL 9.1 をサポート	MySQL 9.1 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「MySQL version 9.1 in the Database Preview environment」を参照してください。	2024 年 12 月 19 日
Amazon RDS が db.m8g および db.r8g インスタンスクラスをサポート	RDS for MySQL、RDS for MariaDB、および RDS for PostgreSQL DB インスタンスで、db.m8g および db.r8g インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスタイプ」を参照してください。	2024 年 11 月 21 日
Amazon RDS は MySQL 8.4 をサポート	MySQL バージョン 8.4 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2024 年 11 月 21 日

Amazon Relational Database Service は EC2 データベースの自動移行をサポート	RDS コンソールを使用して、EC2 データベースを Amazon RDS に移行できます。Amazon RDS は AWS Database Migration Service (AWS DMS) を使用してソース EC2 データベースを移行します。AWS DMS では、リレーショナルデータベースを AWS クラウド に移行できます。詳細については、「Auto migrating EC2 databases to Amazon Relational Database Service using AWS Database Migration Service」を参照してください。	2024 年 11 月 20 日
Amazon RDS が MariaDB バージョン 11.8.5 をサポート	RDS for MariaDB が MariaDB バージョン 11.8.5 をサポートするようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2024 年 11 月 20 日
Amazon RDS ブルー/グリーンデプロイがストレージのスケーリングをサポート	Amazon RDS ブルー/グリーンデプロイを使用することで、リソースの割り当てが最適化されるように、グリーン環境のストレージ設定を調整できます。詳細については、「ブルー/グリーンデプロイを作成するときの変更を指定」を参照してください。	2024 年 11 月 20 日

Amazon RDS ブルー/グリーンデプロイにストレージの初期化を追加	Amazon RDS は、ブルー/グリーンデプロイのグリーン環境でストレージの初期化をサポートします。可用性に影響を与えることなく、初回使用時からボリュームパフォーマンスを向上させます。詳細については、「ブルー/グリーンデプロイを作成する際の遅延読み込みの処理」を参照してください。	2024 年 11 月 20 日
Amazon RDS for Oracle は BYOL の db.m7i および db.r7i インスタンスクラスをサポートします	RDS for Oracle の db.m7i および db.r7i インスタンスクラスは、BYOL ライセンスモデルで使用できます。Oracle Database のすべてのエディションがサポートされています。詳細については、「DB インスタンスクラスタイプ」および「サポートされている RDS for Oracle DB インスタンスクラス」を参照してください。	2024 年 11 月 18 日
Amazon RDS Proxy が RDS for PostgreSQL 17 をサポート	RDS Proxy for PostgreSQL 17 DB インスタンスを使用してプロキシを作成できるようになりました。詳細については、「Amazon RDS Proxy の使用」を参照してください。	2024 年 11 月 15 日

Amazon RDS は MySQL 8.0.40 をサポート	MySQL バージョン 8.0.40 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2024 年 11 月 13 日
Amazon RDS がアフリカ (ケープタウン) とアジアパシフィック (ハイデラバード) のバックアップレプリケーションをサポート	アフリカ (ケープタウン) およびアジアパシフィック (ハイデラバード) リージョンのデータベースで、バックアップレプリケーションが使用可能になりました。詳細については、「別の AWS リージョンへの自動バックアップのレプリケーション」を参照してください。	2024 年 11 月 1 日
Amazon RDS は db.m7i および db.r7i インスタンスクラスをサポートします	RDS for MySQL、RDS for PostgreSQL、および RDS for MariaDB のインスタンスクラスで、db.m7i および db.r7i インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスタイプ」を参照してください。	2024 年 10 月 29 日
Amazon RDS for Oracle が Oracle APEX バージョン 24.1.v1 をサポート	Oracle APEX 24.1.v1 は Oracle Database 19c 以降で使用できます。詳細については、「Oracle Application Express」を参照してください。	2024 年 10 月 22 日

マルチ AZ DB クラスターが IAM データベース認証をサポート	マルチ AZ DB クラスターを、AWS Identity and Access Management (IAM) データベース認証を使用して認証可能になりました。詳細については、「マルチ AZ DB クラスターを作成するための設定」を参照してください。	2024 年 10 月 17 日
Amazon RDS が MariaDB 11.4 をサポート	MariaDB バージョン 11.4 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2024 年 10 月 15 日
Amazon RDS が Console-to-Code をサポート	Console-to-Code を使用して、RDS コンソールで実行するアクションからコードを生成できるようになりました。生成されたコードは、他の AWS のサービスの使用を自動化するためのコードを記述するのに役立ちます。詳細については、「Console-to-Code を使用して Amazon RDS コンソールアクションのコードを生成する」を参照してください。	2024 年 10 月 3 日

Amazon RDS for Oracle は Oracle Management Agent 13.5.0.0.v2 をサポートします	RDS は Oracle Management Agent 13.5.0.0.v2 をサポートしています。これには Oracle Management Service (OMS) 13.5.0.23 が必要です。詳細については、「Management Agent の要件」を参照してください。	2024 年 9 月 25 日
Amazon Redshift とのゼロ ETL 統合が一般利用可能に	ゼロ ETL 統合により、トランザクションデータが RDS for MySQL DB インスタンスに書き込まれてから数秒以内に Amazon Redshift で使用可能になります。この機能が一般公開されました。詳細については、「Amazon Redshift との Amazon RDS ゼロ ETL 統合での作業」を参照してください。	2024 年 9 月 12 日
Amazon RDS が MariaDB 10.11.9、10.6.19、10.5.26 をサポート	MariaDB バージョン 10.11.9、10.6.19、10.5.26 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Management Agent の要件」を参照してください。	2024 年 9 月 4 日

Amazon RDS for Oracle が CDB アーキテクチャの OEM、OEMAGENT、OLS オプションをサポートするようになりました	RDS for Oracle CDB インスタンスで Oracle Enterprise Manager と Oracle Label Security を使用できるようになりました。詳細については、「Oracle Enterprise Manager」および「Oracle Label Security」を参照してください。	2024 年 9 月 4 日
RDS Custom for SQL Server が追加のリージョンで利用可能に	RDS Custom for SQL Server が、米国西部 (北カリフォルニア) リージョン、アジアパシフィック (大阪) リージョン、欧州 (パリ) リージョンで利用可能になりました。詳細については、「RDS Custom でサポートされているリージョンと DB エンジン」を参照してください。	2024 年 8 月 29 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20240808 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2024 年 8 月 29 日
Amazon RDS がアジアパシフィック (マレーシア) リージョンで利用可能に	Amazon RDS がアジアパシフィック (マレーシア) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2024 年 8 月 22 日

Amazon RDS は MySQL 8.0.39 をサポート	MySQL バージョン 8.0.39 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2024 年 8 月 12 日
、、 – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSPreview サービスにリンクされたロールの AmazonRDSPreviewServiceRolePolicy から sns:Publish 許可を削除しました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 8 月 7 日
、、 – 既存のポリシーの更新	Amazon RDS は、AWSServiceRoleForRDSBeta サービスにリンクされたロールの AmazonRDSBetaServiceRolePolicy から sns:Publish 許可を削除しました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 8 月 7 日

Amazon RDS が、データベースプレビュー環境で MySQL 8.4 をサポート	MySQL 8.4 は、米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MySQL バージョン 8.4」を参照してください。	2024 年 8 月 1 日
IAM サービスリンクロール許可に対する更新	AmazonRDSCustomServiceRolePolicy ポリシーは、別の AWS リージョンの Amazon RDS サービスと通信し、EC2 イメージをコピーするための追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 7 月 18 日
Amazon RDS が、データベースプレビュー環境で MariaDB 11.4 をサポート	MariaDB 11.4 が、米国東部 (オハイオ) AWS リージョンのデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境における MariaDB バージョン 11.4」を参照してください。	2024 年 7 月 18 日

MySQL 用 AWS ODBC ドライバーが一般利用可能に	MySQL 用 Amazon Web Services (AWS) ODBC ドライバーは、RDS for MySQL の高可用性を実現するために設計されたクライアントドライバーです。詳細については、「MySQL 用 Amazon Web Services (AWS) ODBC ドライバーによる RDS for MySQL への接続」を参照してください。	2024 年 7 月 18 日
、、 – 既存のポリシーの更新	Amazon RDS は、`AWSServiceRoleForRDS` サービスにリンクされたロールの `AmazonRDSServiceRolePolicy` から `sns:Publish` 許可を削除しました。詳細については、「AWS 管理ポリシー: Amazon RDSServiceRolePolicy」を参照してください。	2024 年 7 月 2 日
Db2 の AWS Marketplace プライベートオファー	AWS Marketplace は、Amazon RDS for Db2 の AWS Marketplace を通じて Db2 ライセンスのプライベートオファーをサポートするようになりました。詳細については、「プライベートオファーを取得する」を参照してください。	2024 年 7 月 1 日

マルチ AZ DB クラスタースナップショットデータを Amazon S3 バケットにエクスポートする	マルチ AZ DB クラスタースナップショットデータを Amazon S3 にエクスポートできるようになりました。詳細については、「Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。	2024 年 6 月 27 日
Amazon RDS for Oracle は、事前設定された r6i メモリ最適化インスタンスクラスをサポート	db.r6i Oracle DB インスタンスクラスは、vCPU ごとに追加のメモリ、ストレージ、および I/O を必要とするワークロード向けに最適化されています。例えば、db.r6i.8xlarge.tpc2.mem4x ではマルチスレッドがオンになっており、db.r6i.8xlarge の 4 倍のメモリを提供します。詳細については、「RDS for Oracle インスタンスクラス」を参照してください。	2024 年 6 月 21 日
RDS for MySQL 向け Amazon RDS 延長サポートバージョン 5.7.44-RDS	RDS 延長サポートバージョン 5.7.44-RDS.20240529 が、RDS for MySQL で利用できるようになりました。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2024 年 6 月 20 日

Amazon RDS は MySQL 8.0.37 をサポート	MySQL バージョン 8.0.37 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2024 年 6 月 18 日
Amazon RDS は、MariaDB 10.11.8、10.6.18、10.5.25、および 10.4.34 をサポート	MariaDB バージョン 10.11.8、10.6.18、10.5.25、および 10.4.34 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2024 年 6 月 14 日
Amazon RDS は、db.m4、db.r4、および db.t2 DB インスタンスクラスのサポートを終了	RDS for MariaDB、RDS for MySQL、および RDS for PostgreSQL DB エンジンについて、db.m4、db.r4、および db.t2 インスタンスクラスを使用する DB インスタンスを作成できなくなりました。RDS は、これらのクラスを使用する既存の DB インスタンスを新しい世代に自動的にアップグレードします。詳細については、「DB インスタンスクラスタイプ」を参照してください。	2024 年 6 月 4 日

マルチ AZ DB クラスターは、追加の AWS リージョン で利用可能	マルチ AZ DB クラスターは、より多くの AWS リージョンで作成できます。サポートされるすべてのリージョンを示す表については、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。	2024 年 5 月 29 日
一般提供されている AWS Python ドライバー	Amazon Web Services (AWS) Python ドライバーは、高度な Python ラッパーとして設計されています。このラッパーは、オープンソースの Psycopg ドライバーの機能を補完し、拡張します。詳細については、「AWS ドライバーを使用した DB インスタンスへの接続」を参照してください。	2024 年 5 月 23 日
RDS Proxy がさらに多くのリージョンで利用可能	RDS Proxy は、アジアパシフィック (ハイデラバード)、アジアパシフィック (メルボルン)、中東 (アラブ首長国連邦)、イスラエル (テルアビブ)、カナダ西部 (カルガリー)、欧州 (チューリッヒ) の各リージョンで利用可能になりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2024 年 5 月 21 日

経由の Db2 ライセンスAWS Marketplace	AWS Marketplace 経由の Db2 ライセンスでは、Amazon RDS for Db2 の Db2 ライセンスのサブスクリプション料金を時間単位で支払うことができるようになりました。詳細については、「Amazon RDS for Db2 のライセンスオプション」を参照してください。	2024 年 5 月 21 日
Amazon RDS が Performance Insights への詳細なアクセスをサポート	Performance Insights で個々のディメンションへのアクセスを許可または拒否できるようになりました。この詳細なアクセスコントロールは、`GetResourceMetrics` 、`DescribeDimensionKeys` 、および `GetDimensionKeyDetails` アクションで使用できます。詳細については、「Performance Insights への詳細なアクセス許可の付与」を参照してください。	2024 年 5 月 21 日
RDS for MySQL のバージョンの Amazon RDS 延長サポート	RDS for MySQL バージョンの RDS 延長サポートのすべてのリリースを表示できます。詳細については、「RDS for MySQL のバージョンの Amazon RDS 延長サポート」を参照してください。	2024 年 5 月 16 日

Amazon RDS が、データベースプレビュー環境で MySQL 8.3 をサポート	MySQL 8.3 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MySQL バージョン 8.3」を参照してください。	2024 年 4 月 30 日
Amazon RDS for Db2 がタイムゾーンをサポート	RDS for Db2 は、新しい RDS for Db2 DB インスタンスのローカルタイムゾーンの設定をサポートするようになりました。詳細については、「Amazon RDS for Db2 DB インスタンスのローカルタイムゾーン」を参照してください。	2024 年 4 月 25 日
IAM サービスリンクロール許可に対する更新	`AmazonRDSCustomServiceRolePolicy` ポリシーでは、サービスロールをインスタンスプロファイルとして RDS Custom インスタンスに関連付けるための追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 4 月 19 日

Amazon RDS for Oracle がすべての AWS リージョンで Oracle Data Guard スイッチオーバーをサポート	サポートされているすべてのリージョンで Oracle Data Guard スイッチオーバーを使用できるようになりました。詳細については、「Oracle Data Guard スイッチオーバーの概要」を参照してください。	2024 年 4 月 16 日
RDS Custom for Oracle が Oracle Standard Edition 2 をサポート	Standard Edition 2 を使用して、Oracle Database 12c Release 1 (12.1)、12c Release 2 (12.2)、18c、および 19c で DB インスタンスを作成できるようになりました。CDB と非 CDB の両方を作成できます。詳細については、「RDS Custom for Oracle のエディションとライセンスのサポート」を参照してください。	2024 年 4 月 11 日
Amazon RDS for Oracle が Oracle APEX バージョン 23.2.v1 をサポート	Oracle APEX 23.2.v1 は、Oracle Database 19c 以降で使用できます。詳細については、「Oracle Application Express」を参照してください。	2024 年 4 月 11 日

Amazon RDS Custom サービスリンクロールのアクセス許可の更新	AmazonRDSCustomServiceRolePolicy は、RDS Custom for SQL Server に対して EC2 インスタンスタイプ情報を取得して DB ホストインスタンスタイプを変更することを許可する追加のアクセス許可を付与するようになりました。詳細については、「AWS マネージドポリシーの更新」を参照してください。	2024 年 4 月 8 日
Amazon RDS Custom for Oracle が db.x2iezn DB インスタンスクラスをサポート	RDS Custom for Oracle DB インスタンスで、db.x2iezn インスタンスクラスを使用できるようになりました。詳細については、「RDS Custom for Oracle の DB インスタンスクラスサポート」を参照してください。	2024 年 3 月 26 日
Amazon RDS がマルチ AZ DB クラスターで db.c6gd インスタンスクラスをサポート	マルチ AZ DB クラスターのデプロイで db.c6gd インスタンスクラスを使用できるようになりました。詳細については、「マルチ AZ DB クラスターで利用できるインスタンスクラス」を参照してください。	2024 年 3 月 21 日

Amazon RDS 延長サポート	RDS for MySQL 5.7 または RDS for PostgreSQL 11 データベースを作成または復元すると、自動的にデータベースが Amazon RDS 延長サポートに登録されるようになりました。これにより、既存のアプリケーションはそのまま引き続き動作します。RDS 延長サポートをオプトアウトして、データベースエンジンの RDS 標準サポート終了日後の料金を避けることができます。詳細については、「Amazon RDS 延長サポートの使用」を参照してください。	2024 年 3 月 21 日
と RDS for Db2 の統合AWS License Manager	RDS for Db2 が AWS License Manager と統合されました。Bring-Your-Own-License (BYOL) モデルを使用している場合は、AWS License Manager との統合により、組織内での Db2 ライセンスの使用状況のモニタリングが容易になります。詳細については、「AWS License Manager との統合」を参照してください。	2024 年 3 月 20 日

マルチ AZ DB クラスターの CA 証明書のローテーション	マルチ AZ DB クラスターの CA 証明書のローテーションができるようになりました。新しい CA 証明書 rds-ca-rsa2048-g1、rds-ca-rsa4096-g1、または rds-ca-ecc384-g1 のいずれかの使用を検討してください。詳細については、「SSL/TLS 証明書のローテーション」を参照してください。	2024 年 3 月 6 日
Amazon RDS が io2 Block Express ストレージをサポート	io2 Block Express ストレージタイプを使用する RDS DB インスタンスを作成できるようになりました。詳細については、「io2 Block Express ストレージ」を参照してください。	2024 年 3 月 6 日
RDS Custom for SQL Server が db.r5b and db.x2iedn DB インスタンスクラスをサポート	RDS Custom for SQL Server DB インスタンスで、db.r5b および db.x2iedn インスタンスクラスを使用できるようになりました。詳細については、「RDS Custom for SQL Server の DB インスタンスクラスサポート」を参照してください。	2024 年 3 月 4 日

RDS Custom for Oracle が中東 (UAE) リージョンで利用可能に	中東 (UAE) リージョンで RDS Custom for Oracle DB インスタンスを作成できます。サポートされているすべての AWS リージョン を示した表については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2024 年 3 月 4 日
新しい AWS マネージドポリシー	Amazon RDS は、RDS Custom が EC2 インスタンスプロファイルを介して自動化アクションとデータベース管理タスクを実行できるように `AmazonRDSCustomInstanceProfileRolePolicy` という名前の新しいマネージドポリシーを追加しました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 2 月 27 日
Amazon RDS が MariaDB 10.11.7、10.6.17、10.5.24、10.4.33 をサポート	MariaDB バージョン 10.11.7、10.6.17、10.5.24、10.4.33 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2024 年 2 月 26 日

Amazon RDS マルチ AZ DB クラスターが Amazon EBS gp3 ストレージボリュームをサポート	マルチ AZ DB クラスターが gp3 SSD ベースの EBS ボリュームをサポートするようになりました。詳細については、「gp3 ストレージ」を参照してください。	2024 年 2 月 26 日
イスラエル (テルアビブ) リージョンでの Amazon RDS の AWS Secrets Manager へのサポート	Amazon RDS は、イスラエル (テルアビブ) リージョンにおいて Secrets Manager をサポートしています。詳細については、「Amazon RDS と AWS Secrets Manager によるパスワード管理」を参照してください。	2024 年 2 月 21 日
Amazon RDS for Db2 が監査ログ記録をサポート	RDS for Db2 で、データベースレベルの監査ログ記録がサポートされるようになりました。RDS for Db2 データベースの監査ログ記録を有効にすると、Amazon RDS によってデータベースアクティビティが記録され、Amazon S3 に監査ログが保存されます。詳細については、「Db2 監査ログ記録」を参照してください。	2024 年 2 月 15 日

Amazon RDS 延長サポート	DB クラスターとグローバルクラスター内の RDS for MySQL と RDS for PostgreSQL のメジャーエンジンバージョンが RDS の標準サポート終了日に達すると、Amazon RDS によって自動的に Amazon RDS 延長サポートが有効化されるようになりました。詳細については、「Amazon RDS 延長サポートの使用」を参照してください。	2024 年 2 月 15 日
Amazon RDS が MySQL 8.0.36 をサポート	MySQL バージョン 8.0.36 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2024 年 2 月 12 日
Amazon RDS が RDS for Db2 の EBCDIC 照合をサポート	EBCDIC 照合順序を使用してデータベース内のコンテンツをソートする Db2 データベースを作成できるようになりました。詳細については、「Amazon RDS での Db2 データベースの EBCDIC 照合」を参照してください。	2024 年 1 月 29 日
デフォルトの CA 証明書の更新	デフォルトの CA 証明書は `rds-ca-rsa2048-g1` に設定されます。詳細については、「SSL/TLS を使用して DB インスタンスへの接続を暗号化する」を参照してください。	2024 年 1 月 26 日

Amazon RDS for PostgreSQL が、PL/Rust の 2 つの新しいクレート、cloaring-rs と num-bigint をサポート	Amazon RDS for PostgreSQL では、2 つの新しいクレートを使用できます。詳細については、「PL/Rust でのクレートの使用」を参照してください。	2024 年 1 月 24 日
Amazon RDS for PostgreSQL が TLS バージョン 1.3 をサポート	RDS for PostgreSQL で Transport Layer Security (TLS) バージョン 1.3 を使用できます。詳細については、「PostgreSQL DB インスタンスで SSL を使用する」を参照してください。	2024 年 1 月 24 日
RDS Custom for SQL Server が Microsoft SQL Server 2022 をサポート	SQL Server 2022 を使用する RDS Custom for SQL Server DB インスタンスを作成できるようになりました。詳細については、「RDS Custom for SQL Server の使用」を参照してください。	2024 年 1 月 22 日
AWS マネージドポリシーアクセス許可の更新	`AWSServiceRoleForRDS` サービスリンクロールの `AmazonRDSServiceRolePolicy` に新しいステートメント ID を追加しました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2024 年 1 月 19 日

RDS Custom for Oracle が欧州 (パリ) リージョンをサポート	欧州 (パリ) リージョンで RDS Custom for Oracle DB インスタンスを作成できます。詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2024 年 1 月 18 日
Amazon RDS for MySQL がマルチソースレプリケーションをサポート	RDS for MySQL DB インスタンスでマルチソースレプリケーションを使用できるようになりました。詳細については、「RDS for MySQL でのマルチソースレプリケーションの設定」を参照してください。	2024 年 1 月 16 日
Amazon RDS が、データベースプレビュー環境で MySQL 8.2 をサポート	MySQL 8.2 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MySQL バージョン 8.2」を参照してください。	2024 年 1 月 11 日
RDS Proxy が 欧州 (スペイン) リージョンで利用可能に	RDS Proxy が 欧州 (スペイン) リージョンで利用可能になりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2024 年 1 月 8 日

Amazon EKS がカナダ西部 (カルガリー) リージョンで使用可能に	Amazon RDS がカナダ西部 (カルガリー) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2023 年 12 月 20 日
Amazon RDS for Db2 が 5,000 人のローカルユーザーをサポート	最大 5,000 人のローカルユーザーを権限リストに追加できるようになりました。詳細については、「rdsadmin.add_user」を参照してください。	2023 年 12 月 20 日
Amazon RDS が推奨事項の表示と対応をサポート	Amazon RDS の推奨事項に、RDS for PostgreSQL についてしきい値ベースのプロアクティブ推奨事項と機械学習ベースのリアクティブ推奨事項が含まれるようになりました。詳細については、「Amazon RDS の推奨事項の表示とこれらに対する対応」を参照してください。	2023 年 12 月 19 日
Amazon RDS が MariaDB 10.11.6、10.6.16、10.5.23、および 10.4.32 をサポート	MariaDB バージョン 10.11.6、10.6.16、10.5.23、および 10.4.32 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2023 年 12 月 12 日

Amazon RDS が Amazon Redshift とのゼロ ETL 統合を導入 (プレビュー)	ゼロ ETL 統合は、トランザクションデータが RDS for MySQL DB インスタンスに書き込まれてから数秒以内に Amazon Redshift で使用可能にするためのフルマネージドソリューションです。詳細については、「Amazon Redshift との Amazon RDS ゼロ ETL 統合の操作 (プレビュー)」を参照してください。	2023 年 11 月 28 日
Amazon RDS が IBM Db2 データベースエンジンをサポート	Amazon RDS で IBM Db2 データベースエンジンを実行できるようになりました。詳細については、「Amazon RDS for Db2」を参照してください。	2023 年 11 月 27 日
RDS for PostgreSQL が、PostgreSQL 16.1 へのメジャーバージョンアップグレードと、15.5、14.10、13.13、12.17、および 11.22 へのマイナーバージョンアップグレードをサポート	RDS for PostgreSQL では、DB エンジンをメジャーバージョン 16.1 にアップグレードし、15.5、14.10、13.13、12.17、および 11.22 にマイナーバージョンアップグレードできるようになりました。詳細については、「Amazon RDS の PostgreSQL DB エンジンのアップグレード」を参照してください。	2023 年 11 月 17 日

RDS Custom for Oracle がオプショングループをサポート	オプショングループを作成または変更して、RDS Custom for Oracle DB インスタンスに関連付けることができます。`Timezone` オプションがサポートされるようになりました。詳細については、「RDS Custom for Oracle でのオプショングループの使用」を参照してください。	2023 年 11 月 17 日
Amazon RDS for MySQL がグループレプリケーションプラグインをサポート	MySQL コミュニティによって開発および保守されているグループレプリケーションプラグインを使用して、RDS for MySQL バージョン 8.0.35 以降の DB インスタンスでアクティブ/アクティブクラスターを設定できるようになりました。詳細については、「RDS for MySQL のアクティブ/アクティブクラスターの設定」を参照してください。	2023 年 11 月 17 日
Amazon RDS Proxy が RDS for PostgreSQL 16.1 をサポート	RDS Proxy for PostgreSQL 16.1 DB インスタンスを使用してプロキシを作成できるようになりました。詳細については、「Amazon RDS Proxy の使用」を参照してください。	2023 年 11 月 17 日

RDS Custom for SQL Server が Microsoft SQL Server 2019 Developer エディションをサポート	SQL Server 2019 Developer エディションを使用する RDS Custom for SQL Server DB インスタンスを作成できます。詳細については、「RDS Custom for SQL Server での Bring Your Own Media」を参照してください。	2023 年 11 月 16 日
最小限のダウンタイムでマルチ AZ DB クラスターのマイナーバージョンアップグレード	マルチ AZ DB クラスターのマイナーバージョンアップグレードを実行すると、Amazon RDS はライターインスタンスの前にリーダー DB インスタンスをアップグレードするようになり、ダウンタイムを大幅に短縮できるようになりました。RDS Proxy を使用すると、ダウンタイムをさらに 1 秒以下に短縮できます。詳細については、「RDS for PostgreSQL マルチ AZ DB クラスターのアップグレード」を参照してください。	2023 年 11 月 16 日
RDS for SQL Server が Microsoft SQL Server 2022 をサポート	SQL Server 2022 を使用する RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での Microsoft SQL Server バージョン」を参照してください。	2023 年 11 月 15 日

RDS for MySQL がスナップショットのバージョン 5.7 から 8.0 へのアップグレードをサポート	RDS for MySQL スナップショットのエンジンバージョンをバージョン 5.7 からバージョン 8.0 にアップグレードできるようになりました。この操作は、AWS マネジメントコンソール、または RDS API の `ModifyDBSnapshot` 操作、または AWS CLI を使用して行うことができます。詳細については、「MySQL DB スナップショットのエンジンバージョンのアップグレード」を参照してください。	2023 年 11 月 15 日
RDS Custom for SQL Server が 1,000 データベースのポイントインタイムリカバリをサポート	RDS Custom for SQL Server DB インスタンスで、フルバックアップとポイントインタイムリカバリの対象となるデータベースを最大 1,000 個作成できるようになりました。詳細については、「RDS Custom for SQL Server インスタンス特定時点に復元する」を参照してください。	2023 年 11 月 15 日

RDS Custom for SQL Server がサービスマスターキーの使用をサポート	RDS Custom for SQL Server が、サービスマスターキー (SMK) の使用をサポートするようになりました。SMK を使用すると、認証情報などのオブジェクトを暗号化し、TDE や列暗号化などの SQL Server 機能を使用できます。詳細については、「RDS Custom for SQL Server でのサービスマスターキーの使用」を参照してください。	2023 年 11 月 13 日
Amazon RDS が、データベースプレビュー環境で MySQL 8.1 をサポート	MySQL 8.1 は、米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境の MySQL バージョン 8.1」を参照してください。	2023 年 11 月 10 日
RDS が MySQL 8.0.35 および 5.7.44 をサポート	MySQL バージョン 8.0.35 および 5.7.44 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 11 月 9 日
RDS Proxy がマルチ AZ DB クラスターをサポート	RDS Proxy がマルチ AZ DB クラスターへの接続をサポートするようになりました。詳細については、「Amazon RDS Proxy エンドポイントの操作」を参照してください。	2022 年 11 月 9 日

RDS Custom for Oracle が AWS GovCloud (US) Regions で利用可能に	Amazon RDS が AWS GovCloud (US) Regions で利用可能になりました。詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2023 年 11 月 9 日
Amazon RDS Optimized Writes で db.m5 DB インスタンスクラスをサポート	Amazon RDS Optimized Writes で db.m5 DB インスタンスクラスがサポートされるようになりました。詳細については、「Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上」と「Amazon RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上」を参照してください。	2022 年 11 月 9 日

Amazon RDS for Oracle が、CDB アーキテクチャのマルチテナント設定をサポート	RDS for Oracle のマルチテナント機能により、RDS は Oracle データベースにフルマネージド型の Oracle マルチテナントアーキテクチャとエクスペリエンスを提供します。RDS API を使用して、テナントデータベースと呼ばれる複数の PDB を CDB 内に作成できます。RDS は、従来のシングルテナント設定に代わるものとして CDB アーキテクチャのマルチテナント設定を提供しています。詳細については、「CDB アーキテクチャのマルチテナント設定」を参照してください。	2023 年 11 月 8 日
Amazon RDS による Performance Insights カウンターメトリクスの Amazon CloudWatch へのエクスポート	Performance Insights では、事前設定済みまたはカスタムメトリックスダッシュボードを Amazon CloudWatch にエクスポートできます。CloudWatch コンソールでエクスポートしたメトリクスを表示できます。選択した Performance Insights メトリクスウィジェットをエクスポートして、CloudWatch コンソールでメトリクスデータを表示することもできます。詳細については、「Performance Insights メトリクスの CloudWatch へのエクスポート」を参照してください。	2022 年 11 月 8 日

Amazon RDS Custom for Oracle を使用すると、DB インスタンスのオペレーティングシステムをアップグレードできます	CLI コマンド `modify-db-instance` を使用して RDS Custom for Oracle DB インスタンスのデータベースまたはオペレーティングシステム (OS) をアップグレードできるようになりました。詳細については、「Amazon RDS Custom for Oracle DB インスタンスのアップグレード」を参照してください。	2023 年 11 月 7 日
RDS Proxy は PostgreSQL 用 RDS の拡張プロトコルをサポート	RDS for PostgreSQL DB インスタンスで拡張クエリプロトコルを実行できるようになりました。詳細については、「Amazon RDS Proxy の使用」を参照してください。	2023 年 11 月 6 日
RDS for PostgreSQL による RDS ブルー/グリーンデプロイのサポート	RDS for PostgreSQL DB インスタンスからブルー/グリーンデプロイを作成できるようになりました。詳細については、「データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する」を参照してください。	2023 年 10 月 26 日

AWS マネージドポリシーの更新	AmazonRDSPerformanceInsightsReadOnly および AmazonRDSPerformanceInsightsFullAccess マネージドポリシーには、Sid (ステートメント ID) がポリシーステートメントの識別子として含まれます。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 10 月 23 日
RDS Custom for Oracle が欧州 (ミラノ) リージョンをサポート	詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2023 年 10 月 23 日
既存のデータベースで RDS Optimized Writes を有効にする	既存の DB インスタンスが RDS Optimized Writes 機能をサポートしていないエンジンバージョン、DB インスタンスクラス、またはファイルシステム設定で作成された場合でも、RDS Optimized Writesを有効にできるようになりました。詳細については、RDS for MySQL の場合「既存のデータベースで RDS Optimized Writes を有効にする」、RDS for MariaDB の場合「既存のデータベースで RDS Optimized Writes を有効にする」を参照してください。	2023 年 10 月 19 日

Amazon RDS は専用ログボリューム (DLV) の使用をサポートしています。	専用ログを、RDS for MariaDB、RDS for PostgreSQL、RDS for PostMySQL、RDS for PostgreSQL で使用できるようになりました。DLV は、割り当てられたストレージの容量が大きいデータベース、1 秒あたりの I/O (IOPS) 要件が高い、または遅延の影響を受けやすいワークロードがあるデータベースに最適です。詳細については、「専用ログボリューム (DLV) の使用」を参照してください。	2023 年 10 月 17 日
Amazon RDS for PostgreSQL、MySQL、および MariaDB が新しい DB インスタンスクラスをサポート	db.m6.in、db.m6idn、db.r6.in、および db.r6.idn の DB インスタンスクラスを使用する PostgreSQL、MySQL、および MariaDB を実行する Amazon RDS DB インスタンスを作成できます。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2023 年 10 月 12 日
Amazon RDS for PostgreSQL で pgactive をサポート	pgactive 拡張機能は Amazon RDS for PostgreSQL で利用できます。詳細については、「Amazon RDS for PostgreSQL で PostgreSQL 拡張機能を使用する」を参照してください。	2023 年 10 月 9 日

RDS Custom for Oracle がアジアパシフィック (ジャカルタ) リージョンで利用可能に	アジアパシフィック (ジャカルタ) リージョンで RDS Custom for Oracle DB インスタンスを作成できます。詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2023 年 10 月 5 日
RDS Custom for SQL Server で新しいサーバーレベルの照合順序をサポート	RDS Custom for SQL Server は、SQL_Latin、日本語、ドイツ語、アラビア語のロケールで、従来のエンコーディングと UTF-8 エンコーディングのどちらでも、さまざまなサーバー照合順序をサポートするようになりました。詳細については、「RDS Custom for SQL Server DB インスタンスの照合順序と文字のサポート」を参照してください。	2023 年 9 月 26 日
AWS マネージドポリシーアクセス許可の更新	`AWSServiceRoleForRDSCustom` のサービスにリンクされたロールの `AmazonRDSCustomServiceRolePolicy` には、EventBridge マネージドルールの作成、変更、削除を RDS Custom に許可する新しいアクセス権限があります。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 9 月 20 日

Amazon RDS が Performance Insights カウンターメトリクスを Amazon CloudWatch に発行	CloudWatch コンソールの DB_PERF_INSIGHTS メトリクス数学関数を使用すると、Amazon RDS にクエリを実行して Performance Insights カウンターメトリクスを取得できます。詳細については、「Amazon RDS をモニタリングするための CloudWatch アラームの作成」を参照してください。	2023 年 9 月 20 日
Performance Insights で SQL Server のダイジェストレベルの統計をサポート	Performance Insights を使用すると、Amazon RDS for SQL Server のステートメントレベルとダイジェストレベルの両方で SQL 統計を表示できます。詳細については、「SQL Server でのクエリの実行を分析」を参照してください。	2023 年 9 月 18 日
Amazon RDS for PostgreSQL、MySQL、および MariaDB は、db.m6.id と db.r6.id DB インスタンスクラスタイプをサポート	メモリが最適化された db.m6.id および db.r6.id DB インスタンスクラスを使用する PostgreSQL、MySQL、および MariaDB を実行する Amazon RDS DB インスタンスを作成できるようになりました。これらのタイプは、NVMe ベースのローカルの SSD ストレージを提供します。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2023 年 9 月 11 日

メジャーバージョンのアップグレードは、RDS for PostgreSQL マルチ AZ DB クラスターをサポート	RDS for PostgreSQL マルチ AZ DB クラスターのメジャーバージョンアップグレードを実行できるようになりました。詳細については、「RDS for PostgreSQL マルチ AZ DB クラスターのアップグレード」を参照してください。	2023 年 9 月 7 日
Amazon RDS は MariaDB 10.11.5、10.6.15、10.5.22、および 10.4.31 をサポート	MariaDB バージョン 10.11.5、10.6.15、10.5.22、および 10.4.31 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2023 年 9 月 7 日
Amazon RDS 延長サポート	Amazon RDS は、RDS の標準サポート終了日を過ぎても、DB インスタンスで RDS for MySQL および RDS for PostgreSQL のメジャーエンジンバージョンを引き続き実行できるようになることを発表しました。詳細については、「Amazon RDS 延長サポートの使用」を参照してください。	2023 年 9 月 1 日

RDS Custom は、RDS Custom for SQL Server DB インスタンスの起動と停止をサポート	RDS Custom は、RDS Custom for SQL Server DB インスタンスの起動と停止をサポートするようになりました。詳細については、「RDS Custom for SQL Server DB インスタンスの起動と停止」を参照してください。	2023 年 8 月 31 日
Amazon RDS Optimized Writes で db.r5 DB インスタンスクラスをサポート	Amazon RDS Optimized Writes で db.r5 DB インスタンスクラスがサポートされるようになりました。詳細については、「Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上」と「Amazon RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上」を参照してください。	2023 年 8 月 31 日
Amazon RDS for Oracle は、CDB のタイムゾーンファイルの自動アップグレードをサポート	`TIMEZONE_FILE_AUTOUPGRADE` オプションを使用すると、現在のタイムゾーンファイルを RDS for Oracle コンテナデータベース (CDB) の最新バージョンにアップグレードできます。詳細については、「Oracle のタイムゾーンファイルの自動アップグレード」を参照してください。	2023 年 8 月 29 日

Amazon RDS Optimized Writes が、db.m6g および db.m6i DB インスタンスクラスをサポート	Amazon RDS Optimized Writes で、db.m6g および db.m6i DB インスタンスクラスがサポートされるようになりました。詳細については、「Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上」と「Amazon RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上」を参照してください。	2023 年 8 月 28 日
Amazon RDS が MariaDB 10.11 をサポート	MariaDB バージョン 10.6 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2023 年 8 月 21 日
AWS マネージドポリシーアクセス許可の更新	`AWSServiceRoleForRDSCustom` サービスリンクロールの `AmazonRDSCustomServiceRolePolicy` には、RDS Custom がネットワークインターフェイスを作成できる新しい権限が追加されました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 8 月 18 日

AWS マネージドポリシーアクセス許可の更新	AmazonRDSFullAccess 管理ポリシーに、一定期間のパフォーマンス分析レポートの生成、表示、削除を許可する新しいアクセス許可が追加されました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 8 月 17 日
AWS マネージドポリシーアクセス許可の更新	AmazonRDSPerformanceInsightsReadOnly 管理ポリシーへの新しいアクセス許可の追加と、新しい管理ポリシー AmazonRDSPerformanceInsightsFullAccess の追加により、一定期間の DB 負荷分析レポートを生成できるようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 8 月 16 日
Amazon RDS が一定期間のパフォーマンス分析をサポート	Performance Insights では、特定の期間のパフォーマンス分析レポートを作成し表示できます。レポートには、特定されたインサイトと、パフォーマンス問題を解決するための推奨事項が記載されています。詳細については、「一定期間の DB 負荷の分析」を参照してください。	2023 年 8 月 16 日

Amazon RDS Custom for Oracle が db.r5b および db.x2iedn DB インスタンスクラスをサポート	RDS Custom for Oracle DB インスタンスで、db.r5b および db.x2iedn インスタンスクラスを使用できるようになりました。詳細については、「RDS Custom for Oracle の DB インスタンスクラスサポート」を参照してください。	2023 年 8 月 16 日
Amazon RDS Custom for Oracle が db.m6i、db.r6i、および db.t3 DB インスタンスクラスをサポート	RDS Custom for Oracle DB インスタンスで、db.m6i, db.r6i, および db.t3 インスタンスクラスを使用できるようになりました。詳細については、「RDS Custom for Oracle の DB インスタンスクラスサポート」を参照してください。	2023 年 8 月 15 日
Amazon RDS for PostgreSQL が、PostgreSQL バージョン 16 ベータ 3 をデータベースプレビュー環境でサポート	PostgreSQL バージョン 16 Beta 3 が米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で使用可能になりました。詳細については、「データベースプレビュー環境での作業」を参照してください。	2023 年 8 月 11 日
Amazon RDS が MySQL 8.0.34 および 5.7.43 をサポート	MySQL バージョン 8.0.34 および 5.7.43 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2023 年 8 月 9 日

RDS for SQL Server がスタンバイレプリカの OS メトリクスビューをサポート	RDS for SQL Server のスタンバイレプリカの OS メトリクスを表示できるようになりました。詳細については、「RDS コンソールでの OS メトリクスの表示」を参照してください。	2023 年 8 月 3 日
RDS for Oracle が Oracle Data Guard for CDB をサポート	RDS for Oracle は、Oracle Database 19c および 21c コンテナデータベース (CDB) について、Data Guard リードレプリカをサポートしています。CDB 以外と同様に、CDB でも既存の RDS API を使用して、リードレプリカを作成、管理、および昇格できます。詳細については、「マルチテナントリードレプリカ」を参照してください。	2023 年 8 月 1 日
Amazon EKS がイスラエル (テルアビブ) リージョンで使用可能に	Amazon RDS がイスラエル (テルアビブ) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2023 年 8 月 1 日
Amazon RDS が Oracle APEX バージョン 23.1.v1 をサポート	Oracle APEX 23.1.v1 は Oracle Database 19c 以降で使用できます。詳細については、「Oracle Application Express」を参照してください。	2023 年 7 月 26 日

Amazon RDS Custom for Oracle が、デフォルト以外の Oracle SID をサポート	Oracle Database 19c を使用して RDS Custom for Oracle DB インスタンスを作成する場合、デフォルト以外の Oracle システム識別子 (Oracle SID) を指定できます。この値は CDB の名前でもあります。詳細については、「マルチテナントアーキテクチャの考慮事項」を参照してください。	2023 年 7 月 21 日
RDS for SQL Server がセルフマネージド Active Directory をサポート	セルフマネージド Active Directory を使用して、RDS for SQL Server DB インスタンスを Microsoft Active Directory (AD) ドメインに直接参加させることができるようになりました。セルフマネージド AD ドメインは、オンプレミスでもクラウドでも構いません。詳細については、「セルフマネージド Active Directory の使用」を参照してください。	2023 年 7 月 7 日
マルチ AZ DB クラスターでの PostgreSQL 論理レプリケーションのサポート	マルチ AZ DB クラスターとの PostgreSQL 論理レプリケーションを使用して、データベースインスタンス全体ではなく、個々のテーブルをレプリケートおよび同期できるようになりました。詳細については、「Amazon RDS のマルチ AZ DB クラスターを使用した PostgreSQL 論理レプリケーションの設定」を参照してください。	2023 年 7 月 6 日

Amazon RDS for PostgreSQL が、PostgreSQL バージョン 16 ベータ 2 をデータベースプレビュー環境でサポート	PostgreSQL バージョン 16 Beta 2 が米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境での作業」を参照してください。	2023 年 7 月 6 日
AWS マネージドポリシーアクセス許可の更新	`AWSServiceRoleForRDSCustom` サービスリンクロールの `AmazonRDSCustomServiceRolePolicy` には、RDS Custom for Oracle がスナップショットを使用できるようにする新しい権限が追加されました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 6 月 23 日
RDS が MariaDB バージョン 10.6.14、10.5.21、10.4.30 をサポート	MariaDB バージョン 10.6.14、10.5.21、および 10.4.30 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2023 年 6 月 22 日

RDS が MySQL 8.0.33 および 5.7.42 をサポート	MySQL バージョン 8.0.33 および 5.7.42 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2023 年 6 月 15 日
RDS が MariaDB 10.6.13、10.5.20、10.4.29、10.3.39 をサポート	MariaDB バージョン 10.6.13、10.5.20、10.4.29、および 10.3.39 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2023 年 6 月 15 日
RDS for Oracle でトランスポータブル表領域をサポート	トランスポータブル表領域を使用して、オンプレミスの Oracle データベースから RDS for Oracle DB インスタンスにデータを移行できます。この手法は追加のライセンスを必要とせず、ダウンタイムが最も少ない移行手法です。詳細については、「Oracle トランスポータブル表領域を使用した移行」を参照してください。	2023 年 6 月 15 日

Amazon RDS で RDS for MariaDB バージョン 10.6 を使用した RDS Proxy をサポート	RDS for MariaDB バージョン 10.6 データベースを使用して RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2023 年 6 月 15 日
RDS Custom for SQL Server が Bring Your Own Media (BYOM) をサポート	独自の SQL Server メディアを使用してカスタムエンジンバージョン (CEV) を作成できるようになりました。詳細については、「RDS Custom for SQL Server での Bring Your Own Media」を参照してください。	2023 年 6 月 8 日
RDS for Oracle で Oracle Database 19c 非 CDB を CDB に変換可能	お使いの DB インスタンスが April 2021 以降の RU で Oracle Database 19c を実行している場合、非 CDB を CDB (コンテナデータベース) に変換できます。アーキテクチャを変換したら、19c CDB を 21c CDB にアップグレードできます。1 つのコマンドでデータベースをアップグレードしたり、アーキテクチャを変換することはできないため、この手順は必須です。詳細については、「RDS for Oracle の 非 CDB から CDB への変換」を参照してください。	2023 年 5 月 31 日

中国リージョンで利用可能なマルチ AZ DB クラスター	マルチ AZ DB クラスターが AWS リージョン 中国 (北京) と中国 (寧夏) で利用できるようになりました。詳細については、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。	2023 年 5 月 30 日
マルチ AZ DB クラスターの Amazon RDS Optimized Reads サポート	Amazon RDS Optimized Reads は、マルチ AZ DB クラスターをサポートするようになりました。詳細については、「Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上」と「Amazon RDS Optimized Reads による RDS for PostgreSQL のクエリパフォーマンスの向上」を参照してください。	2023 年 5 月 30 日
RDS Custom for Oracle がアジアパシフィック (ジャカルタ) リージョンをサポート	詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2023 年 5 月 29 日

PostgreSQL Multi-AZ DB クラスター向けにソース RDS を使った DB インスタンスのリードレプリカを作成する	DB インスタンスのリードレプリカは、RDS for MySQL DB マルチ AZ DB クラスターをソースとして作成するできるようになりました。以前は、RDS for MySQL のみがサポートされていました。詳細については、「マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成する」を参照してください。	2023 年 5 月 24 日
Amazon RDS は、Performance Insights ダッシュボードに Performance Insights と CloudWatch メトリクスの統合を提供しています	Amazon RDS の Performance Insights ダッシュボードで、Performance Insights と CloudWatch メトリクスの統合ビューが提供されるようになりました。詳細については、「Amazon RDS コンソールでの組み合わせたメトリクスの表示」を参照してください。	2023 年 5 月 24 日
Amazon RDS Optimized Reads が、中国リージョンで利用可能	Amazon RDS Optimized Reads が AWS リージョン 中国 (北京) および中国 (寧夏) で利用可能になりました。詳細については、「Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上」と「Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上」を参照してください。	2023 年 4 月 24 日

Amazon RDS が中国リージョンで AWS Secrets Manager をサポート	Amazon RDS が、中国 (北京) および中国 (寧夏) リージョンで Secrets Manager をサポートします。詳細については、「Amazon RDS と AWS Secrets Manager によるパスワード管理」を参照してください。	2023 年 4 月 20 日
RDS Custom for Oracle が、新しい CEV について AMI ID の再利用をサポート	カスタムエンジンバージョン (CEV) を作成すると、RDS Custom for Oracle が、使用可能な最新の Amazon マシンイメージ (AMI) になります。以前の CEV で使用された AMI ID を指定できるようになりました。詳細については、「CEV の作成」を参照してください。	2023 年 4 月 19 日
Amazon RDS が、トピックサブスクライバーへのタグ付きイベントの発行をサポート	Amazon Simple Notification Service (Amazon SNS) または Amazon EventBridge に送信された Amazon RDS イベント通知に、メッセージ本文にイベントタグが含まれるようになりました。これらのタグは、サービスイベントの影響を受けたリソースデータを提供します。詳細については、「Amazon RDS event notification tags and attributes」(Amazon Redshift イベント通知タグと属性) を参照してください。	2023 年 4 月 17 日

マルチ AZ DB クラスターのリザーブドインスタンスを購入する	マルチ AZ DB クラスターのリザーブド DB インスタンスを購入できるようになりました。詳細については、「マルチ AZ DB クラスターのリザーブド DB インスタンス」を参照してください。	2023 年 4 月 12 日
Amazon RDS が db.m7g および db.r7g インスタンスクラスをサポート	RDS for MySQL、RDS for MariaDB、および RDS for PostgreSQL DB インスタンスで、db.m7g および db.r7g インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2023 年 4 月 12 日
Amazon RDS Custom のサービスにリンクされたロールのアクセス許可の更新	`AmazonRDSCustomServiceRolePolicy` は、RDS Custom for SQL Server が Amazon SQS を使用してスナップショットを作成できるように、追加のアクセス許可を付与するようになりました。詳細については、「AWS 管理ポリシーに関する更新」を参照してください。	2023 年 4 月 6 日

リードレプリカを使用して RDS for MySQL マルチ AZ DB クラスターに移行する	リードレプリカを使用して、RDS for MySQL シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイを RDS for MySQL マルチ AZ DB クラスターデプロイにダウンタイムを短縮して移行できるようになりました。詳細については、「リードレプリカを使用してマルチ AZ DB クラスターに移行する」を参照してください。	2023 年 4 月 6 日
マルチ AZ DB クラスターから DB インスタンスリードレプリカを作成する	ソースクラスターのコンピューティング能力を超えてスケーリングするために、マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成できるようになりました。詳細については、「マルチ AZ DB クラスターから DB インスタンスのリードレプリカを作成する」を参照してください。	2023 年 4 月 6 日
Amazon RDS Custom for SQL Server がマルチ AZ をサポート	RDS Custom for SQL Server でマルチ AZ 配置を作成できます。詳細については、「Managing a Multi-AZ deployment for RDS Custom for SQL Server」(RDS Custom for SQL Server のマルチ AZ 配置の管理) を参照してください。	2023 年 4 月 6 日

AWS 管理ポリシーアクセス許可の更新	AmazonRDSFullAccess および AmazonRDSReadOnlyAccess ポリシーは、RDS コンソールで Amazon DevOps Guru の検出結果を表示できる追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 3 月 30 日
Amazon RDS が Oracle APEX バージョン 22.2.v1 をサポート	Oracle APEX 22.2.v1 は、サポートされているすべてのバージョンの Oracle データベースで使用できます。詳細については、「Oracle Application Express」を参照してください。	2023 年 3 月 30 日

Amazon DevOps Guru が RDS for PostgreSQL で利用可能	RDS for PostgreSQL は、Amazon DevOps Guru によって検出された最近の異常についてアラートを送信します。コンソールのデータベース詳細ページに、現在の異常と過去 24 時間に発生した異常の両方が通知されます。DevOps Guru は、RDS for PostgreSQL データベースの問題が発生すると予測される前に、問題の対処に役立つ推奨事項と共にプロアクティブインサイトを発行します。詳細については、「DevOps Guru for RDS の動作」を参照してください。	2023 年 3 月 30 日
RDS Custom が Amazon EBS gp3 ストレージボリュームをサポート	RDS Custom for Oracle および RDS Custom for SQL Server の両方が io1、gp2、および gp3 SSD ベースの EBS ボリュームをサポートします。詳細については、「RDS Custom for Oracle の一般的な要件」および「RDS Custom for SQL Server の一般的な要件」を参照してください。	2023 年 3 月 29 日

AWS 管理ポリシーアクセス許可の更新	AmazonRDSFullAccess および AmazonRDSReadOnlyAccess ポリシーは、Amazon CloudWatch に追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2023 年 3 月 16 日
RDS Proxy が中国リージョンで使用可能	RDS Proxy が中国 (北京) および中国 (寧夏) リージョンで利用可能になりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2023 年 3 月 15 日
RDS Proxy がアジアパシフィック (ジャカルタ) リージョンで利用可能に	RDS Proxy がアジアパシフィック (ジャカルタ) リージョンで利用可能になりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2023 年 3 月 8 日
Amazon RDS Optimized Writes による RDS for MariaDB の書き込みトランザクションのパフォーマンスの向上	Amazon RDS Optimized Writes によって、RDS for MariaDB DB インスタンスの書き込みトランザクションのパフォーマンスを向上させることができます。詳細については、「Amazon RDS Optimized Writes for MariaDB による書き込みパフォーマンスの向上」を参照してください。	2023 年 3 月 7 日

Amazon RDS for PostgreSQL バージョン 15.2	Amazon RDS for PostgreSQL 15.2 の新機能には、条件付き SQL クエリ用の SQL 標準の「MERGE」コマンド、インメモリとディスクベースの両方のソートのパフォーマンスの向上、論理レプリケーションのための 2 フェーズコミットと行/列フィルタリングのサポートなどがあります。	2023 年 2 月 27 日
RDS Custom for Oracle が、カナダ (中部) リージョンと南米 (サンパウロ) リージョンで使用可能	サポートされているすべての AWS リージョン を示した表については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2023 年 2 月 22 日
Amazon RDS が RDS for MariaDB および RDS for MySQL のクロスリージョン自動バックアップをサポート	RDS for MariaDB と RDS for MySQL DB インスタンスについて、AWS リージョン 間で DB スナップショットおよびトランザクションログをレプリケートできるようになりました。詳細については、「自動バックアップを別の AWS リージョン にレプリケートする」を参照してください。	2023 年 2 月 22 日

Amazon RDS for Oracle が自動マイナーバージョンアップグレードの事前通知をサポート	RDS は、RDS for Oracle エンジンの新しいマイナーバージョンが利用可能になる日付を事前に通知します。RDS は、利用可能日に RDS for Oracle DB インスタンスの自動マイナーバージョンアップグレードのスケジューリングを開始します。詳細については、「マイナーバージョンの自動アップグレードをスケジュールする前に」を参照してください。	2023 年 2 月 21 日

Amazon RDS for SQL Server がデータベースアクティビティストリームをサポート	データベースアクティビティストリームを使用して SQL Server DB インスタンスをモニタリングできるようになりました。SQL Server データベースインスタンスにはサーバー監査があり、Amazon RDS によって管理されます。サーバー監査仕様にサーバーイベントを記録するポリシーを定義できます。データベース監査仕様を作成し、データベースイベントを記録するポリシーを定義できます。アクティビティのストリーミングは、収集後、Amazon Kinesis に送信されます。Kinesis から、アクティビティストリームをモニタリングして詳細な分析を行うことができます。詳細については、「データベースアクティビティストリームによる Amazon RDS のモニタリング」を参照してください。	2023 年 2 月 15 日
RDS が MySQL 8.0.32 および 5.7.41 をサポート	MySQL バージョン 8.0.32 および 5.7.41 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2023 年 2 月 7 日

Amazon RDS for Oracle は、新しい SSL 暗号スイートをサポートしています	Oracle Database 19c または 21c を実行する場合、RDS for Oracle の SSL オプションで 6 つの新しい暗号スイートを指定できます。これらのスイートは FIPS をサポートしており、FedRAMP に準拠しています。詳細については、「Oracle セキュアソケットレイヤー」を参照してください。	2023 年 2 月 3 日
Amazon RDS for Oracle が Oracle Enterprise Manager について新しい暗号スイートをサポート	OEM オプションとして、4 つの新しい FedRAMP 準拠の暗号スイートを使用できます。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。	2023 年 2 月 3 日
RDS for Oracle が、アジアパシフィック (ハイデラバード)、欧州 (スペイン)、中東 (アラブ首長国連邦) リージョンで、データベースアクティビティストリームをサポート	詳細については、「Amazon RDS のデータベースアクティビティストリームでサポートされているリージョンと DB エンジン」を参照してください。	2023 年 1 月 27 日

リードレプリカを使用して RDS for PostgreSQL マルチ AZ DB クラスターに移行する	リードレプリカを使用することで、RDS for PostgreSQL シングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイを RDS for PostgreSQL マルチ AZ DB クラスターデプロイにダウンタイムを短縮して移行できます。詳細については、「リードレプリカを使用してマルチ AZ DB クラスターに移行する」を参照してください。	2023 年 1 月 23 日
Amazon RDS がアジアパシフィック (メルボルン) リージョンで利用可能に	Amazon RDS がアジアパシフィック (メルボルン) リージョンで使用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2023 年 1 月 23 日
RDS for MariaDB が SSL/TLS 接続の適用をサポート	RDS for MariaDB は、`require_secure_transport` パラメータを `ON` に設定することで、SSL/TLS 接続の適用をサポートするようになりました。詳細については、「MariaDB DB インスタンスへのすべての接続に SSL/TLS を要求する」を参照してください。	2023 年 1 月 19 日

Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上	Amazon RDS Optimized Reads によって、MariaDB DB インスタンスの高速クエリ処理を実現できます。詳細については、「Amazon RDS Optimized Reads による RDS for MariaDB のクエリパフォーマンスの向上」を参照してください。	2023 年 1 月 11 日
マルチ AZ DB クラスターのスナップショットを DB インスタンスに復元する	マルチ AZ DB クラスターのスナップショットをシングル AZ デプロイまたはマルチ AZ DB インスタンスデプロイに復元できるようになりました。詳細については、「マルチ AZ DB クラスターのスナップショットから DB インスタンスに復元する」を参照してください。	2023 年 1 月 10 日
DB インスタンスの作成時に認証局 (CA) を指定	DB インスタンスの作成時に、DB インスタンスのサーバー証明書に使用する CA を指定できるようになりました。詳細については「Certificate authorities」(認証局) を参照してください。	2023 年 1 月 5 日

RDS Custom for SQL Server でカスタムエンジンバージョンをサポート	RDS Custom for SQL Server のカスタムエンジンバージョン (CEV) は Microsoft SQL Server がプリインストールされた Amazon マシンイメージ (AMI) です。ベースイメージとして使用する Amazon EC2 Windows AMI を選択し、オペレーティングシステム (OS) に他のソフトウェアをインストールできます。OS と SQL Server の設定は、企業のニーズに合わせてカスタマイズできます。詳細については、「RDS Custom for SQL Server のカスタムエンジンバージョンの使用」を参照してください。	2022 年 12 月 28 日
Amazon RDS ブルー/グリーンデプロイが追加の AWS リージョン で利用可能に	ブルー/グリーンデプロイ機能が、中国 (北京) および中国 (寧夏) リージョンで利用可能になりました。詳細については、「Using Amazon RDS Blue/Green Deployments for database updates」(データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する) を参照してください。	2022 年 12 月 22 日

IAM サービスリンクロール許可に対する更新	AmazonRDSServiceRolePolicy ポリシーによって、AWS Secrets Manager に追加のアクセス許可を付与するようになりました。詳細については、「Amazon RDS の AWS マネージドポリシーに関する更新」を参照してください。	2022 年 12 月 22 日
Amazon RDS でマルチ AZ DB クラスターの名前変更をサポート	マルチ AZ DB クラスターの名前を変更できるようになりました。詳細については、「マルチ AZ DB クラスターの名前の変更」を参照してください。	2022 年 12 月 22 日
パスワード管理用に Amazon RDS と AWS Secrets Manager が統合	Amazon RDS では、DB インスタンスまたはマルチ AZ DB クラスターのマスターユーザーパスワードを Secrets Manager 管理するように指定できます。詳細については、「Amazon RDS と AWS Secrets Manager によるパスワード管理」を参照してください。	2022 年 12 月 22 日
Amazon RDS Optimized Writes で db.r6g および db.r6gd DB インスタンスクラスをサポート	Amazon RDS Optimized Writes で、db.r6g および db.r6gd DB インスタンスクラスがサポートされるようになりました。詳細については、「Amazon RDS Optimized Writes による書き込みパフォーマンスの向上」を参照してください。	2022 年 12 月 22 日

Amazon RDS Custom for Oracle が新規の AWS リージョン をサポート	アジアパシフィック (ソウル)、アジアパシフィック (大阪) リージョンで RDS Custom for Oracle DB を作成できます。詳細については、「RDS Custom for Oracle でサポートされているリージョンと DB エンジン」を参照してください。	2022 年 12 月 21 日
Amazon RDS on AWS Outposts でリードレプリカをサポート	RDS on Outposts MySQL または PostgreSQL DB インスタンスからリードレプリカを作成できます。詳細については、「AWS Outposts での Amazon RDS のリードレプリカの作成」を参照してください。	2022 年 12 月 19 日
RDS Custom for Oracle で DB インスタンスクラスの変更をサポート	RDS Custom for Oracle DB インスタンスのインスタンスクラスを変更できるようになりました。詳細については、「Modifying your RDS Custom for Oracle DB instance」(RDS Custom for Oracle DB インスタンスの変更) を参照してください。	2022 年 12 月 16 日

RDS for MySQL および RDS for PostgreSQL で db.x2iedn DB インスタンスクラスをサポート	RDS for MySQL および RDS for PostgreSQL DB インスタンスで、db.x2iedn DB インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2022 年 12 月 14 日
Amazon RDS Optimized Writes で db.x2iedn DB インスタンスクラスをサポート	Amazon RDS Optimized Writes で、db.x2iedn DB インスタンスクラスがサポートされるようになりました。詳細については、「Amazon RDS Optimized Writes による書き込みパフォーマンスの向上」を参照してください。	2022 年 12 月 14 日
Amazon RDS で DB スナップショットのコピー時に DB オプショングループのコピーをサポート	RDS for Oracle データベースのスナップショットのコピーリクエストの一部として、オプショングループを AWS アカウント 間でコピーできるようになりました。詳細については、「オプショングループの考慮事項」をご参照ください。	2022 年 12 月 13 日
Amazon RDS で RDS for PostgreSQL バージョン 14 を使用した RDS Proxy をサポート	RDS for PostgreSQL バージョン 14 データベースを使用して RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2022 年 12 月 13 日

Amazon RDS for Oracle で db.x2idn、db.x2iedn、db.x2iezn インスタンスクラスをサポート	Amazon RDS for Oracle DB インスタンスで、db.x2idn、db.x2iedn、db.x2iezn インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」および「サポートされている RDS for Oracle インスタンスクラス」を参照してください。	2022 年 12 月 12 日
Trusted Language Extensions for PostgreSQL で RDS for PostgreSQL DB インスタンスをサポート	Trusted Language Extensions for PostgreSQL は、高性能の PostgreSQL 拡張機能を構築して、RDS for PostgreSQL DB インスタンスで安全に実行できるようにするオープンソースの開発キットです。詳細については「Working with Trusted Language Extensions for PostgreSQL」(Trusted Language Extensions for PostgreSQL の使用) を参照してください。	2022 年 11 月 30 日

データベース更新のための Amazon RDS ブルー/グリーンデプロイの使用	ステージング環境で DB インスタンスを変更し、本稼働環境の DB インスタンスに影響を与えずに変更をテストできます。準備ができたら、ダウンタイムを最小限に抑えながら、ステージング環境を新しい本稼働環境に昇格できます。詳細については、「Using Amazon RDS Blue/Green Deployments for database updates」(データベース更新のために Amazon RDS ブルー/グリーンデプロイを使用する) を参照してください。	2022 年 11 月 27 日
Amazon RDS Optimized Writes による RDS for MySQL の書き込みトランザクションのパフォーマンスの向上	Amazon RDS Optimized Writes によって、RDS for MySQL DB インスタンスの書き込みトランザクションのパフォーマンスを向上させることができます。詳細については、「Amazon RDS Optimized Writes for MySQL による書き込みパフォーマンスの向上」を参照してください。	2022 年 11 月 27 日

Amazon RDS Optimized Reads による RDS for MySQL のクエリパフォーマンスの向上	Amazon RDS Optimized Reads によって、RDS for MySQL DB インスタンスの高速クエリ処理を実現できます。詳細については、「Amazon RDS Optimized Reads によるクエリパフォーマンスの向上」を参照してください。	2022 年 11 月 27 日
Amazon RDS がアジアパシフィック (ハイデラバード) リージョンで利用可能に	Amazon RDS がアジアパシフィック (ハイデラバード) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2022 年 11 月 22 日
RDS で MariaDB 10.6.11、10.5.18、10.4.27、10.3.37 をサポート	MariaDB バージョン 10.6.11、10.5.18、10.4.27、10.3.37 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2022 年 11 月 18 日

RDS Custom for Oracle がカスタムエンジンバージョン (CEV) でデフォルト以外のインストールパラメータの設定をサポート	CEV を作成すると、Oracle ベース、Oracle ホーム、UNIX ユーザー名と ID、UNIX グループ名と ID にデフォルト以外の値を設定できます。これにより、RDS Custom for Oracle DB インスタンスへのデータベースのインストールをより詳細に制御できます。詳細については、「CEV マニフェストの準備」を参照してください。	2022 年 11 月 18 日
Amazon RDS で Oracle APEX バージョン 22.1.v1 をサポート	Oracle APEX 22.1.v1 は、サポートされているすべてのバージョンの Oracle データベースで使用できます。詳細については、「Oracle Application Express」を参照してください。	2022 年 11 月 18 日
RDS for SQL Server でクロスリージョンリードレプリカをサポート	クロスリージョンリードレプリカを作成することで、ディザスタリカバリ機能を強化し、アプリケーションの読み取りレイテンシーを軽減し、プライマリ DB インスタンスからの読み取りワークロードをオフロードできるようになりました。詳細については、「別の AWS リージョン でのリードレプリカの作成」を参照してください。	2022 年 11 月 16 日

Amazon RDS が 欧州 (スペイン) リージョンで利用可能に	Amazon RDS が 欧州 (スペイン) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2022 年 11 月 16 日
RDS for SQL Server で Oracle データベースのリンクサーバーをサポート	リンクサーバーを作成することで、外部の Oracle データベースにアクセスしてデータを読み取り、SQL コマンドを実行できるようになりました。詳細については、「Linked Servers with Oracle OLEDB with RDS for SQL Server」(RDS for SQL Server と Oracle OLEDB を使用したリンクサーバー) を参照してください。	2022 年 11 月 15 日
RDS Custom for Oracle で Oracle Multitenant をサポート	RDS Custom for Oracle DB インスタンスをコンテナデータベース (CDB) として作成できます。作成後、CDB には CDB ルート、PDB シード、1 つの PDB が含まれます。Oracle SQL を使用して、手動で PDB を追加作成できます。詳細については、「Amazon RDS Custom for Oracle アーキテクチャの概要」を参照してください。	2022 年 11 月 15 日

Amazon RDS for Oracle で Amazon EFS 統合をサポート	オプショングループに EFS_INTEGRATION オプションを追加する場合は、RDS for Oracle DB インスタンスと Amazon EFS ファイルシステムの間でファイルを転送できます。詳細については、「Amazon EFS」を参照してください。	2022 年 11 月 15 日
RDS で MySQL 8.0.31 および 5.7.40 をサポート	MySQL バージョン 8.0.31 および 5.7.40 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 11 月 10 日
Amazon RDS が 欧州 (チューリッヒ) リージョンで利用可能に	Amazon RDS が 欧州 (チューリッヒ) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2022 年 11 月 9 日
RDS for SQL Server でトランザクションログのバックアップにアクセス可能に	データベーストランザクションログのバックアップを表示して、Amazon S3 バケットにコピーできるようになりました。詳細については、「Access to transaction log backups」(トランザクションログのバックアップへのアクセス) を参照してください。	2022 年 11 月 7 日

マルチ AZ DB クラスターが追加の AWS リージョン で利用可能に	マルチ AZ DB クラスターが追加の AWS リージョン で利用可能になりました。詳細については、「Amazon RDS のマルチ AZ DB クラスターでサポートされているリージョンと DB エンジン」を参照してください。	2022 年 11 月 4 日
Amazon RDS で gp3 ストレージをサポート	Amazon EBS 汎用 SSD (gp3) ストレージボリュームを使用する Amazon RDS DB インスタンスを作成できるようになりました。これにより、ストレージ容量に関係なくストレージパフォーマンスをカスタマイズできます。詳細については、「汎用 SSD ストレージ」を参照してください。	2022 年 11 月 4 日
Amazon RDS でオペレーティングシステムアップデートの新しいイベントをサポート	Amazon RDS で、セキュリティパッチのイベントカテゴリで新しい DB インスタンスイベント RDS-EVENT-0230 がサポートされるようになりました。この新しいイベントでは、DB インスタンスでオペレーティングシステムの更新が可能になったときにアラートが発行されます。詳細については、「Amazon RDS イベントのモニタリング」および「オペレーティングシステムアップデートの操作」を参照してください。	2022 年 10 月 28 日

Amazon RDS for Oracle で、事前設定された r5b メモリ最適化インスタンスクラスをサポート	db.r5b Oracle DB インスタンスクラスは、vCPU ごとに追加のメモリ、ストレージ、および I/O を必要とするワークロード向けに最適化されています。例えば、db.r5b.4xlarge.tpc2.mem2x ではマルチスレッドが有効になっており、db.r5b.4xlarge の 2 倍のメモリが利用できます。詳細については、「RDS for Oracle インスタンスクラス」を参照してください。	2022 年 10 月 27 日
Amazon RDS で、RDS for MariaDB、MySQL、PostgreSQL DB インスタンスに対して 15 個のリードレプリカをサポート	RDS for MariaDB、MySQL、PostgreSQL DB インスタンスに対して、最大 15 個のリードレプリカを作成できるようになりました。リードレプリカの詳細については、「リードレプリカの使用」を参照してください。	2022 年 10 月 20 日
Amazon RDS for PostgreSQL で PostgreSQL バージョン 15 RC 3 のデータベースプレビュー環境でのサポートを開始	PostgreSQL バージョン 15 Beta 3 が米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境での作業」を参照してください。	2022 年 10 月 18 日

Amazon RDS は RDS データベースと EC2 インスタンスとの接続を自動的にセットアップすることをサポート	AWS マネジメントコンソールを使用して、既存の RDS DB インスタンスまたはマルチ AZ DB クラスターと EC2 インスタンス間の接続を設定できます。詳細については、「EC2 インスタンスと RDS データベースへの自動接続」を参照してください。	2022 年 10 月 14 日
PostgreSQL 用 AWS JDBC ドライバーが一般利用可能に	PostgreSQL 用 AWS JDBC ドライバーは、RDS for PostgreSQL 用に設計されたクライアントドライバーです。PostgreSQL 用 AWS JDBC ドライバーの一般利用が可能になりました。詳細については、「PostgreSQL 用 AWS JDBC ドライバーを使用した接続」を参照してください。	2022 年 10 月 6 日
Amazon RDS for Oracle が Oracle APEX バージョン 21.2.v1 をサポート	Oracle APEX 21.2 にはパッチ 33420059 が含まれています。詳細については、「APEX バージョンの要件」を参照してください。	2022 年 10 月 3 日
RDS で MySQL 5.7.39 をサポート	MySQL バージョン 5.7.39 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 9 月 29 日

RDS が MariaDB 10.6.10 をサポート	MariaDB バージョン 10.6.10 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2022 年 9 月 29 日
RDS Proxy は、RDS for SQL Server をサポートしています	Microsoft SQL Server バージョン 2014 以降を実行する RDS DB インスタンスの RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2022 年 9 月 19 日
RDS が MariaDB バージョン 10.5.17、10.4.26、および 10.3.36 をサポート	MariaDB バージョン 10.5.17、10.4.26、および 10.3.36 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2022 年 9 月 15 日

Amazon RDS for Oracle が、一時データのローカルインスタンスストレージをサポート	これで、インスタンスストアを使用するように構成された一時テーブルスペースとデータベースのスマートフラッシュキャッシュ (フラッシュキャッシュ) を使用して、Amazon EC2 db.r5d および db.m5d インスタンスタイプで Amazon RDS for Oracle を起動できます。一時データをローカルに保存することで、Amazon EBS に基づくスタンダードストレージベースのサービスに比べて読み取りと書き込みのレイテンシーを低く押さえることができます。詳細については、「Oracle の一時データをインスタンスストアに保存する」を参照してください。	2022 年 9 月 14 日
パフォーマンスインサイトは上位 25 行の SQL クエリを表示します	[Top SQL] (上位の SQL) タブは DB の負荷に最も影響している 25 行の SQL クエリを表示します。詳細については、「上位の SQL タブの概要」を参照してください。	2022 年 9 月 13 日
RDS で MySQL 8.0.30 をサポート	MySQL バージョン 8.0.30 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 9 月 9 日

Amazon RDS が中東 (アラブ首長国連邦) リージョンで利用可能に	Amazon RDS が、中東 (UAE) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2022 年 8 月 30 日
Amazon RDS for SQL Server が SSRS E メールサブスクリプションをサポート	SQL Server Reporting Services (SSRS) の E メール拡張機能を使用して、レポートをユーザーに送信したり、レポートサーバー上のレポートを購読できるようになりました。詳細については、「RDS for SQL Server での SQL Server Reporting Services のサポート」を参照してください。	2022 年 8 月 26 日
RDS for Oracle でリードレプリカのバックアップをサポート	RDS for Oracle レプリカの手動スナップショットを作成したり、自動バックアップを有効にすることはできません。詳細については、「RDS for Oracle レプリカのバックアップの使用」を参照してください。	2022 年 8 月 23 日

RDS for Oracle が Oracle Data Guard スイッチオーバーをサポート	スイッチオーバーとは、プライマリデータベースと、マウントされた、または開いている Oracle レプリカの間で役割を逆転させることです。スイッチオーバー中、元のプライマリデータベースはスタンバイロールに移行し、元のスタンバイデータベースはプライマリロールに移行します。詳細については、「Oracle Data Guard スイッチオーバーの実行」を参照してください。	2022 年 8 月 23 日
Amazon RDS は EC2 インスタンスとの接続を自動的にセットアップすることをサポート	DB インスタンスとマルチ AZ DB クラスターを作成する場合は、AWS マネジメントコンソール を使用して Amazon Elastic Compute Cloud インスタンスと新しい DB インスタンスまたは DB クラスター間の接続をセットアップできます。詳細については、新しい DB インスタンスの「EC2 インスタンスとの自動ネットワーク接続を設定する」および新しい DB クラスターの「EC2 インスタンスとの自動ネットワーク接続を設定する」を参照してください。	2022 年 8 月 22 日

RDS Custom for Oracle が Oracle レプリカの昇格をサポート	RDS Custom for Oracle を使用している場合、`promote-read-replica` CLI コマンドを使用して、管理している Oracle レプリカを昇格させることができます。また、プライマリ DB インスタンスを削除することで、RDS Custom for Oracle はマネージド Oracle レプリカをスタンドアロンインスタンスに昇格させることができます。詳細については、「RDS Custom for Oracle の Oracle レプリカの使用」を参照してください。	2022 年 8 月 5 日
RDS for MySQL は SSL/TLS 接続の適用をサポートしています	RDS for MySQL は、`require_secure_transport` パラメータを `ON` に設定することで、SSL/TLS 接続の適用をサポートするようになりました。詳細については、「MySQL DB インスタンスへの SSL/TLS 接続の要求」を参照してください。	2022 年 8 月 1 日

Amazon RDS は、Oracle Database 12c Release 1 (12.1.0.2) のサポートを終了しました	バージョン 12.1.0.2 のサポートは、BYOL と LIの両方のライセンスモデルで廃止されています。2022 年 8 月 1 日に、RDS for Oracle は 12c Release 1 (12.1.0.2) DB インスタンスの自動アップグレードを開始し、12.1.0.2 スナップショットを Oracle Database 19c に復元しました。詳細については、AWS re:Post の「サポート終了タイムライン」を参照してください。	2022 年 8 月 1 日
RDS Proxy で RDS for MariaDB をサポート	MariaDB バージョン 10.2、10.3、10.4、10.5 を実行する RDS DB インスタンスの RDS Proxy を作成できるようになりました。MariaDB サポートは MySQL エンジンファミリーに含まれています。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2022 年 7 月 26 日
RDS for MariaDB で db.r5b DB インスタンスクラスをサポート	db.r5b DB インスタンスクラスを使用する RDS for MariaDB DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2022 年 7 月 25 日

RDS for Oracle がデータベースアクティビティストリーミングの変更をサポート	RDS for Oracle を使用している場合は、データベースアクティビティストリーミングの監査ポリシーの状態をロック (デフォルト) またはロック解除のいずれかに変更できます。アクティビティストリーミングを停止する代わりに、そのポリシーの状態をロック解除し、監査ポリシーをカスタマイズしてから、ポリシーの状態を再度ロックできます。詳細については、「データベースアクティビティストリーミングの変更」を参照してください。	2022 年 7 月 22 日
Performance Insights がアジアパシフィック (ジャカルタ) リージョンをサポート	以前は、アジアパシフィック (ジャカルタ) リージョンでは Performance Insights を使用できませんでした。この制限は解除されました。詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	2022 年 7 月 21 日

Microsoft SQL Server 2012 は Amazon RDS でのサポートを終了しました	Microsoft SQL Server 2012 は、2022 年 7 月 12 日にこのバージョンの拡張サポートを終了するという Microsoft の計画と一致して、サポートを終了しました。2022 年 6 月 1 日より、既存の Microsoft SQL Server 2012 インスタンスは、最新のマイナーバージョンの Microsoft SQL Server 2014 に自動アップグレードされます。詳細については、「Amazon RDS での Microsoft SQL Server 2012 のサポート」を参照してください。	2022 年 7 月 12 日
RDS が MariaDB 10.6.8、10.5.16、10.4.25、10.3.35、10.2.44 をサポート	MariaDB バージョン 10.6.8、10.5.16、10.4.25、10.3.35、および 10.2.44 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS でサポートされている MariaDB のバージョン」を参照してください。	2022 年 7 月 8 日

RDS Performance Insights が追加の保持期間をサポート	以前は、Performance Insights の保持期間は 7 日 (デフォルト) と 2 年 (731 日) の 2 つのみでした。これで、パフォーマンスデータを 7 日以上保持する必要がある場合は、1 ~ 24 か月を指定できます。詳細については、「Performance Insights の料金とデータ保持」を参照してください。	2022 年 7 月 1 日
RDS Custom は、アジアパシフィック (ムンバイ) および欧州 (ロンドン) リージョンをサポートします	新しい 2 つの AWS リージョン：アジアパシフィック (ムンバイ) および欧州 (ロンドン) で、RDS Custom for Oracle および RDS Custom for SQL Server DB インスタンスを作成できます。詳細については、「AWS リージョン RDS Custom for Oracle のサポート」および「AWS リージョン RDS Custom for SQL Server のサポート」を参照してください。	2022 年 6 月 21 日

RDS Custom for Oracle は Database 18c および 12c Release 2 (12.2) をサポートしています	Oracle Database 18c および 12c Release 2 (12.2) のインストールファイルを使用して、RDS Custom for Oracle の CEV を作成できるようになりました。これらの CEV を使用して、RDS Custom for Oracle DB インスタンスを作成できます。詳細については、「Amazon RDS Custom for Oracle のカスタムエンジンバージョンの使用」を参照してください。	2022 年 6 月 21 日
マルチ AZ DB クラスターでは、db.m5d および db.r5d DB インスタンスクラスがサポートされています	これで、db.m5d および db.r5d DB インスタンスクラスを使用するマルチ AZ DB クラスターを作成できるようになりました。詳細については、「マルチ AZ DB クラスター配置」および「DB インスタンスクラスタイプ」を参照してください。	2022 年 6 月 21 日
マルチ AZ DB クラスターは、追加の AWS リージョン で利用可能	欧州 (フランクフルト) および欧州 (ストックホルム) のリージョンで、マルチ AZ DB クラスターを作成できるようになりました。詳細については、「マルチ AZ DB クラスターデプロイ」を参照してください。	2022 年 6 月 21 日

RDS for Microsoft SQL Server は、透過的なデータ暗号化 (TDE) を使用するデータベースの移行をサポートします。	RDS for SQL Server は、ネイティブバックアップと復元を使用して、TDE をオンにした状態で Microsoft SQL Server データベースの移行をサポートするようになりました。詳細については、「SQL サーバーの透過的なデータ暗号化サポート」を参照してください。	2022 年 6 月 14 日
Amazon RDS が、暗号化された Amazon SNS トピックへのイベントの発行をサポート	Amazon RDS では、サーバー側の暗号化 (SSE) が有効になっている Amazon Simple Notification Service (Amazon SNS) トピックにイベントを発行できるようになりました。これにより、機密データを伝送するイベントの保護を強化できます。詳細については、「Amazon Redshift イベント通知にサブスクライブする」を参照してください。	2022 年 6 月 1 日
RDS で MySQL 5.7.38 をサポート	MySQL バージョン 5.7.38 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 5 月 31 日

RDS for PostgreSQL がリードレプリカのカスケードをサポート	PostgreSQL バージョン 14.1 以降のリリースで RDS でカスケードリードレプリカを使用できるようになりました。詳細については、「Amazon RDS での PostgreSQL リードレプリカの使用」を参照してください。	2022 年 5 月 4 日
Amazon RDS on AWS Outposts がスケールストレージと自動スケーリングオペレーションをサポート	これで、Outpost で DB インスタンスのストレージサイズを変更し、ストレージのオートスケーリングを使用できるようになりました。詳細については、「Amazon RDS 特徴の AWS Outpostsサポートに関する Amazon RDS」を参照してください。	2022 年 5 月 2 日
マルチ AZ DB クラスターは、追加の AWS リージョン で利用可能	アジアパシフィック (シンガポール) およびアジアパシフィック (シドニー) のリージョンで、マルチ AZ DB クラスターを作成できるようになりました。詳細については、「マルチ AZ DB クラスター配置」を参照してください。	2022 年 4 月 29 日

Amazon RDS はデュアルスタックモードをサポート	DB インスタンスが、デュアルスタックモードで実行できるようになりました。デュアルスタックモードでは、リソースは IPv4、IPv6、またはその両方で DB インスタンスと通信できます。詳細については、「Amazon RDS の IP アドレス指定」を参照してください。	2022 年 4 月 29 日
Amazon RDS は使用状況メトリクスを Amazon CloudWatch に公開します	Amazon CloudWatch の AWS/Usage 名前空間には、Amazon RDS サービスクォータのアカウントレベルの使用状況メトリクスが含まれています。詳細については、「Amazon RDS の Amazon CloudWatch 使用状況メトリクス」を参照してください。	2022 年 4 月 28 日
Amazon RDS for MySQL は db.m6i および db.r6i DB インスタンスクラスをサポートします	MySQL を実行する Amazon RDS DB インスタンスの db.m6i および db.r6i DB インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2022 年 4 月 28 日

Amazon RDS for PostgreSQL は db.m6i および db.r6i DB インスタンスクラスをサポートします	PostgreSQL を実行する Amazon RDS DB インスタンスの db.m6i および db.r6i DB インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2022 年 4 月 27 日
Amazon RDS for MariaDB は db.m6i および db.r6i DB インスタンスクラスをサポートします	MariaDB を実行する Amazon RDS DB インスタンスの db.m6i および db.r6i DB インスタンスクラスを使用できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2022 年 4 月 26 日
AWS Outposts 上の Amazon RDS はマルチ AZ 配置をサポート	別の Outpost にスタンバイ DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS 特徴の AWS Outpostsサポートに関する Amazon RDS」を参照してください。	2022 年 4 月 19 日
Amazon RDS for Oracle は db.m6i および db.r6i インスタンスクラスをサポートします	Oracle Database 19c を実行する場合、db.m6i および db.r6i インスタンスクラスを使用できます。db.m6i クラスは、幅広いワークロードに適した汎用インスタンスクラスです。詳細については、「RDS for Oracle インスタンスクラス」を参照してください。	2022 年 4 月 8 日

Amazon RDS for SQL Server が SQL Server Agent ジョブレプリケーションをサポート	この機能を有効にすると、プライマリホスト上で作成、変更、または削除された SQL Server Agent ジョブは、マルチ AZ 構成のセカンダリホストに自動的に同期されます。詳細については、「SQL Server エージェントの使用」を参照してください。	2022 年 4 月 7 日
Amazon RDS は PostgreSQL バージョン 13 の RDS Proxy を RDS でサポート	RDS for PostgreSQL バージョン 13 データベースを使用して RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2022 年 4 月 4 日
Amazon RDS は Oracle Database 12c を非推奨にすることを計画	Oracle Database 12c は非推奨になる予定です。Oracle Corporation は、サポート終了日以降、Oracle Database 12c のパッチを提供しなくなります。Amazon RDS は、Oracle Database 12c DB インスタンスを Oracle Database 19c に自動的にアップグレードすることを計画しています。	2022 年 3 月 22 日
Amazon RDS for PostgreSQL リリースノート	現在、Amazon RDS for PostgreSQL リリースノートには、別のガイドがあります。詳細については、Amazon RDS for PostgreSQL リリースノートを参照してください。	2022 年 3 月 22 日

Amazon RDS for Oracle リリースノート	現在、Amazon RDS for Oracle リリースノートには、別のガイドがあります。詳細については、Amazon RDS for Oracle リリースノートを参照してください。	2022 年 3 月 22 日
マルチ AZ DB クラスターは、追加の AWS リージョン で利用可能	米国東部 (オハイオ) およびアジアパシフィック (東京) のリージョンで、マルチ AZ DB クラスターを作成できるようになりました。詳細については、「マルチ AZ DB クラスター配置」を参照してください。	2022 年 3 月 15 日

Amazon RDS for PostgreSQL バージョン 14.2、13.6、12.10、11.15、10.20	RDS for PostgreSQL では、バージョン 14.2、13.6、12.10、11.15、10.20 がサポートされるようになりました。バージョン 14.2 および 13.6 では、2 つの新しい外部データラッパーのサポートが追加されました。`mysql_fdw` 拡張機能により、PostgreSQL は MySQL、MariaDB、Aurora MySQL データベースに格納されたデータを操作できます。`tds_fdw` 拡張機能により、PostgreSQL は SQL Server データベースに格納されたデータを操作できます。詳細については、「サポートされている PostgreSQL データベースのバージョン」を参照してください。	2022 年 3 月 12 日
RDS が MySQL 5.7.37 をサポート	MySQL バージョン 5.7.37 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 3 月 11 日

Amazon RDS for SQL Server が新しい DB インスタンスクラスをサポート	db.m6i および db.r6i DB インスタンスクラスを使用する Microsoft SQL Server を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。	2022 年 3 月 9 日
Amazon RDS for Oracle が Oracle Database 21c をサポート	Oracle Database 21c (21.0.0.0) を実行する Amazon RDS DB インスタンスを作成できるようになりました。これは、マルチテナント (CDB) アーキテクチャのみをサポートする最初の Oracle Database リリースです。詳細については、「Amazon RDS での Oracle Database 21c」を参照してください。	2022 年 3 月 7 日
RDS で MariaDB 10.6.7、10.5.15、10.4.24、10.3.34、および 10.2.43 をサポート	MariaDB バージョン 10.6.7、10.5.15、10.4.24、10.3.34 および 10.2.43 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2022 年 3 月 3 日

MySQL 用 AWS JDBC ドライバーが一般利用可能に	MySQL 用 AWS JDBC ドライバーは、RDS for MySQL 用に設計されたクライアントドライバーです。MySQL 用 AWS JDBC ドライバーの一般利用が可能になりました。詳細については、「Connecting with the Amazon Web Services JDBC Driver for MySQL」(MySQL 用アマゾン ウェブサービス JDBC ドライバーとの接続) を参照してください。	2022 年 3 月 2 日
マルチ AZ DB クラスターが一般利用可能に	マルチ AZ DB クラスターデプロイとは、2 つの読み取り可能なスタンバイ DB インスタンスを備えた Amazon RDS の高可用性デプロイモードです。マルチ AZ DB クラスターの一般利用が可能になりました。詳細については、「Multi-AZ DB cluster deployments」(マルチ AZ DB クラスター配置) を参照してください。	2022 年 3 月 1 日
RDS で MySQL 8.0.28 をサポート	MySQL バージョン 8.0.28 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2022 年 2 月 28 日

Amazon RDS for Oracle は、ネイティブネットワーク暗号化 (NNE) 用の新しい設定をサポートしています。	非セキュアな暗号化方式やチェックサム方式を使用したクライアント接続の可否を制御するには、NNE オプションで SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS と SQLNET.ALLOW_WEAK_CRYPTO を設定します。安全でない方式の例としては、DES、3DES、RC4、MD5 などがあります。詳細については、「NNE オプション設定」を参照してください。	2022 年 2 月 25 日
Amazon RDS for SQL Server が Microsoft SQL Server 2017 Standard Edition の Always On 可用性グループをサポート	SQL Server 2017 Standard Edition 14.00.3401.7 以降のバージョンでマルチ AZ 設定を使用して DB インスタンスを作成すると、RDS が自動的にアベイラビリティーグループを使用します。詳細については、「Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2022 年 2 月 18 日
RDS for Oracle が、アジアパシフィック (ジャカルタ) リージョンでデータベースアクティビティストリーミングをサポート	詳細については、「データベースアクティビティストリーミングのための AWS リージョン のサポート」を参照してください。	2022 年 2 月 16 日

Amazon RDS Custom for Oracle で Oracle Database 12.1 をサポート	Oracle Database 12.1 Enterprise Edition を使用する RDS Custom for Oracle のカスタムエンジンバージョンを作成できるようになりました。詳細については、「Amazon RDS Custom for Oracle のカスタムエンジンバージョンの使用」を参照してください。	2022 年 2 月 4 日
Amazon RDS for MariaDB での新しいメジャーバージョンのサポート	MariaDB バージョン 10.6 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「MariaDB 10.6 support on Amazon RDS」(Amazon RDS での MariaDB 10.6 のサポート) を参照してください。	2022 年 2 月 3 日

Performance Insights は、Oracle クエリのプランキャプチャをサポートします	Performance Insights コンソールは、トップ SQL で新しいプランディメンションをサポートします。プラン別にスライスすると、Oracle のトップクエリが使用しているプランを確認できます。クエリで複数のプランを使用する場合に、コンソールでプランを並べて比較して、最も効率的なプランを特定できます。ドリルダウンすると、プランの内で最もコストが高いステップを確認することもできます。詳細については、「Analyzing Oracle execution plans using the Performance Insights dashboard」(Performance Insights ダッシュボードを使用して Oracle 実行プランを分析する) を参照してください。	2022 年 1 月 27 日

Performance Insights で新しい API をサポート	Performance Insights では、`GetResourceMetadata` 、`ListAvailableResourceDimensions` 、および `ListAvailableResourceMetrics` の API がサポートされています。詳細については、本マニュアルならびに「Amazon RDS Performance Insights API Reference」(Amazon RDS Performance Insights API リファレンス) の、「Retrieving metrics with the Performance Insights API」(Performance Insights API を使用したメトリクスの取得) を参照してください。	2022 年 1 月 12 日
RDS Proxy でイベントがサポートに	RDS Proxy でイベントを生成し、それにサブスクライブして CloudWatch イベント内に表示したり、設定することで Amazon EventBridge への送信ができるようになりました。詳細については、「Working with RDS Proxy events」(RDS Proxy イベントの使用) を参照してください。	2022 年 1 月 11 日

Amazon RDS for SQL Server が SSAS 多次元モードをサポート	RDS for SQL Server は、表形式モードまたは多次元モードでの SQL Server Analysis Services (SSAS) の実行をサポートしています。詳細については、「Support for SQL Server Analysis Services in RDS for SQL Server」(RDS for SQL Server での SQL Server Analysis Services のサポート) を参照してください。	2022 年 1 月 7 日
RDS Proxy が追加の AWS リージョン で使用可能に	RDS Proxy が、アフリカ (ケープタウン)、アジアパシフィック (香港)、アジアパシフィック (大阪)、欧州 (ミラノ)、欧州 (パリ)、欧州 (ストックホルム)、中東 (バーレーン)、南米 (サンパウロ) の各リージョンで利用可能になりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2022 年 1 月 5 日
RDS が MySQL 8.0.27 をサポート	MySQL バージョン 8.0.27 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2021 年 12 月 21 日

Amazon RDS がアジアパシフィック (ジャカルタ) リージョンで利用可能に	Amazon RDS がアジアパシフィック (ジャカルタ) リージョンで利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2021 年 12 月 13 日
Amazon RDS が MariaDB 10.5.13、10.4.22、10.3.32、10.2.41 をサポート	MariaDB バージョン 10.5.13、10.4.22、10.3.32、10.2.41 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2021 年 12 月 8 日
Amazon RDS Custom for SQL Server	Amazon RDS Custom は、基盤となるOSとデータベース環境へのアクセスを必要とするレガシー、カスタム、およびパッケージアプリケーション向けのマネージドデータベースサービスです。Amazon RDS Custom を使用すると、Amazon RDS のオートメーションと Amazon EC2 の柔軟性が得られます。詳細については、「Amazon RDS Custom の使用」を参照してください。	2021 年 12 月 1 日

マルチ AZ DB クラスター (プレビュー)	RDS for MySQL および RDS for PostgreSQL のマルチ AZ DB クラスターを作成できるようになりました。マルチ AZ DB クラスターデプロイとは、2 つの読み取り可能なスタンバイ DB インスタンスを備えた Amazon RDS の高可用性デプロイモードです。マルチ AZ DB クラスターはプレビュー中です。詳細については、「マルチ AZ DB クラスターデプロイ (プレビュー)」を参照してください。	2021 年 11 月 23 日
Amazon RDS は PostgreSQL バージョン 12 の RDS Proxy を RDS でサポート	RDS for PostgreSQL バージョン 12 データベースを使用して RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、「Amazon RDS Proxy の使用」を参照してください。	2021 年 11 月 22 日
AWS Outposts上の Amazon RDS はローカルバックアップをサポートします	自動バックアップやマニュアルスナップショットは AWS リージョン や Outpost にローカルに保存することができます。詳細については、Amazon RDS機能の AWS Outpostsサポートに関するAmazonRDSを参照してください。	2021 年 11 月 22 日

クロスアカウントに対する Amazon RDS のサポートAWS KMS keys	DB スナップショットを Amazon S3 にエクスポートする際、他のAWSアカウントから KMS キーを使用して暗号化することができます。詳細については、「Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。	2021 年 11 月 3 日
AWSOutposts の Amazon RDS は、データベースエンジンログの CloudWatch Logs への発行をサポートしています	Outposts の RDS は、データベースエンジンログの CloudWatch Logs への公開をサポートするようになりました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 11 月 2 日
Amazon RDS Custom for Oracle	Amazon RDS Custom は、基盤となるOSとデータベース環境へのアクセスを必要とするレガシー、カスタム、およびパッケージアプリケーション向けのマネージドデータベースサービスです。Amazon RDS Custom を使用すると、Amazon RDS のオートメーションと Amazon EC2 の柔軟性が得られます。詳細については、「Amazon RDS Custom の使用」を参照してください。	2021 年 10 月 26 日

RDS for MySQL バージョン 8.0 での遅延レプリケーションのサポート	RDS for MySQL バージョン 8.0.26 以降、RDS for MySQL バージョン 8.0 DB インスタンスの RDS の遅延レプリケーションを設定できるようになりました。詳細については、「MySQL での遅延レプリケーションの設定」を参照してください。	2021 年 10 月 25 日
MySQL 8.0.26 のサポート	MySQL バージョン 8.0.26 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2021 年 10 月 25 日
RDS for MySQL バージョン 8.0 の GTID ベースレプリケーションサポート	RDS for MySQL バージョン 8.0.26 以降、RDS for MySQL バージョン 8.0 DB インスタンスにおける GTID ベースのレプリケーションを設定できるようになりました。詳細については、「RDS for MySQL で GTID ベースのレプリケーションを使用する」を参照してください。	2021 年 10 月 25 日
Amazon RDS は RDS for MySQL 8.0 で RDS Proxy をサポート	RDS for MySQL 8.0 データベースインスタンスの RDS Proxy を作成できるようになりました。詳細については、「Amazon RDS Proxy の使用」を参照してください。	2021 年 10 月 21 日

AWSOutposts の Amazon RDS は、MySQL バージョンの追加の RDS をサポートします	Outposts の RDS は、MySQL バージョン 8.0.23 および 8.0.25 の RDS をサポートするようになりました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 10 月 20 日
Amazon RDS for PostgreSQL が PostgreSQL バージョン 14 RC 1 のデータベースプレビュー環境でのサポートスタート	PostgreSQL バージョン 14 RC 1 が米国東部 (オハイオ) AWS リージョン のデータベースプレビュー環境で利用可能になりました。詳細については、「データベースプレビュー環境での作業」を参照してください。	2021 年 10 月 19 日
Amazon RDS が Performance Insights を追加の AWS リージョン でサポート	Performance Insights は中東 (バーレーン)、アフリカ (ケープタウン)、欧州 (ミラノ)、およびアジアパシフィック (大阪) のリージョンで利用できます。詳細については、「Amazon RDS の Performance Insights でサポートされているリージョンと DB エンジン」を参照してください。	2021 年 10 月 5 日

Performance Insights はOracle のダイジェストレベルの統計をサポート	Performance Insights を使用すると、Amazon RDS for Oracle のステートメントレベルとダイジェストレベルの両方で SQL 統計を表示できます。詳細については、「Oracle でのクエリの実行を分析」を参照してください。	2021 年 10 月 4 日
Amazon RDS on AWS Outposts が追加の RDS for PostgreSQL バージョンをサポート	RDS on Outposts が RDS for PostgreSQL バージョン 12.8、13.4 のサポートをスタートしました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 10 月 1 日
Amazon RDS が Oracle APEX バージョン 21.1.v1 をサポート	Oracle APEX 21.1.v1 は、サポートされているすべてのバージョンの Oracle データベースで使用できます。詳細については、「Oracle Application Express」を参照してください。	2021 年 9 月 24 日

Amazon RDS for Oracle で NNE のクライアント側の暗号化をサポート	NNE を設定する場合、サーバー側で暗号化の強制を回避できます。例えば、サーバーで必要としているからといって、すべてのクライアント通信で暗号化の使用を強制することはありません。この場合、SQLNET.*CLIENT オプションを使用してクライアント側で暗号化を強制できます。詳細については、「Oracle native network encryption」を参照してください。	2021 年 9 月 24 日
Amazon RDS for MySQL および RDS for PostgreSQL が新しい DB インスタンスクラスをサポート	db.r5b、db.t4g、および db.x2g インスタンスクラスを使用して、MySQL や PostgreSQL を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2021 年 9 月 15 日

Amazon RDS for Microsoft SQL Server が Microsoft 分散トランザクションコーディネーター (MSDTC) を使用して Java Database Connectivity をサポート	JDBC XA トランザクションが SQL Server 2017 バージョン 14.00.3223.3 以降、および SQL Server 2019 の MSDTC を使用してサポートされるようになりました。詳細については、「Support for Microsoft Distributed Transaction Coordinator in RDS for SQL Server」を参照してください。	2021 年 9 月 7 日
Amazon RDS が MariaDB 10.5.12、10.4.21、10.3.31、10.2.40 をサポート	MariaDB バージョン 10.5.12、10.4.21、10.3.31、10.2.40 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2021 年 9 月 2 日
Amazon RDS が Oracle Database 18c のサポートを終了	DB インスタンスは、Oracle Database 12c および Oracle Database 19c に対してのみ作成できます。Oracle Database 18c スナップショットがある場合は、それ以降のリリースにアップグレードしてください。詳細については、「Oracle DB スナップショットのアップグレード」を参照してください。	2021 年 8 月 17 日

Amazon RDS for PostgreSQL が、PostgreSQL バージョン 14 ベータ 2 をデータベースプレビュー環境でサポートスタート	PostgreSQL バージョン 14 ベータ 1 の詳細については、PostgreSQL 14 ベータ 1 リリースノートを参照してください。PostgreSQL バージョン 14 ベータ 2 の詳細については、PostgreSQL 14 ベータ 2 リリースノートを参照してください。データベースプレビュー環境の詳細については、「データベースプレビュー環境の使用」を参照してください。	2021 年 8 月 9 日
Amazon RDS が共有 VPC で RDS Proxy をサポート	共有 VPC で RDS Proxy を作成できるようになりました。RDS Proxy の詳細については、Amazon RDS ユーザーガイドの「Amazon RDS Proxy による接続の管理」、または Aurora ユーザーガイドを参照してください。	2021 年 8 月 6 日
Amazon RDS が MariaDB 10.2.39 をサポート	MariaDB バージョン 10.2.39 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2021 年 8 月 4 日

Amazon RDS for Oracle が TIMEZONE_FILE_AUTOUPGRADE オプションを追加	オプションを使用すると、現在のタイムゾーンファイルを DB インスタンスの最新バージョンにアップグレードできます。詳細については、Oracle time zone file autoupgrade を参照してください。	2021 年 7 月 30 日
Amazon RDS がクロスリージョン自動バックアップのサポートを拡張	DB スナップショットとトランザクションログを、さらに多くの AWS リージョン 間でレプリケートできるようになりました。詳細については、「別の AWS リージョン への自動バックアップのレプリケーション」を参照してください。	2021 年 7 月 19 日
MySQL 5.7.34 のサポート	MySQL バージョン 5.7.34 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2021 年 7 月 8 日
Amazon RDS on AWS Outposts が追加の RDS for PostgreSQL バージョンをサポート	RDS on Outposts が RDS for PostgreSQL バージョン 12.7、13.3 をサポートをスタートしました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 7 月 8 日

Amazon RDS for PostgreSQL が oracle_fdw をサポート	oracle_fdw エクステンションを使用して、Oracle データベースにアクセスするための外部データラッパーを利用できるようになりました。詳細については、「oracle_fdw エクステンションを使用した外部データへのアクセス」を参照してください。	2021 年 7 月 8 日
Amazon RDS が Oracle Management Agent (OMA) バージョン 13.5 をサポート	Oracle Enterprise Manager (OEM) Cloud Control 13c Release 5 以降では、Oracle Management Agent (OMA) バージョン 13.5 を使用できます。Amazon RDS for Oracle は、Oracle Management Service (OMS) と通信してモニタリング情報を提供する OMA をインストールします。OMS 13.5 を実行する場合、OMA 13.5 をインストールしてデータベースを管理できます。詳細については、Enterprise Manager Cloud Control 向け Oracle Management Agent を参照してください。	2021 年 7 月 7 日

Amazon RDS for Oracle が Simple Storage Service (Amazon S3) からのログのダウンロードをサポート	アーカイブされた REDO ログがインスタンスにはなく、バックアップの保持期間によって守られている場合、`rdsadmin.rdsadmin_archive_log_download` を使用して、Amazon S3 からダウンロードできます。RDS for Oracle は、ログをDB インスタンスの `/rdsdbdata/log/arch` ディレクトリに保存します。詳細については、「Amazon S3 からアーカイブされた REDO ログをダウンロードする」を参照してください。	2021 年 7 月 2 日
Amazon RDS が MariaDB バージョン 10.4.18 と 10.5.9 をサポート	MariaDB バージョン 10.4.18 および 10.5.9 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2021 年 6 月 30 日

Amazon RDS for Oracle がデータベースアクティビティストリーミングをサポート	データベースアクティビティストリーミングを使用して Oracle DB インスタンスをモニタリングできるようになりました。Oracle データベースでは、統合監査証跡に監査レコードが書き込まれます。Oracle DB インスタンスでデータベースアクティビティストリーミングをスタートすると、Amazon Kinesis では Oracle データベースの監査ポリシーに一致するすべてのアクティビティがストリーミングされます。詳細については、「データベースアクティビティストリームによる Amazon RDS のモニタリング」を参照してください。	2021 年 6 月 23 日
Amazon RDS for Oracle でメモリ最適化インスタンスクラスを導入	新しい Oracle DB インスタンスクラスは、vCPU ごとに追加のメモリ、ストレージ、および I/O を必要とするワークロード向けに最適化されています。詳細については、「RDS for Oracle インスタンスクラス」を参照してください。	2021 年 6 月 23 日

MySQL 8.0.25 のサポート	MySQL バージョン 8.0.25 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2021 年 6 月 18 日
Amazon RDS on AWS Outposts が追加の RDS for PostgreSQL バージョンをサポート	RDS on Outposts が RDS for PostgreSQL バージョン 12.5、12.6、13.1、および 13.2 をサポートをスタートしました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 5 月 28 日
Amazon RDS が MariaDB バージョン 10.2.37 と 10.3.28 をサポート	MariaDB バージョン 10.2.37 および 10.3.28 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2021 年 5 月 27 日

Amazon RDS for Oracle がマルチテナントのコンテナデータベース (CDB) をサポート	マルチテナントアーキテクチャにより、Oracle データベースを CDB にすることができます。Oracle Database 19c では、CDB に単一の PDB を含めることができます。PDB のユーザーエクスペリエンスは、非 CDB のユーザーエクスペリエンスとほとんど同じです。詳細については、「RDS for Oracle アーキテクチャ」を参照してください。	2021 年 5 月 25 日
Amazon RDS on AWS Outposts が Amazon RDS for SQL Server をサポート	RDS on Outposts が Amazon RDS for SQL Server のサポートをスタートしました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 5 月 11 日
Amazon RDS がクロスリージョン自動バックアップのサポートを拡張	Microsoft SQL Server を実行している Amazon RDS データベースインスタンスを設定して、DB スナップショットとトランザクションログを別の AWS リージョンにレプリケートできるようになりました。詳細については、自動バックアップの別の AWS リージョンへのレプリケーションを参照してください。	2021 年 5 月 7 日

Amazon RDS が暗号化された DB インスタンスのクロスリージョン自動バックアップをサポート	Oracle または PostgreSQL を実行している暗号化された Amazon RDS データベースインスタンスについて、DB スナップショットとトランザクションログを別の AWS リージョンにレプリケートできるようになりました。詳細については、自動バックアップの別の AWS リージョンへのレプリケーションを参照してください。	2021 年 5 月 3 日
Amazon RDS on AWS Outposts が Amazon CloudWatch モニタリングをサポート	RDS on Outposts が Amazon CloudWatch モニタリングのサポートをスタート 詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2021 年 4 月 21 日
RDS for PostgreSQL が AWS Lambda 関数をサポート	RDS for PostgreSQL DB インスタンスで、AWS Lambda 関数を呼び出すことができるようになりました。詳細については、PostgreSQL DB インスタンスの RDS から AWS Lambda 関数を呼び出すを参照してください。	2021 年 4 月 13 日

RDS for SQL Server は拡張イベントをサポート	SQL Server 拡張イベントを使用して、デバッグおよびトラブルシューティング情報をキャプチャできます。詳細については、「Amazon RDS for Microsoft SQL Server で拡張イベントを使用する」を参照してください。	2021 年 4 月 8 日
MySQL 8.0.23、5.7.33 および 5.6.51 のサポート	MySQL バージョン 8.0.23、5.7.33 および 5.6.51 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2021 年 3 月 31 日
Amazon RDS for MySQL アップグレードの自動ロールバックに失敗しました	MySQL バージョン 5.7 から MySQL バージョン 8.0 への DB インスタンスのアップグレードが失敗した場合、Amazon RDS はアップグレードのために実行された変更を自動的にロールバックします。ロールバック後、MySQL DB インスタンスは MySQL バージョン 5.7 を実行しています。詳細については、「MySQL 5.7 から 8.0 へのアップグレードに失敗した後のロールバック」を参照してください。	2021 年 3 月 18 日

Amazon RDS はオプトインリージョンでクロスリージョンリードレプリカをサポート	DB インスタンスをオプトインリージョンにレプリケートできるようになりました。詳細については、「別の AWS リージョンでのリードレプリカの作成」を参照してください。	2021 年 3 月 18 日
Amazon RDS は Oracle Database 18c を非推奨にすることを計画	Oracle Database 18c (18.0.0.0) は非推奨になる予定です。Oracle Corporation は、サポート終了日以降、Oracle Database 18c のパッチを提供しなくなります。2021 年 7 月 1 日、Amazon RDS は、Oracle Database 18c インスタンスを Oracle Database 19c に自動的にアップグレードすることを計画しています。自動アップグレードをスタートする前に、既存の Oracle Database 18c インスタンスを Oracle Database 19c に手動でアップグレードすることを強くお勧めします。詳細については、「Oracle Database 18c の自動アップグレードの準備」を参照してください。	2021 年 3 月 11 日

Amazon RDS は Oracle Database 11g のサポートが終了しました	DB インスタンスは、Oracle Database 12c Release 1 (12.1.0.2) 以降でのみ作成できます。Oracle Database 11g スナップショットがある場合は、それ以降のリリースにアップグレードしてください。詳細については、「Oracle DB スナップショットのアップグレード」を参照してください。	2021 年 3 月 11 日
Amazon RDS が AWS Backup での DB インスタンスの継続的なバックアップをサポート	これで、AWS Backup で自動バックアップを作成し、このバックアップから指定された時間に DB インスタンスを復元できます。詳細については、「AWS Backup を使用して自動バックアップを管理する」を参照してください。	2021 年 3 月 10 日

Amazon RDS が Oracle Management Agent (OMA) バージョン 13.4 をサポート	Oracle Enterprise Manager (OEM) Cloud Control 13c Release 4 Update 9 では、Oracle Management Agent (OMA) バージョン 13.4 を使用できます。Amazon RDS for Oracle は、Oracle Management Service (OMS) と通信してモニタリング情報を提供する OMA をインストールします。OMS 13.4 を実行する場合は、OMA 13.4 をインストールしてデータベースを管理できます。詳細については、Enterprise Manager Cloud Control 向け Oracle Management Agent を参照してください。	2021 年 3 月 10 日

RDS Proxy エンドポイントのエクステンション	各 RDS Proxy に関連付けられた追加のエンドポイントを作成できます。別の VPC にエンドポイントを作成すると、プロキシの VPC 間アクセスが有効になります。Aurora MySQL クラスターのプロキシは、読み取り専用エンドポイントを持つこともできます。これらのリーダーエンドポイントは、クラスター内のリーダー DB インスタンスに接続し、クエリを多用するアプリケーションの読み取りスケーラビリティと可用性を向上させることができます。RDS Proxy の詳細については、Amazon RDS ユーザーガイドの「Amazon RDS Proxy による接続の管理」、または Aurora ユーザーガイドを参照してください。	2021 年 3 月 8 日
Amazon RDS はクロスリージョン自動バックアップのサポートを拡張	PostgreSQL を実行している Amazon RDS データベースインスタンスを設定して、DB スナップショットとトランザクションログを別の AWS リージョンにレプリケートできるようになりました。詳細については、自動バックアップの別の AWS リージョンへのレプリケーションを参照してください。	2021 年 3 月 8 日

中国 (北京) リージョンおよび中国 (寧夏) リージョンで Amazon RDS for MariaDB および Amazon RDS for MySQL のレプリケーションフィルターがサポート	中国 (北京) リージョンおよび中国 (寧夏) リージョンで、レプリケーションフィルタリングがサポートされるようになりました。詳細については、Configuring replication filters with MariaDB および Configuring replication filters with MySQL を参照してください。	2021 年 3 月 5 日
Amazon RDS はオプトインリージョンでのクロスリージョン DB スナップショットコピーをサポート	DB スナップショットをオプトイン AWS リージョンとの間でコピーできるようになりました。詳細については、「AWS リージョン間のスナップショットのコピー」を参照してください。	2021 年 3 月 4 日
Amazon RDS for SQL Server が Standard Edition の Always On 可用性グループをサポート	SQL Server 2019 でマルチ AZ 設定を使用して Standard Edition データベースエンジン用の DB インスタンスを作成すると、RDS は自動的にアベイラビリティーグループを使用します。詳細については、「Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2021 年 2 月 23 日

Amazon RDS for Oracle では、アドバイザー関連の手順を導入	rdsadmin_util パッケージには、手順 advisor_task_set_parameter 、advisor_task_drop 、dbms_stats_init が含まれています。これらの手順を使用して、AUTO_STATS_ADVISOR_TASK などのアドバイザータスクの変更、停止、および再有効化を行うことができます。詳細については、アドバイザータスクのパラメータの設定を参照してください。	2021 年 2 月 23 日
Amazon RDS は、マルチ AZ DB インスタンスがフェイルオーバーした理由を提供します	マルチ AZ DB インスタンスがスタンバイレプリカにフェイルオーバーしたときの詳細な説明が表示されるようになりました。詳細については、Failover process for Amazon RDS を参照してください。	2021 年 2 月 18 日
Amazon RDS が Amazon S3 へのスナップショットのエクスポートのサポートを拡張	これで、中国 で DB スナップショットデータを Amazon S3 にエクスポートできます。詳細については、「Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。	2021 年 2 月 17 日

Amazon RDS for MariaDB および MySQL のレプリケーションフィルター	MySQL インスタンスと MariaDB インスタンスのレプリケーションフィルターを設定できます。レプリケーションフィルターは、リードレプリカでレプリケートされるデータベースとテーブルを指定します。お客様は、各レプリカに含める、または各レプリカから除外するデータベースとテーブルのリストを作成できます。詳細については、 Configuring replication filters with MariaDB および Configuring replication filters with MySQL を参照してください。	2021 年 2 月 12 日
RDS for Oracle が Oracle APEX 20.2v1 をサポート	Oracle APEX 20.2.v1 は、サポートされているすべてのバージョンの Oracle データベースで使用できます。詳細については、「Oracle Application Express」を参照してください。	2021 年 2 月 2 日

Amazon RDS for SQL Server が、tempdb データベースのローカルインスタンスストレージをサポート	これで、インスタンスストアを使用するように構成された tempdb データベースを使用して、Amazon EC2 db.r5d および db.m5d インスタンスタイプで Amazon RDS for SQL Server を起動できます。tempdb データファイルとログファイルをローカルに配置することで、Amazon EBS に基づくスタンダードストレージベースのサービスに比べて読み取りと書き込みのレイテンシーを低く押さえることができます。詳細については、Instance store support for the tempdb database on Amazon RDS for SQL Server を参照してください。	2021 年 1 月 27 日
Amazon RDS for PostgreSQL が pg_partman と pg_cron をサポート	Amazon RDS for PostgreSQL は、pg_partman および pg_cron エクステンションをサポートするようになりました。pg_partman エクステンションの詳細については、「pg_partman エクステンションを使用した PostgreSQL パーティションの管理」を参照してください。pg_cron エクステンションの詳細については、「PostgreSQL pg_cron エクステンションを使用したメンテナンスのスケジューリング」を参照してください。	2021 年 1 月 12 日

Amazon RDS がOracle Management Agent ログの Amazon CloudWatch Logs への発行をサポート	Oracle Management Agent ログは、emctl.log、emdctlj.log、gcagent.log、gcagent_errors.log、emagent.nohup、および secure.log で構成されます。Amazon RDS は、これらの各ログを個別の CloudWatch ログストリーミングとして発行します。詳細については、「Amazon CloudWatch Logs への Oracle ログの発行」を参照してください。	2020 年 12 月 28 日
Amazon RDS on AWS Outposts は、追加のデータベースバージョンをサポートしています。	RDS on Outposts は、追加の MySQL および PostgreSQL バージョンをサポートするようになりました。詳細については、「Amazon RDS on AWS Outposts による Amazon RDS 機能のサポート」を参照してください。	2020 年 12 月 23 日
Amazon RDS on AWS Outposts は CoIP をサポートしています。	RDS on Outposts は、お客様所有の IP アドレス (CoIP) をサポートするようになりました。CoIP は、オンプレミスネットワークを介して、お客様のOutpost サブネット内のリソースにローカル接続または外部接続を提供します。詳細については、「RDS on Outposts のお客様所有の IP アドレス」を参照してください。	2020 年 12 月 22 日

Amazon RDS for Oracle は BYOL インスタンスの 11g から 19cへの更新を計画	2021 年 1 月 4 日、Bring-Your-Own-License (BYOL) モデルは、Oracle Database 11g インスタンスのすべてのエディションの Oracle Database 19c への自動アップグレードスタートを計画しています。リザーブドインスタンスを含むすべての Oracle Database 11g インスタンスは、最新の利用可能な Release Update (RU) に移動します。詳細については、「Oracle Database 11g BYOL の自動アップグレードの準備」を参照してください。	2020 年 12 月 11 日
Amazon RDS は、自動バックアップの別の AWS リージョンへのレプリケーションをサポートします。	選択した送信先 AWS リージョンに、スナップショットとトランザクションログをレプリケートするよう Amazon RDS データベースインスタンスを設定できるようになりました。詳細については、自動バックアップの別の AWS リージョンへのレプリケーションを参照してください。	2020 年 12 月 4 日

Amazon RDS for Oracle および Microsoft SQL Server が新しい DB インスタンスクラスをサポート	db.r5b インスタンスクラスを使用して、Oracle または SQL Server を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラスでサポートされている DB エンジン」を参照してください。	2020 年 12 月 4 日
MariaDB 10.2.32 のサポート	MariaDB バージョン 10.2.32 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2020 年 11 月 25 日
Amazon RDS for SQL Server が SQL Server 2019 で Microsoft Business Intelligence Suite のサポートをスタート	最新メジャーバージョンを使用して、DB インスタンスで SQL Server Analysis Services、SQL Server Integration Services、および SQL Server Reporting Services を実行できるようになりました。詳細については、「 Microsoft SQL Server データベースエンジンオプション」を参照してください。	2020 年 11 月 24 日

データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 13 をデータベースプレビュー環境でサポートするようになりました。詳細については、PostgreSQL バージョン 13を参照してください。	2020 年 11 月 24 日
Amazon RDS Performance Insights が新しいディメンションを導入	データベース (PostgreSQL、MySQL、および MariaDB)、アプリケーション (PostgreSQL)、セッションタイプ (PostgreSQL) のディメンショングループに従い、データベースのロードをグループ化できます。また Amazon RDS は、ディメンションの db.name (PostgreSQL、MySQL、および MariaDB)、db.application.name (PostgreSQL)、db.session_type.name (PostgreSQL) をサポートしています。詳細については、トップロードテーブルを参照してください。	2020 年 11 月 24 日
Amazon RDS for MariaDB での新しいメジャーバージョンのサポート	MariaDB バージョン 10.5 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2020 年 11 月 23 日

MySQL 5.6.49 のサポート	MySQL バージョン 5.6.49 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 11 月 20 日
MySQL 5.5.62 のサポート	MySQL バージョン 5.5.62 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 11 月 20 日
Performance Insights が実行中の PostgreSQL クエリの統計分析をサポート	PostgreSQL DB インスタンスの Performance Insights を使用して、実行中のクエリの統計を分析できるようになりました。詳細については、「PostgreSQL の統計」を参照してください。	2020 年 11 月 18 日
Amazon RDS がストレージのオートスケーリングのサポートを拡張	リードレプリカの作成、指定時刻への DB インスタンスの復元、または Amazon S3 バックアップからの MySQL DB インスタンスの復元の時に、ストレージのオートスケーリングを有効にできるようになりました。詳細については、「Amazon RDS ストレージのオートスケーリングによる容量の自動管理 」を参照してください。	2020 年 11 月 18 日

Amazon RDS for SQL Server は Database Mail をサポート	Database Mail で、Amazon RDS for SQL Server データベースインスタンスから E メールメッセージを送信できます。E メールの受信者を指定した後、送信するメッセージにファイルまたはクエリ結果を追加できます。詳細については、「Amazon RDS for SQL Server でのデータベースメールの使用」を参照してください。	2020 年 11 月 4 日
MySQL 8.0.21 のサポート	MySQL バージョン 8.0.21 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 10 月 22 日
Amazon RDS が Amazon S3 へのスナップショットのエクスポートのサポートを拡張	すべての商用 AWS リージョンの DB スナップショットデータを Amazon S3 にエクスポートできるようになりました。詳細については、「Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。	2020 年 10 月 22 日

Amazon RDS for PostgreSQL がリードレプリカのアップグレードをサポート	Amazon RDS for PostgreSQL では、プライマリ DB インスタンスのメジャーバージョンアップグレードを実行すると、リードレプリカも自動的にアップグレードされます。詳細については、「PostgreSQL DB エンジンのアップグレード」を参照してください。	2020 年 10 月 15 日
Amazon RDS for MariaDB、MySQL、PostgreSQL が Graviton2 DB インスタンスクラスをサポート	Graviton2 DB インスタンスクラスの db.m6g.x と db.r6g.x を使用して、MariaDB、MySQL、PostgreSQL のいずれかを実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2020 年 10 月 15 日
Amazon RDS for SQL Server が SQL Server 2019 へのアップグレードをサポート	SQL Server DB インスタンスを SQL Server 2019 にアップグレードできます。詳細については、「Microsoft SQL Server DB エンジンのアップグレード」を参照してください。	2020 年 10 月 6 日

Amazon RDS for Oracle が各国語文字セットの指定をサポート	NCHAR 文字セットとも呼ばれる各国語文字セットは、NCHAR、NVARCHAR2 、NCLOB のデータ型で使用されます。データベースの作成時に NCHAR 文字セットとして AL16UTF16 (デフォルト) または UTF8 のいずれかを指定できます。詳細については、Amazon RDS でサポートされている Oracle 文字セットを参照してください。	2020 年 10 月 2 日
MySQL 5.7.31 のサポート	MySQL バージョン 5.7.31 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 10 月 1 日
Amazon RDS for PostgreSQL で Simple Storage Service (Amazon S3) へのデータのエクスポートをサポート	PostgreSQL DB インスタンスからデータをクエリし、Amazon S3 バケットに保存されているファイルに直接エクスポートできます。詳細については、「RDS for PostgreSQL DB インスタンスから Amazon S3 へのデータのエクスポート」を参照してください。	2020 年 9 月 24 日

Amazon RDS for MySQL 8.0 で、Percona XtrabackUp をサポート	Percona XtrabackUp を使用して、バックアップを Amazon RDS for MySQL 8.0 DB インスタンスに復元できるようになりました。詳細については、「MySQL DB インスタンスへのバックアップの復元」を参照してください。	2020 年 9 月 17 日
Amazon RDS for SQL Server でリードレプリカを使用した DB インスタンスでのネイティブバックアップ/復元のサポート	リードレプリカが設定されている DB インスタンスに、SQL Server ネイティブバックアップを復元できます。詳細については、「SQL Server データベースのインポートとエクスポート」を参照してください。	2020 年 9 月 16 日
Amazon RDS for SQL Server が追加のタイムゾーンをサポート	DB インスタンスのタイムゾーンを、選択したタイムゾーンと一致させることができます。詳細については、「Microsoft SQL Server DB インスタンスのローカルタイムゾーン」を参照してください。	2020 年 9 月 11 日
データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン 13 ベータ	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 13 ベータ 3 をデータベースプレビュー環境でサポートするようになりました 詳細については、PostgreSQL バージョン 13を参照してください。	2020 年 9 月 9 日

Amazon RDS for SQL Server がトレースフラグ 692 をサポート	DB パラメータグループを使用して、起動フラグとしてトレースフラグ 692 を使用できるようになりました。このトレースフラグを有効にすると、ヒープまたはクラスター化インデックスにデータを一括でロードする際の高速挿入が無効になります。詳細については、「一括ロード中の高速挿入の無効化」を参照してください。	2020 年 8 月 27 日
Amazon RDS for SQL Server が Microsoft SQL Server 2019 をサポート	SQL Server 2019 を使用する RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での Microsoft SQL Server バージョン」を参照してください。	2020 年 8 月 26 日

Oracle 用 RDS は、マウントされたレプリカデータベースをサポート	Oracle レプリカを作成または変更するときは、マウントモードにすることができます。レプリカデータベースはユーザー接続を受け付けないため、読み取り専用ワークロードを処理できません。マウントされたレプリカは、アーカイブされた REDO ログファイルを適用した後に削除します。マウントされたレプリカの主な用途は、クロスリージョンの災害対策です。詳細については、「Oracle レプリカの概要」を参照してください。	2020 年 8 月 13 日
RDS for Oracle で 11g SE1 LI インスタンスのアップグレードを予定	2020 年 11 月 1 日に、Oracle Database 11g SE1 License Included (LI) インスタンスの Amazon RDS for Oracle 用 Oracle Database 19c への自動アップグレードがスタートされる予定です。リザーブドインスタンスを含むすべての 11g インスタンスが最新の利用可能な Oracle Release Update (RU) に移動されます。詳細については、「Oracle Database 11g SE1 の自動アップグレードの準備」を参照してください。	2020 年 7 月 31 日

Amazon RDS が PostgreSQL および MySQL のプレビューリリースで新しい Graviton2 DB インスタンスクラスをサポート	db.m6g.x および db.r6g.x DB インスタンスクラスを使用する PostgreSQL または MySQL を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2020 年 7 月 30 日
RDS for Oracle が Oracle APEX 20.1v1 をサポート	Oracle APEX 20.1.v1 は、サポートされているすべてのバージョンの Oracle データベースで使用できます。詳細については、「Oracle Application Express」を参照してください。	2020 年 7 月 28 日
MySQL 8.0.20 のサポート	MySQL バージョン 8.0.20 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 7 月 23 日

Amazon RDS for MariaDB および MySQL が新しい DB インスタンスクラスをサポート	db.m5.16xlarge、db.m5.8xlarge、db.r5.16xlarge、および db.r5.8xlarge DB インスタンスクラスを使用する、MariaDB と MySQL を実行している Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2020 年 7 月 23 日
SQL Server 用 RDS では、古いバージョンの TLS と暗号の無効化がサポートされています	特定のセキュリティプロトコルと暗号のオン/オフを切り替えることができます。詳細については、「セキュリティプロトコルと暗号の設定」を参照してください。	2020 年 7 月 21 日
RDS で Oracle Spatial SE2 をサポート	12.2、18c、19c のすべてのバージョンで Standard Edition 2 (SE2) の Oracle Spatial を使用できます。詳細については、「Oracle Spatial」を参照してください。	2020 年 7 月 9 日

Amazon RDS が AWS PrivateLink をサポート	Amazon RDS で Amazon RDS API コールの Amazon VPC エンドポイントを作成し、AWS ネットワーク内のアプリケーションと Amazon RDS 間のトラフィックを維持できるようになりました。詳細については、「Amazon RDS とインターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。	2020 年 7 月 9 日
Amazon RDS for PostgreSQL バージョン 9.4.x はサポートを終了しました。	Amazon RDS for PostgreSQL はバージョン 9.4.x をサポートしなくなりました。サポートされているバージョンについては、「 サポートされている PostgreSQL データベースのバージョン」を参照してください。	2020 年 7 月 8 日
MariaDB 10.3.23 および 10.4.13 のサポート	MariaDB バージョン 10.3.23 および 10.4.13 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2020 年 7 月 6 日
Amazon RDS on AWS Outposts	Amazon RDS DB インスタンスは、AWS Outposts 上で作成できます。詳細については、「AWS Outposts on Amazon RDS の使用」を参照してください。	2020 年 7 月 6 日

Amazon RDS for Oracle はインベントリファイルを自動的に作成	BYOL のお客様用のサービスリクエストを開くため、Oracle Support では Opatch によって生成されるインベントリファイルがリクエストされます。Amazon RDS for Oracle を使用して、BDUMP ディレクトリに 1 時間ごとにインベントリファイルを自動で作成できます。詳細については、「Opatch ファイルへのアクセス」を参照してください。	2020 年 7 月 6 日
MySQL 5.7.30 および 5.6.48 のサポート	MySQL バージョン 5.7.30 および 5.6.48 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 6 月 25 日

Amazon RDS for Oracle は ADRCI をサポートします	自動診断リポジトリコマンドインタープリタ (ADRCI) ユーティリティは、診断データの管理に使用する Oracle コマンドラインツールです。Amazon RDS パッケージ `rdsadmin_adrci_util` 内の関数を使用すると、問題やインシデントをリストしてパッケージ化でき、トレースファイルも表示できます。詳細については、「Oracle DB インスタンスの一般的な DBA 診断タスク」を参照してください。	2020 年 6 月 17 日
MySQL 8.0.19 のサポート	MySQL バージョン 8.0.19 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 6 月 2 日
MySQL 8.0 は、小文字のテーブル名をサポート	MySQL バージョン 8.0.19 および 8.0 以降のバージョンを実行している Amazon RDS DB インスタンスでは、`lower_case_table_names` パラメータを 1 に設定できるようになりました。詳細については、「Amazon RDS DB インスタンスの MySQL パラメータの例外」を参照してください。	2020 年 6 月 2 日

Amazon RDS for Microsoft SQL Server による SQL Server Integration Services (SSIS) のサポート	SSIS は、データ統合およびワークフローアプリケーションに対応したプラットフォームです。SSIS は、既存のまたは新しい DB インスタンスで有効にすることができます。データベースエンジンと同じ DB インスタンスにインストールされます。詳細については、「SQL Server での SQL Server Integration Services のサポート」を参照してください。	2020 年 5 月 19 日
Amazon RDS for Microsoft SQL Server が SQL Server Reporting Services (SSRS) をサポート	SSRS は、レポートの生成とディストリビューションに使用されるサーバーベースのアプリケーションです。SSRS は、既存または新規の DB インスタンスで有効にすることができます。データベースエンジンと同じ DB インスタンスにインストールされます。詳細については、「SQL Server での SQL Server Reporting Services のサポート」を参照してください。	2020 年 5 月 15 日

Amazon RDS for Microsoft SQL Server がマルチ AZ インスタンスで S3 統合をサポート	マルチ AZ DB インスタンスでは、一括挿入などの SQL Server 機能に Amazon S3 を使用できるようになりました。詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。	2020 年 5 月 15 日
Amazon RDS for Oracle がごみ箱を空にする操作をサポート	`rdsadmin.rdsadmin_util.purge_dba_recyclebin` プロシージャは、ごみ箱を空にします。詳細については、「ごみ箱を空にする」を参照してください。	2020 年 5 月 13 日
Amazon RDS for Oracle により自動ワークロードリポジトリ (AWR) の管理性が向上	`rdsadmin.rdsadmin_diagnostic_util` プロシージャは、AWR レポートを生成し、AWR データをダンプファイルに抽出します。詳細については、「自動ワークロードリポジトリ (AWR) を使用したパフォーマンスレポートの生成」を参照してください。	2020 年 5 月 13 日
Amazon RDS for Microsoft SQL Server が マイクロソフト分散トランザクションコーディネーター (MSDTC) をサポート	Amazon RDS for SQL Server は、ホスト間の分散トランザクションをサポートしています。詳細については、「SQL Server での Microsoft 分散トランザクションコーディネーターのサポート」を参照してください。	2020 年 5 月 4 日

Amazon RDS for Microsoft SQL Server が新しいバージョンをサポート	SQL Server バージョン 2017 CU19 14.00.3281.6、2016 SP2 CU11 13.00.5598.27、2014 SP3 CU4 12.00.6329.1、および 2012 SP4 GDR 11.0.7493.4 を実行する Amazon RDS DB インスタンスをすべての工ディションで作成できるようになりました。詳細については、「Amazon RDS での Microsoft SQL Server バージョン」を参照してください。	2020 年 4 月 28 日
Amazon RDS が 欧州 (ミラノ) リージョン で利用可能に	Amazon RDS が 欧州 (ミラノ) リージョン で利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2020 年 4 月 28 日
Amazon RDS による Local Zones のサポート	DB インスタンスをローカルゾーンのサブネット内で起動できるようになりました。詳細については、「リージョン、アベイラビリティーゾーン、および Local Zones」を参照してください。	2020 年 4 月 23 日
Amazon RDS が アフリカ (ケープタウン) リージョン で利用可能に	Amazon RDS が アフリカ (ケープタウン) リージョン で利用可能になりました。詳細については、「リージョンとアベイラビリティーゾーン」を参照してください。	2020 年 4 月 22 日

Amazon RDS for Microsoft SQL Server は SQL Server Analysis Services (SSAS) をサポートしています。	SSAS は、SQL Server 内にインストールされているオンライン分析処理 (OLAP) およびデータマイニングツールです。SSAS は、既存または新規の DB インスタンスで有効にすることができます。データベースエンジンと同じ DB インスタンスにインストールされます。詳細については、「SQL Server での SQL Server Analytic Services のサポート」を参照してください。	2020 年 4 月 17 日
PostgreSQL 用の Amazon RDS Proxy	Amazon RDS Proxy が PostgreSQL で利用可能になりました。RDS Proxy を使用すると、DB インスタンスでの接続管理のオーバーヘッドを削減し、「接続が多すぎます」というエラーが発生する可能性も減らすことができます。RDS Proxy は、現在 PostgreSQL でパブリックプレビュー中です。詳細については、「Amazon RDS Proxy による接続の管理 (プレビュー)」を参照してください。	2020 年 4 月 8 日

Amazon RDS for Oracle が Oracle APEX バージョン 19.2.v1 をサポート	Amazon RDS for Oracle が Oracle Application Express (APEX) バージョン 19.2.v1 をサポートするようになりました。詳細については、「Oracle Application Express」を参照してください。	2020 年 4 月 8 日
Amazon RDS for MariaDB での新しいメジャーバージョンのサポート	MariaDB バージョン 10.4 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2020 年 4 月 6 日
Amazon RDS for MariaDB 10.4 で Amazon RDS Performance Insights が利用可能	Amazon RDS Performance Insights が Amazon RDS for MariaDB バージョン 10.4 で使用できるようになりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2020 年 4 月 6 日
Amazon RDS for PostgreSQL バージョン 9.3.x はサポートを終了しました	Amazon RDS for PostgreSQL はバージョン 9.3.x をサポートしなくなりました。サポートされているバージョンについては、「 サポートされている PostgreSQL データベースのバージョン」を参照してください。	2020 年 4 月 3 日

Amazon RDS for Microsoft SQL Server がリードレプリカをサポート	SQL Server DB インスタンスのリードレプリカを作成できるようになりました。詳細については、「リードレプリカの使用」を参照してください。	2020 年 4 月 3 日
Amazon RDS for Microsoft SQL Server が複数ファイルのバックアップをサポート	SQL Server のネイティブバックアップおよび復元を使用して、データベースを複数のファイルにバックアップできるようになりました。詳細については、「データベースのバックアップ」を参照してください。	2020 年 4 月 2 日
と Amazon RDS for Oracle が統合AWS License Manager	Amazon RDS for Oracle が AWS License Manager と統合されました。Bring-Your-Own-License モデルを使用する場合は、AWS License Manager の統合により、組織内での Oracle ライセンスの使用状況のモニタリングが容易になります。詳細については、「AWS License Managerとの統合」を参照してください。	2020 年 3 月 23 日
Amazon RDS for MariaDB および MySQL の db.r5 インスタンスでの 64 TiB のサポート	最大 64 TiB のストレージを持つ db.r5 DB インスタンスクラスを使用する MariaDB および MySQL 用の Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「ストレージのパフォーマンスに影響する要因」を参照してください。	2020 年 3 月 18 日

MySQL 8.0.17 のサポート	MySQL バージョン 8.0.17 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 3 月 10 日
Amazon RDS Performance Insights が Amazon RDS for MySQL 8.0 で利用可能	Amazon RDS Performance Insights は、Amazon RDS for MySQL バージョン 8.0.17 以降の 8.0 バージョンで利用できるようになりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2020 年 3 月 10 日
MySQL 5.6.46 のサポート	MySQL バージョン 5.6.46 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 2 月 28 日
Amazon RDS Performance Insights が Amazon RDS for MariaDB 10.3 で利用可能	Amazon RDS Performance Insights が Amazon RDS for MariaDB バージョン 10.3.13 以降の 10.3 バージョンで利用できるようになりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2020 年 2 月 26 日

MySQL 5.7.28 のサポート	MySQL バージョン 5.7.28 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2020 年 2 月 20 日
MariaDB 10.3.20 のサポート	MariaDB バージョン 10.3.20 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2020 年 2 月 20 日
Amazon RDS for Microsoft SQL Server が新しい DB インスタンスクラスをサポート	db.z1d DB インスタンスクラスを使用する SQL Server を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。	2020 年 2 月 19 日
Amazon RDS for SQL Server でのクロスアカウント、クロス VPC Active Directory ドメインのサポート	Amazon RDS for Microsoft SQL Server では、異なるアカウントおよび VPC が所有する Active Directory ドメインと DB インスタンスとの関連付けをサポートするようになりました。詳細については、「Microsoft SQL Server DB インスタンスでの Windows 認証の使用」を参照してください。	2020 年 2 月 13 日

Oracle OLAP オプション	Amazon RDS for Oracle では、Oracle DB インスタンスのオンライン分析処理 (OLAP) オプションがサポートされるようになりました。Oracle OLAP を使用すると、OLAP スタンダードに従って次元オブジェクトやキューブを作成することにより、大量のデータを分析できます。詳細については、「Oracle OLAP」を参照してください。	2020 年 2 月 13 日
Oracle に対する FIPS 140-2 のサポート	Amazon RDS for Oracle は、SSL/TLS 接続に対し、連邦情報処理規格公告 140-2 (FIPS 140-2) をサポートしています。詳細については、「FIPS のサポート」を参照してください。	2020 年 2 月 11 日
Amazon RDS for PostgreSQL で新しい DB インスタンスクラスをサポート	db.m5.16xlarge、db.m5.8xlarge、db.r5.16xlarge、および db.r5.8xlarge DB インスタンスクラスを使用する、PostgreSQL を実行している Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「使用可能なすべての DB インスタンスクラスでサポートされる DB エンジン」を参照してください。	2020 年 2 月 11 日

Performance Insights は実行中の MariaDB と MySQL クエリの統計分析をサポートします	Performance Insights for MariaDB と MySQL DB インスタンスを使用して、実行中のクエリの統計を分析できるようになりました。詳細については、「クエリの実行の統計を分析する」を参照してください。	2020 年 2 月 4 日
MariaDB、MySQL、および PostgreSQL の DB スナップショットデータを Amazon S3 にエクスポートするためのサポート	Amazon RDS では、MariaDB、MySQL、および PostgreSQL の DB スナップショットデータを Amazon S3 にエクスポートできます。詳細については、「Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。	2020 年 1 月 23 日
Amazon RDS for MySQL が Kerberos 認証をサポート	Amazon RDS for MySQL DB インスタンスに接続するユーザーを Kerberos 認証を使用して認証できるようになりました。詳細については、「MySQL で Kerberos 認証を使用する」を参照してください。	2020 年 1 月 21 日

Amazon RDS Performance Insights で Amazon RDS for Microsoft SQL Server 用の SQL テキストをより多く表示可能に	Amazon RDS Performance Insights は、Amazon RDS for Microsoft SQL Server DB インスタンス用の SQL テキストをより多く Performance Insights ダッシュボードで表示できるようになりました。詳細については、「Performance Insights ダッシュボードでの SQL テキストの表示量を増やす」を参照してください。	2019 年 12 月 17 日
Amazon RDS Proxy	Amazon RDS Proxy を使用すると、クラスターでの接続管理のオーバーヘッドを減らし、「接続が多すぎます」エラーが発生する可能性を減らすことができます。各プロキシを RDS DB インスタンスまたは Aurora DB クラスターに関連付けます。次に、アプリケーションの接続文字列でプロキシエンドポイントを使用します。Amazon RDS Proxy は現在パブリックプレビュー状態です。これは、RDS for MySQL データベースエンジンをサポートしています。詳細については、「Amazon RDS Proxy による接続の管理 (プレビュー)」を参照してください。	2019 年 12 月 3 日

Amazon RDS on AWS Outposts (プレビュー)	Amazon RDS on AWS Outposts を使用すると、オンプレミスのデータセンターに AWS 管理のリレーショナルデータベースを作成できます。RDS on Outposts を使用すると、AWS Outposts で RDS データベースを実行できます。詳細については、「Amazon RDS on AWS Outposts (プレビュー)」を参照してください。	2019 年 12 月 3 日
Amazon RDS for Oracle でクロスリージョンリードレプリカをサポート	Amazon RDS for Oracle で、Active Data Guard を使用したクロスリージョンリードレプリカをサポートするようになりました。詳細については、「リードレプリカの使用」と「Oracle リードレプリカの使用」を参照してください。	2019 年 11 月 26 日
Performance Insights が実行中の Oracle クエリの統計分析をサポート	Oracle DB インスタンスの Performance Insights を使用して、実行中のクエリの統計を分析できるようになりました。詳細については、「クエリの実行の統計を分析する」を参照してください。	2019 年 11 月 25 日

Amazon RDS for Microsoft SQL Server は、CloudWatch Logs へのログの発行をサポートしています	ログイベントを Amazon CloudWatch Logs に直接公開するように、Amazon RDS for SQL Server DB インスタンスを設定できます。詳細については、「Amazon CloudWatch Logs への SQL Server ログの発行」を参照してください。	2019 年 11 月 25 日
Amazon RDS for Microsoft SQL Server が新しい DB インスタンスクラスをサポート	SQL Server を実行する Amazon RDS DB インスタンスを db.x1e および db.x1 DB インスタンスクラスで作成できるようになりました。詳細については、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。	2019 年 11 月 25 日
Amazon RDS for Microsoft SQL Server は差分復元とログ復元をサポートしています	SQL Server のネイティブバックアップと復元を使用して、差分バックアップとログを復元できます。詳細については、「ネイティブバックアップおよび復元」を参照してください。	2019 年 11 月 25 日
新しいリージョンの Amazon RDS for Microsoft SQL Server でサポートされているマルチ AZ	SQL Server の マルチ AZ が、中国、中東 (バーレーン)、および 欧州 (ストックホルム) で使用できるようになりました。詳細については、「Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2019 年 11 月 22 日

Amazon RDS for Microsoft SQL Server は一括挿入と S3 統合をサポートするようになりました	SQL Server DB インスタンスと Amazon S3 バケットの間でファイルを転送することができます。その後、一括挿入 などの SQL Server 特性で Amazon S3 を使用することができます。詳細については、「Amazon RDS for SQL Server DB インスタンスと Amazon S3 の統合」を参照してください。	2019 年 11 月 21 日
Amazon RDS for Microsoft SQL Server の Performance Insights カウンター	Microsoft SQL Server DB インスタンスの Performance Insights 図にパフォーマンスカウンターを追加できるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server の Performance Insights カウンター」を参照してください。	2019 年 11 月 12 日
Amazon RDS for Microsoft SQL Server が新しい DB インスタンスクラスサイズをサポート	db.m5 および db.r5 DB インスタンスクラスに 8xlarge および 16xlarge インスタンスサイズを使用する SQL Server を実行する Amazon RDS DB インスタンスを作成できるようになりました。db.t3 インスタンスクラスでは、small から 2xlarge までのインスタンスサイズが利用可能になりました。詳細については、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。	2019 年 11 月 11 日

PostgreSQL スナップショットアップグレードのサポート	Amazon RDS PostgreSQL DB インスタンスの既存の手動 DB スナップショットがある場合は、PostgreSQL データベースエンジンの新しいバージョンにアップグレードできます。詳細については、「PostgreSQL DB スナップショットのアップグレード」を参照してください。	2019 年 11 月 7 日
Amazon RDS for Oracle で新しいメジャーバージョンをサポート	Oracle Database 19c (19.0) を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での Oracle Database 19c」を参照してください。	2019 年 11 月 7 日
データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 12.0 をデータベースプレビュー環境でサポートするようになりました。詳細については、「データベースプレビュー環境における PostgreSQL バージョン 12.0」を参照してください。	2019 年 11 月 1 日

Amazon RDS for PostgreSQL は Kerberos 認証をサポートしています	ユーザーが PostgreSQL を実行している Amazon RDS DB インスタンスに接続する場合、Kerberos 認証を使用してユーザーを認証できるようになりました。詳細については、「Amazon RDS for PostgreSQL で Kerberos 認証を使用する」を参照してください。	2019 年 10 月 28 日
Oracle DB インスタンスの OEM Management Agent データベースタスク	Amazon RDS for Oracle DB インスタンスは、現在、Management Agent で特定の EMCTL コマンドを起動する手順をサポートしています。詳細に関しては、「OEM Agent データベースタスク」を参照してください。	2019 年 10 月 24 日
Amazon RDS for PostgreSQL が PostgreSQL トランスポータブルデータベースをサポート	PostgreSQL トランスポータブルデータベースは、2 つの DB インスタンス間で RDS PostgreSQL データベースを超高速で移行する方法を提供しています。詳細については、「DB インスタンス間での PostgreSQL データベースの移行」を参照してください。	2019 年 10 月 8 日

Amazon RDS for Oracle が Kerberos 認証をサポート	ユーザーが Oracle を実行している Amazon RDS DB インスタンスに接続する場合、Kerberos 認証を使用してユーザーを認証できるようになりました。詳細については、「Amazon RDS for Oracle で Kerberos 認証を使用する」を参照してください。	2019 年 9 月 30 日
データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン 12 ベータ	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 12 ベータ 3 をデータベースプレビュー環境でサポートするようになりました 詳細については、「データベースプレビュー環境の Amazon RDS における PostgreSQL バージョン 12 ベータ 3」を参照してください。	2019 年 8 月 28 日
MySQL 8.0.16 のサポート	MySQL バージョン 8.0.16 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2019 年 8 月 19 日
Amazon RDS for Oracle で新しいメジャーバージョンをサポート	Oracle Database 18c (18.0) を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での Oracle Database 18c」を参照してください。	2019 年 8 月 15 日

OEM 13c リリース 3 の Management Agent	Amazon RDS for Oracle DB インスタンスで、Oracle Enterprise Manager (OEM) Cloud Control 13c Release 3 の Management Agent のサポートがスタートされました。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。	2019 年 8 月 7 日
データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン 12 ベータ	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 12 ベータ 2 をデータベースプレビュー環境でサポートするようになりました。詳細については、「データベースプレビュー環境の Amazon RDS における PostgreSQL バージョン 12 ベータ 2」を参照してください。	2019 年 8 月 6 日
Amazon RDS で SQL Server のサーバー照合をサポート	Amazon RDS for SQL Server では、新しい DB インスタンスの照合の選択をサポートしています。詳細については、「Microsoft SQL Server の照合順序と文字セット」を参照してください。	2019 年 7 月 29 日

Amazon RDS for Oracle が Oracle APEX バージョン 19.1.v1 をサポート	Amazon RDS for Oracle が Oracle Application Express (APEX) バージョン 19.1.v1 をサポートするようになりました。詳細については、「Oracle Application Express」を参照してください。	2019 年 6 月 28 日
データベースプレビュー環境の Amazon RDS for PostgreSQL バージョン 13 ベータ	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 13 ベータ 1 をデータベースプレビュー環境でサポートするようになりました 詳細については、PostgreSQL バージョン 13を参照してください。	2019 年 6 月 22 日
Amazon RDS ストレージのオートスケーリング	Amazon RDS DB インスタンスのストレージのオートスケーリングを使用すると、Amazon RDS で DB インスタンスに関連付けられているストレージを自動的に拡張して、スペース不足になる可能性を抑えることができます。ストレージのオートスケーリングについては、「Amazon RDS DB インスタンスのストレージの使用」を参照してください。	2019 年 6 月 20 日

Amazon RDS for Oracle で db.z1d DB インスタンスクラスをサポート	db.z1d DB インスタンスクラスを使用する Oracle を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2019 年 6 月 13 日
Amazon RDS Performance Insights で Amazon RDS for Oracle の SQL テキストをさらに表示可能に	Amazon RDS Performance Insights で、Amazon RDS for Oracle DB インスタンスの Performance Insights ダッシュボードで SQL テキストをより多く表示できるようになりました。詳細については、「Performance Insights ダッシュボードでの SQL テキストの表示量を増やす」を参照してください。	2019 年 6 月 10 日
Amazon RDS には最大 16 TB の SQL Server データベースのネイティブ復元のサポートが追加されています	これで、SQL Server から Amazon RDS に最大 16 TB のネイティブ復元を実行できます。詳細については、「Amazon RDS for SQL Server: 制限と推奨事項」を参照してください。	2019 年 6 月 4 日

Amazon RDS が Microsoft SQL Server Audit のサポートを追加	Amazon RDS for Microsoft SQL Server を使用すると、SQL Server Audit を使用してサーバーおよびデータベースレベルのイベントを監査し、結果を DB インスタンスで表示したり、監査ログファイルを直接 Amazon S3 に送信したりできます。詳細については、「SQL Server Audit」を参照してください。	2019 年 5 月 23 日
Amazon RDS 推奨事項の改善	Amazon RDS は、データベースリソースに対して自動化された推奨事項を改善しました。例えば、Amazon RDS では、データベースパラメータの推奨事項が提供されています。詳細については、「Amazon RDS 推奨事項を使用する」を参照してください。	2019 年 5 月 22 日
Amazon RDS for SQL Server の DB インスタンスあたりのサポート対象のデータベースを拡大	Microsoft SQL Server を実行している DB インスタンスごとに最大 30 のデータベースを作成できます。詳細については、「Microsoft SQL Server DB インスタンスの制限」を参照してください。	2019 年 5 月 21 日

Amazon RDS for MariaDB、MySQL、および PostgreSQL 用に 64 TiB および 80k IOPS のストレージをサポート	最大 64 TiB のストレージと最大 80,000 プロビジョンド IOPS を備えた MariaDB、MySQL、PostgreSQL 用 Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスストレージ」を参照してください。	2019 年 5 月 20 日
Amazon RDS for MySQL でアップグレードの事前確認をサポート	DB インスタンスを MySQL 5.7 から MySQL 8.0 にアップグレードすると、Amazon RDS は非互換性について事前チェックを実行します。詳細については、「MySQL 5.7 から 8.0 へのアップグレードの事前確認」を参照してください。	2019 年 5 月 17 日
MySQL パスワード検証プラグインのサポート	MySQL `validate_password` プラグインを使用して、Amazon RDS for MySQL DB インスタンスのセキュリティを高められるようになりました。詳細については、「パスワード検証プラグインの使用」を参照してください。	2019 年 5 月 16 日

Amazon RDS for Oracle の Performance Insights カウンター	Oracle DB インスタンスの Performance Insights 図にパフォーマンスカウンターを追加できるようになりました。詳細については、「Amazon RDS for Oracle の Performance Insights カウンター」を参照してください。	2019 年 5 月 8 日
秒単位の請求のサポート	オンデマンドインスタンスの場合、Amazon RDS は、すべての AWS リージョン (AWS GovCloud (米国) は除く) で 1 秒単位で請求されます。詳細については、「Amazon RDS DB インスタンスの請求」を参照してください。	2019 年 4 月 25 日
Amazon S3 からの Amazon RDS for PostgreSQL のデータのインポートのサポート	Amazon S3 ファイルから、RDS PostgreSQL DB インスタンスのテーブルにデータをインポートできるようになりました。詳細については、「RDS PostgreSQL DB インスタンスへの Amazon S3 データのインポート」を参照してください。	2019 年 4 月 24 日

Amazon S3 から 5.7 のバックアップを復元するためのサポート	MySQL バージョン 5.7 のバックアップを作成して Amazon S3 に保存し、MySQL を実行する新しい Amazon RDS DB インスタンスにバックアップファイルを復元できます。詳細については、「MySQL DB インスタンスへのバックアップの復元」を参照してください。	2019 年 4 月 17 日
Amazon RDS for PostgreSQL の複数のメジャーバージョンのアップグレードをサポート	Amazon RDS for PostgreSQL を使用して、DB エンジンのアップグレード時に複数のメジャーバージョンから選択できるようになりました。この機能では、特定の PostgreSQL エンジンのバージョンのアップグレード時に、新しいメジャーバージョンにスキップすることができます。詳細については、「PostgreSQL DB エンジンのアップグレード」を参照してください。	2019 年 4 月 16 日
Amazon RDS for Oracle で 64 TiB ストレージのサポート	最大 64 TiB のストレージと最大 80,000 プロビジョンド IOPS を備えた Oracle 用 Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスストレージ」を参照してください。	2019 年 4 月 4 日

MySQL 8.0.15 のサポート	MySQL バージョン 8.0.15 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2019 年 4 月 3 日
MariaDB 10.3.13 のサポート	MariaDB バージョン 10.3.13 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2019 年 4 月 3 日
Microsoft SQL Server 2008 R2 は、Amazon RDS でのサポートを終了しました	Microsoft SQL Server 2008 R2 は、2019 年 7 月 9 日にこのバージョンの拡張サポートを終了するという Microsoft の計画と一致して、サポートを終了しました。2019 年 6 月 1 日より、既存の Microsoft SQL Server 2008 R2 スナップショットは、最新のマイナーバージョンの Microsoft SQL Server 2012 に自動アップグレードされます。詳細については、「Amazon RDS での Microsoft SQL Server 2008 R2 のサポート」を参照してください。	2019 年 4 月 2 日

Microsoft SQL Server 2017 で Always On 可用性グループをサポート	SQL Server 2017 Enterprise Edition 14.00.3049.1 以降で Always On 可用性グループがサポートされるようになりました。詳細については、「Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2019 年 3 月 29 日
ボリュームメトリクスの表示	データベースおよびログストレージに使用される物理デバイスである、Amazon Elastic Block Store (Amazon EBS) ボリュームのメトリクスを表示できるようになりました。詳細については、「拡張モニタリングの表示」を参照してください。	2019 年 3 月 20 日
MySQL 5.7.25 のサポート	MySQL バージョン 5.7.25 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2019 年 3 月 19 日
Amazon RDS for Oracle で RMAN DBA タスクをサポート	Amazon RDS for Oracle で、Oracle Recovery Manager (RMAN) の DBA タスク (RMAN バックアップ) がサポートされるようになりました。詳細については、「Oracle DB インスタンスの一般的な DBA Recovery Manager (RMAN) タスク」を参照してください。	2019 年 3 月 14 日

Amazon RDS for PostgreSQL でバージョン 11.1 のサポートをスタート	PostgreSQL バージョン 11.1 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での PostgreSQL バージョン 11.1」を参照してください。	2019 年 3 月 12 日
Amazon RDS for SQL Server で複数のファイル復元が可能に	Amazon RDS for SQL Server を使用して複数のファイルから復元できるようになりました。詳細については、「データベースの復元」を参照してください。	2019 年 3 月 11 日
MariaDB 10.2.21	MariaDB バージョン 10.2.21 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2019 年 3 月 11 日
Amazon RDS for Oracle でリードレプリカをサポート	Amazon RDS for Oracle で Active Data Guard を使用したリードレプリカをサポートするようになりました。詳細については、「リードレプリカの使用」と「Oracle リードレプリカの使用」を参照してください。	2019 年 3 月 11 日

Amazon RDS の Performance Insights が Amazon RDS for MariaDB で利用可能に	Amazon RDS の Performance Insights が Amazon RDS for MariaDB で使用できるようになりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2019 年 3 月 11 日
MySQL 8.0.13 と 5.7.24	MySQL バージョン 8.0.13 および 5.7.24 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2019 年 3 月 8 日
Amazon RDS の Performance Insights が Amazon RDS for SQL Server で利用可能に	Amazon RDS の Performance Insights が Amazon RDS for SQL Server で使用できるようになりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2019 年 3 月 4 日
Amazon RDS for Oracle で Amazon S3 統合のサポートをスタート	Amazon RDS for Oracle DB インスタンスと Amazon S3 バケットの間でファイルを転送できるようになりました。詳細については、「Amazon RDS for Oracle と Amazon S3 の統合」を参照してください。	2019 年 2 月 26 日

Amazon RDS for MySQL と Amazon RDS for MariaDB で DB インスタンスクラス db.t3 のサポートをスタート	DB インスタンスクラス db.t3 を使用する MySQL または MariaDB を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2019 年 2 月 20 日
Amazon RDS for MySQL と Amazon RDS for MariaDB で DB インスタンスクラス db.r5 のサポートをスタート	DB インスタンスクラス db.r5 を使用する MySQL または MariaDB を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2019 年 2 月 20 日
RDS for MySQL と PostgreSQL の Performance Insights のカウンター	MySQL と PostgreSQL DB インスタンスの Performance Insights 図にパフォーマンスカウンターを追加できるようになりました。詳細については、「Performance Insights ダッシュボードのコンポーネント」を参照してください。	2019 年 2 月 19 日
Amazon RDS for PostgreSQL で適応可能な autovacuum パラメータチューニングのサポートをスタート	Amazon RDS for PostgreSQL を使用して適応可能な autovacuum パラメータチューニングは、autovacuum パラメータ値の自動調整によってトランザクション ID の循環を防ぐのに役立ちます。詳細については、「トランザクション ID の循環の可能性を減らす」を参照してください。	2019 年 2 月 12 日

Amazon RDS for Oracle で Oracle APEX バージョン 18.1.v1 および 18.2.v1 をサポート	Amazon RDS for Oracle で Oracle Application Express (APEX) バージョン 18.1.v1 および 18.2.v1 がサポートされるようになりました。詳細については、「Oracle Application Express」を参照してください。	2019 年 2 月 11 日
Amazon RDS Performance Insights で RDS for MySQL テキストをさらに表示可能に	Amazon RDS Performance Insights で、MySQL DB インスタンスの Performance Insights ダッシュボードで SQL テキストをより多く表示できるようになりました。詳細については、「Performance Insights ダッシュボードでの SQL テキストの表示量を増やす」を参照してください。	2019 年 2 月 6 日
Amazon RDS for PostgreSQL で DB インスタンスクラス db.t3 のサポートをスタート	DB インスタンスクラス db.t3 を使用する PostgreSQL を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2019 年 1 月 25 日
Amazon RDS for Oracle で DB インスタンスクラス db.t3 をサポート	DB インスタンスクラス db.t3 を使用する Oracle を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2019 年 1 月 25 日

Amazon RDS Performance Insights で Amazon RDS PostgreSQL の SQL テキストをさらに表示可能に	Amazon RDS Performance Insights で、Amazon RDS PostgreSQL DB インスタンスの Performance Insights ダッシュボードで SQL テキストをより多く表示できるようになりました。詳細については、「Performance Insights ダッシュボードでの SQL テキストの表示量を増やす」を参照してください。	2019 年 1 月 24 日
Amazon RDS for Oracle が SQLT の新しいバージョンをサポートするようになりました	Amazon RDS for Oracle が SQLT バージョン 12.2.180725 をサポートするようになりました。詳細については、「Oracle SQLT」を参照してください。	2019 年 1 月 22 日
Amazon RDS for PostgreSQL で db.r5 DB インスタンスクラスをサポート	db.r5 DB インスタンスクラスを使用する PostgreSQL を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2018 年 12 月 19 日

Amazon RDS for PostgreSQL で制限されたパスワード管理をサポート	Amazon RDS for PostgreSQL では、`rds.restrict_password_commands` パラメータと `rds_password` ロールを使用して、ユーザーのパスワードおよびパスワードの有効期限の変更をだれが管理するかを制限できるようになりました。詳細については、「パスワード管理の制限」を参照してください。	2018 年 12 月 19 日
Amazon RDS for PostgreSQL で Amazon CloudWatch Logs へのデータベースログのアップロードをサポート	Amazon RDS for PostgreSQL は、CloudWatch Logs へのデータベースログのアップロードをサポートするようになりました。詳細については、「CloudWatch Logs への PostgreSQL ログの発行」を参照してください。	2018 年 12 月 10 日
Amazon RDS for Oracle で db.r5 DB インスタンスクラスをサポート	db.r5 DB インスタンスクラスを使用する Oracle を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2018 年 11 月 20 日
DB インスタンスの削除時にバックアップを保持	Amazon RDS は、DB インスタンスの削除時に自動バックアップの保持をサポートするようになりました。詳細については、「バックアップの使用」を参照してください。	2018 年 11 月 15 日

Amazon RDS for PostgreSQL で db.m5 DB インスタンスクラスをサポート	db.m5 DB インスタンスクラスを使用する PostgreSQL を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2018 年 11 月 15 日
Amazon RDS for Oracle で新しいメジャーバージョンをサポート	Oracle バージョン 12.2 を実行する Amazon RDS DB インスタンスを作成できるようになりました。	2018 年 11 月 13 日
Amazon RDS for SQL Server で Always On をサポート	Amazon RDS for SQL Server では Always On 可用性グループがサポートされています。詳細については、「Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2018 年 11 月 8 日
Amazon RDS for PostgreSQL で、カスタム DNS サーバーを使用したアウトバウンドネットワークアクセスをサポート	Amazon RDS for PostgreSQL は、カスタム DNS サーバーを使用したアウトバウンドネットワークアクセスをサポートするようになりました。詳細については、「アウトバウンドネットワークアクセスでのカスタム DNS サーバーの使用」を参照してください。	2018 年 11 月 8 日

Amazon RDS for MariaDB、MySQL、および PostgreSQL で 32 TiB のストレージをサポート	MySQL、MariaDB、および PostgreSQL 用の最大 32 TiB のストレージを備えた Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスストレージ」を参照してください。	2018 年 11 月 7 日
Amazon RDS for Oracle は、拡張データ型をサポートします	Oracle を実行する Amazon RDS DB インスタンスで拡張データ型を有効にできるようになりました。拡張データ型では、VARCHAR2、NVARCHAR2 および RAW データ型の最大サイズは 32,767 バイトです。詳細については、「拡張データ型を使用する」を参照してください。	2018 年 11 月 6 日
Amazon RDS for Oracle は db.m5 DB インスタンスクラスをサポートしています	db.m5 DB インスタンスクラスを使用する Oracle を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2018 年 11 月 2 日

Amazon RDS for Oracle の SE、SE1、または SE2 から EE への移行	任意の Oracle Database Standard Edition (SE、SE1、または SE2) から Oracle Database Enterprise Edition (EE) に移行できるようになりました。詳細については、「Oracle のエディション間での移行」を参照してください。	2018 年 10 月 31 日
Amazon RDS でマルチ AZ インスタンスを停止可能に	Amazon RDS は、マルチ AZ 配置の一部である DB インスタンスを停止できるようになりました。以前は、インスタンスの停止機能はマルチ AZ インスタンスに対して制限されていました。詳細については、「一時的に Amazon RDS DB インスタンスを停止する」を参照してください。	2018 年 10 月 29 日
Amazon RDS Performance Insights が Amazon RDS for Oracle で利用可能	Amazon RDS Performance Insights が Amazon RDS for Oracle で利用可能になりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2018 年 10 月 29 日

Amazon RDS for PostgreSQL が PostgreSQL バージョン 11 をデータベースプレビュー環境でサポート	Amazon RDS for PostgreSQL は、PostgreSQL バージョン 11 をデータベースプレビュー環境でサポートするようになりました。詳細については、「データベースプレビュー環境の Amazon RDS における PostgreSQL バージョン 11」を参照してください。	2018 年 10 月 25 日
MySQL での新しいメジャーバージョンのサポート	MySQL バージョン 8.0 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2018 年 10 月 23 日
MariaDB での新しいメジャーバージョンのサポート	MariaDB バージョン 10.3 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2018 年 10 月 23 日
Amazon RDS for Oracle での Oracle JVM のサポート	Amazon RDS for Oracle が Oracle Java Virtual Machine (JVM) オプションをサポートするようになりました。詳細については、「Oracle Java Virtual Machine」を参照してください。	2018 年 10 月 16 日

復元およびポイントインタイムリカバリのカスタムパラメータグループ	スナップショットを復元するとき、またはポイントインタイムリカバリ操作を実行するときに、カスタムパラメータグループを指定できるようになりました。詳細については、「DB スナップショットの復元」および「特定の時間への DB インスタンスの復元」を参照してください。	2018 年 10 月 15 日
Amazon RDS for Oracle での 32 TiB ストレージのサポート	最大 32 TiB のストレージを備えた Oracle RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスストレージ」を参照してください。	2018 年 10 月 15 日
Amazon RDS for MySQL での GTID のサポート	Amazon RDS for MySQL が、すべての DB インスタンスで一意の、レプリケーション設定内のグローバルトランザクション識別子 (GTID) をサポートするようになりました。詳細については、「RDS for MySQL で GTID ベースのレプリケーションを使用する」を参照してください。	2018 年 10 月 10 日
MySQL 5.7.23、5.6.41、および 5.5.61	MySQL バージョン 5.7.23、5.6.41、および 5.5.61 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2018 年 10 月 8 日

Amazon RDS for Oracle が SQLT の新しいバージョンをサポートするようになりました	Amazon RDS for Oracle が SQLT バージョン 12.2.180331 をサポートするようになりました。詳細については、「Oracle SQLT」を参照してください。	2018 年 10 月 4 日
Amazon RDS for PostgreSQL が IAM 認証をサポート	Amazon RDS for PostgreSQL が IAM 認証をサポートするようになりました。詳細については、「MySQL および PostgreSQL の IAM データベース認証」を参照してください。	2018 年 9 月 27 日
Amazon RDS DB インスタンスの削除保護を有効にできます	DB インスタンスの削除保護を有効にすると、どのユーザーもデータベースを削除できません。詳細については、「DB インスタンスの削除」を参照してください。	2018 年 9 月 26 日
Amazon RDS for MySQL と Amazon RDS for MariaDB で db.m5 DB インスタンスクラスをサポート	db.m5 DB インスタンスクラスを使用する MySQL または MariaDB を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2018 年 9 月 18 日

Amazon RDS が SQL Server 2017 へのアップグレードをサポート	SQL Server 2008 を除く任意のバージョンから既存の DB インスタンスを SQL Server 2017 にアップグレードできます。SQL Server 2008 からアップグレードするには、まず他のいずれかのバージョンにアップグレードしてください。詳細については、「Microsoft SQL Server DB エンジンのアップグレード」を参照してください。	2018 年 9 月 11 日
Amazon RDS for PostgreSQL が PostgreSQL バージョン 11 ベータ 3 をデータベースプレビュー環境でサポート	このリリースでは、ログ先行書き込み (WAL) セグメントサイズ (wal_segment_size) が 64 MB に設定されました。PostgreSQL バージョン 11 Beta 3 の詳細については、「PostgreSQL 11 Beta 3 Released」を参照してください。データベースプレビュー環境の詳細については、「データベースプレビュー環境の使用」を参照してください。	2018 年 9 月 7 日

Amazon Aurora ユーザーガイド	Amazon Aurora ユーザーガイドは、Amazon Aurora のすべての概念と、コンソールおよびコマンドラインインターフェイスの両方でのさまざまな機能の使用手順について説明します。Amazon RDS ユーザーガイドは、 Aurora 以外のデータベースエンジンについて説明するようになりました。	2018 年 8 月 31 日
Amazon RDS Performance Insights が RDS for MySQL で利用可能	Amazon RDS Performance Insights が RDS for MySQL で利用可能になりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2018 年 8 月 28 日
Aurora PostgreSQL 互換エディション が Aurora Auto Scaling をサポート	Aurora レプリカの Auto Scaling が Aurora PostgreSQL 互換エディションで利用可能になりました。 詳細については、「Aurora レプリカでの Amazon Aurora Auto Scaling の使用」を参照してください。	2018 年 8 月 16 日
Aurora ServerlessAurora MySQL 用の	Aurora Serverless は、Amazon Aurora 用のオンデマンドの Auto Scaling 設定です。詳細については、「Amazon Aurora Serverless の使用」を参照してください。	2018 年 8 月 9 日

MySQL 5.7.22、5.6.40	MySQL バージョン 5.7.22 および 5.6.40 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MySQL のバージョン」を参照してください。	2018 年 8 月 6 日
Aurora が中国 (寧夏) リージョンで利用可能に	Aurora MySQL と Aurora PostgreSQL が中国 (寧夏) リージョンで利用可能になりました。詳細については、「Amazon Aurora MySQL の可用性」および「Amazon Aurora PostgreSQL の可用性」を参照してください。	2018 年 8 月 6 日
Amazon RDS for MySQL での遅延レプリケーションのサポート	Amazon RDS for MySQL は、災害対策用の戦略として遅延レプリケーションをサポートするようになりました。詳細については、「MySQL での遅延レプリケーションの設定」を参照してください。	2018 年 8 月 6 日
Amazon RDS Performance Insights が Aurora MySQL で利用可能に	Amazon RDS Performance Insights が Aurora MySQL で利用可能になりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2018 年 8 月 6 日

Amazon RDS Performance Insights と Amazon CloudWatch の統合	Amazon RDS Performance Insights はメトリクスを自動的に Amazon CloudWatch に発行します。詳細については、「Performance Insights から CloudWatch に発行されるメトリクス」を参照してください。	2018 年 8 月 6 日
Amazon RDS 推奨事項	Amazon RDS は、データベースリソースに対して自動化された推奨事項を示すようになりました。詳細については、「Amazon RDS 推奨事項を使用する」を参照してください。	2018 年 7 月 25 日
AWS リージョン間での差分スナップショットコピー	Amazon RDS では、暗号化されていないインスタンスと暗号化されたインスタンスの両方に対して、AWS リージョン間での差分スナップショットコピーがサポートされています。詳細については、AWS リージョン間のスナップショットのコピーを参照してください。	2018 年 7 月 24 日
Amazon RDS for PostgreSQL で Amazon RDS Performance Insights が利用可能に	Amazon RDS for PostgreSQL で Amazon RDS Performance Insights が利用可能になりました。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2018 年 7 月 18 日

Amazon RDS for Oracle が Oracle APEX バージョン 5.1.4.v1 をサポート	Amazon RDS for Oracle が Oracle Application Express (APEX) バージョン 5.1.4.v1 をサポートするようになりました。詳細については、「Oracle Application Express」を参照してください。	2018 年 7 月 10 日
Amazon RDS for Oracle が Amazon CloudWatch Logs へのログの発行をサポート	Amazon RDS for Oracle が、CloudWatch Logs 内のロググループへのアラート、監査、トレース、リスナーログデータの発行をサポートするようになりました。詳細については、「Amazon CloudWatch Logs への Oracle ログの発行」を参照してください。	2018 年 7 月 9 日
MariaDB 10.2.15、10.1.34、10.0.35	MariaDB バージョン 10.2.15、10.1.34、および 10.0.35 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2018 年 7 月 5 日
Aurora PostgreSQL 1.2 が使用可能、PostgreSQL 9.6.8 と互換性があります	Aurora PostgreSQL 1.2 が使用可能になり、PostgreSQL 9.6.8 と互換性があります。詳細については、「バージョン 1.2」を参照してください。	2018 年 27 月 6 日

Amazon RDS PostgreSQL のリードレプリカがマルチ AZ 配置をサポート	Amazon RDS PostgreSQL の RDS リードレプリカが、複数のアベイラビリティーゾーンをサポートするようになりました。詳細については、「PostgreSQL リードレプリカの使用」を参照してください。	2018 年 25 月 6 日
Aurora PostgreSQL で Performance Insights が利用可能に	Aurora PostgreSQL で Performance Insightsが一般で利用できるようになりました。パフォーマンスデータの保持期間の延長もサポートされます。詳細については、「Amazon RDS Performance Insights の使用」を参照してください。	2018 年 6 月 21 日
Aurora PostgreSQL が米国西部 (北カリフォルニア) リージョンで使用可能に	Aurora PostgreSQL が米国西部 (北カリフォルニア) リージョンで使用可能になりました。アベイラビリティーゾーンの詳細については、「Amazon Aurora PostgreSQL の使用可否」を参照してください。	2018 年 6 月 11 日
Amazon RDS for Oracle が CPU 設定をサポート	Amazon RDS for Oracle で、DB インスタンスクラスのプロセッサの CPU コア数および各コアのスレッド数の設定がサポートされます。詳細については、「DB インスタンスクラスのプロセッサの設定」を参照してください。	2018 年 6 月 5 日

以前の更新

次の表に、2018 年 6 月以前の Amazon RDS ユーザーガイドの各リリースにおける重要な変更点を示します。

変更	説明	変更日
Amazon RDS for PostgreSQL は、PostgreSQL バージョン 11 ベータ 1 をデータベースプレビュー環境でサポートするようになりました	PostgreSQL バージョン 11 ベータ 1 には、「PostgreSQL 11 ベータ 1 がリリース」に記載されているいくつかの改善点が含まれています。 データベースプレビュー環境の詳細については、「データベースプレビュー環境の使用」を参照してください。	2018 年 5 月 31 日
Amazon RDS for Oracle が、TLS バージョン 1.0 および 1.2 をサポートしました。	Amazon RDS for Oracle は、Transport Layer Security (TLS) バージョン 1.0 および 1.2 をサポートしています。詳細については、「Oracle SSL オプションの TLS バージョン」を参照してください。	2018 年 5 月 30 日
Aurora MySQL が Amazon CloudWatch Logs へのログの発行をサポート	Aurora MySQL が、全般ログ、スローログ、監査ログ、およびエラーログデータの CloudWatch Logs のロググループへの発行をサポートしました。詳細については、「Aurora MySQL の CloudWatch Logs への発行」を参照してください。	2018 年 5 月 23 日
Amazon RDS PostgreSQL のデータベースプレビュー環境	Amazon RDS PostgreSQL の新しいインスタンスをプレビューモードで起動できるようになりました。データベースプレビュー環境の詳細については、「データベースプレビュー環境の使用」を参照してください。	2018 年 5 月 22 日
Amazon RDS for Oracle DB インスタンスで新しい	Oracle DB インスタンスが、db.x1e および db.x1 DB インスタンスクラスをサポートしました。詳細については、「 DB インスタンスクラス」および「RDS for	2018 年 5 月 22 日

変更	説明	変更日
DB インスタンスクラスをサポート	Oracle DB インスタンスクラス」を参照してください。	
Amazon RDS PostgreSQL が、リードレプリカで postgres_fdw をサポートするようになりました。	postgres_fdw を使用して、リードレプリカからリモートサーバーに接続できます。詳細については、「外部データへのアクセスのための postgres_fdw 拡張機能の使用」を参照してください。	2018 年 5 月 17 日
Amazon RDS for Oracle が sqlnet.ora パラメータの設定をサポート	Amazon RDS for Oracle が sqlnet.ora パラメータの設定をサポート 詳細については、「sqlnet.ora パラメータを使用した接続プロパティの変更」を参照してください。	2018 年 5 月 10 日
Aurora PostgreSQL が、アジアパシフィック (ソウル) リージョンで使用できるようになりました。	Aurora PostgreSQL が、アジアパシフィック (ソウル) リージョンで使用できるようになりました。アベイラビリティーゾーンの詳細については、「Amazon Aurora PostgreSQL の使用可否」を参照してください。	2018 年 5 月 9 日
Aurora MySQL がバックトラックをサポート	Aurora MySQL が、バックアップからデータを復元しないで、DB クラスターを特定の時刻に「巻き戻し」することができるようになりました。詳細については、「Aurora DB クラスターのバックトラック」を参照してください。	2018 年 5 月 9 日
Aurora MySQL は、外部の MySQL からの暗号化された移行およびレプリケーションをサポートしています。	Aurora MySQL は、外部の MySQL データベースからの暗号化された移行およびレプリケーションをサポートするようになりました。詳細については、「外部の MySQL データベースから Amazon Aurora MySQL DB クラスターへのデータ移行」と「Aurora と MySQL との間、または Aurora と別の Aurora DB クラスターとの間のレプリケーション」を参照してください。	2018 年 4 月 25 日

変更	説明	変更日
Aurora PostgreSQL 互換エディション はコピーオンライトプロトコルをサポートしています。	Aurora PostgreSQL データベースクラスターでデータベースのクローンができるようになりました。詳細については、「Aurora クラスターでのデータベースのクローン作成」を参照してください。	2018 年 4 月 10 日
MariaDB 10.2.12、10.1.31、10.0.34	MariaDB バージョン 10.2.12、10.1.31、10.0.34 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2018 年 3 月 21 日
新しいリージョン向けの Aurora PostgreSQL のサポート	Aurora PostgreSQL が欧州 (ロンドン) とアジアパシフィック (シンガポール) の各リージョンで利用可能になりました。アベイラビリティーゾーンの詳細については、「Amazon Aurora PostgreSQL の使用可否」を参照してください。	2018 年 3 月 13 日
MySQL 5.7.21、5.6.39、および 5.5.59	MySQL バージョン 5.7.21、5.6.39、5.5.59 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での MySQL のバージョン」を参照してください。	2018 年 3 月 9 日
Amazon RDS for Oracle で Oracle REST Data Services のサポートをスタート	Amazon RDS for Oracle は、APEX オプションの一部として Oracle REST Data Services をサポートしています。詳細については、「Oracle Application Express (APEX)」を参照してください。	2018 年 3 月 9 日
Amazon Aurora MySQL 互換エディションが新しい AWS リージョンで利用可能に	Aurora MySQL がアジアパシフィック (シンガポール) リージョンで使用できるようになりました。Aurora MySQL の AWS リージョンの詳細なリストについては、「Amazon Aurora MySQL の使用可否」を参照してください。	2018 年 3 月 6 日

変更	説明	変更日
Microsoft SQL Server を実行している Amazon RDS DB インスタンスの変更データキャプチャ (CDC) のサポート	Amazon RDS for Microsoft SQL Server を実行している DB インスタンスが、変更データキャプチャ (CDC) をサポートしました。詳細については、「Microsoft SQL Server DB インスタンスの変更データキャプチャのサポート」を参照してください。	2018 年 2 月 6 日
Aurora MySQL での新しいメジャーバージョンのサポート	MySQL バージョン 5.7 を実行する Aurora MySQL DB クラスターを作成できるようになりました。詳細については、「Amazon Aurora MySQL Database Engine Updates 2018-02-06」を参照してください。	2018 年 2 月 6 日
MySQL および MariaDB ログを Amazon CloudWatch Logs に発行する	MySQL および MariaDB ログデータを CloudWatch Logs に発行できるようになりました。詳細については、「Amazon CloudWatch Logs への MySQL ログの発行」および「MariaDB ログを Amazon CloudWatch Logs に発行する」を参照してください。	2018 年 1 月 17 日
リードレプリカのマルチ AZ のサポート	リードレプリカをマルチ AZ DB インスタンスとして作成できるようになりました。Amazon RDS では、レプリカのフェイルオーバーをサポートするため、別のアベイラビリティーゾーンにレプリカのスタンバイを作成します。リードレプリカは、ソースのデータベースがマルチ AZ DB インスタンスであるかどうかに関係なく、マルチ AZ DB インスタンスとして作成できます。詳細については、「DB インスタンスのリードレプリカの操作」を参照してください。	2018 年 1 月 11 日
Amazon RDS for MariaDB での新しいメジャーバージョンのサポート	MariaDB バージョン 10.2 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での MariaDB 10.2 のサポート」を参照してください。	2018 年 1 月 3 日

変更	説明	変更日
Amazon Aurora PostgreSQL 互換エディション が新しい AWS リージョンで利用可能に	Aurora PostgreSQL が欧州 (パリ) リージョンで利用可能になりました。Aurora PostgreSQL の AWS リージョンの詳細なリストについては、「Amazon Aurora PostgreSQL の使用可否」を参照してください。	2017 年 22 月 12 日
Aurora PostgreSQL では、インスタンスタイプがサポートされます。	Aurora PostgreSQL で新しいインスタンスタイプがサポートされるようになりました。インスタンスタイプの詳細なリストについては、「 DB インスタンスクラスの選択」を参照してください。	2017 年 20 月 12 日
Amazon Aurora MySQL 互換エディション が新しい AWS リージョンで利用可能に	Aurora MySQL が欧州 (パリ) リージョンで利用可能になりました。Aurora MySQL の AWS リージョンの詳細なリストについては、「Amazon Aurora MySQL の使用可否」を参照してください。	2017 年 18 月 12 日
Aurora MySQL でハッシュ結合をサポート	等価結合を使用して大量のデータを結合する必要がある場合は、この機能によりクエリのパフォーマンスが向上することがあります。詳細については、「Aurora MySQL でのハッシュ結合の使用」を参照してください。	2017 年 12 月 11 日
Aurora MySQL で AWS Lambda 関数を呼び出すネイティブ関数をサポート	Aurora MySQL を使用すると、ネイティブ関数 `lambda_sync` と `lambda_async` を呼び出すことができます。詳細については、「Amazon Aurora MySQL DB クラスターから Lambda 関数を呼び出す」を参照してください。	2017 年 12 月 11 日
Aurora PostgreSQL HIPAA 適格性が追加されました。	Aurora PostgreSQL は現在、HIPAA 準拠のアプリケーションの構築をサポートしています。詳細については、「Amazon Aurora PostgreSQL の使用」を参照してください。	2017 年 6 月 12 日

変更	説明	変更日
PostgreSQL との互換性を備えた Amazon Aurora がさらなる AWS リージョンで利用可能に	PostgreSQL との互換性を備えた Amazon Aurora は 4 か所の AWS リージョンで新たに利用可能になりました。アベイラビリティーゾーンの詳細については、「Amazon Aurora PostgreSQL の使用可否」を参照してください。	2017 年 11 月 22 日
Microsoft SQL Server を実行している Amazon RDS DB インスタンスのストレージを変更	SQL Server を実行している Amazon RDS DB インスタンスのストレージを変更できるようになりました。詳細については、「Amazon RDS DB インスタンスを変更する」を参照してください。	2017 年 11 月 21 日
Amazon RDS で Linux ベースのエンジン用に 16 TiB ストレージをサポート	最大 16 TiB のストレージを備えた MySQL、MariaDB、PostgreSQL、および Oracle RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。	2017 年 11 月 21 日
Amazon RDS でストレージの高速スケールアップをサポート	MySQL、MariaDB、PostgreSQL、および Oracle RDS DB インスタンスにストレージを数分で追加できるようになりました。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。	2017 年 11 月 21 日
Amazon RDS で MariaDB バージョン 10.1.26 と 10.0.32 をサポート	MariaDB バージョン 10.1.26 および 10.0.32 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2017 年 11 月 20 日
Amazon RDS for Microsoft SQL Server で新しい DB インスタンスクラスをサポート	SQL Server を実行する Amazon RDS DB インスタンスを db.r4 および db.m4.16xlarge DB インスタンスクラスで作成できるようになりました。詳細については、「Microsoft SQL Server の DB インスタンスクラスのサポート」を参照してください。	2017 年 11 月 20 日

変更	説明	変更日
Amazon RDS for MySQL および Amazon RDS for MariaDB で新しい DB インスタンスクラスをサポート	MySQL および MariaDB を実行する Amazon RDS DB インスタンスを db.r4、db.m4.16xlarge、db.t2.xlarge、および db.t2.2xlarge DB インスタンスクラスで作成できるようになりました。詳細については、「DB インスタンスクラス」を参照してください。	2017 年 11 月 20 日
SQL Server 2017	Microsoft SQL Server 2017 を実行する Amazon RDS DB インスタンスを作成できるようになりました。SQL Server 2016 SP1 CU5 を実行する DB インスタンスを作成することもできます。詳細については、「Amazon RDS for Microsoft SQL Server」を参照してください。	2017 年 11 月 17 日
Amazon S3 から MySQL バックアップを復元	オンプレミスデータベースのバックアップを作成して Amazon S3 に保存し、MySQL を実行する新しい Amazon RDS DB インスタンスにバックアップファイルを復元できます。詳細については、「Amazon RDS for MySQL DB インスタンスへのバックアップの復元」を参照してください。	2017 年 11 月 17 日
Aurora レプリカによる Auto Scaling	Amazon Aurora MySQL が Aurora Auto Scaling をサポートするようになりました。Aurora Auto Scaling によって、接続やワークロードの増減に基づいて、Aurora レプリカの数が動的に調整されます。詳細については、「Aurora レプリカでの Amazon Aurora Auto Scaling の使用」を参照してください。	2017 年 11 月 17 日
Oracle デフォルトエディションのサポート	Amazon RDS for Oracle DB インスタンスでは、DB インスタンスのデフォルトエディションの設定がサポートされるようになりました。詳細については、「DB インスタンスのデフォルトエディションの設定」を参照してください。	2017 年 11 月 3 日

変更	説明	変更日
Oracle DB インスタンスファイルの確認	Amazon RDS for Oracle DB インスタンスでは、Oracle Recovery Manager (RMAN) 論理的な検証ユーティリティを使用した DB インスタンスファイルの検証がサポートされるようになりました。詳細については、「RDS for Oracle でのデータベースファイルの検証」を参照してください。	2017 年 11 月 3 日
OEM 13c の Management Agent	Amazon RDS for Oracle DB インスタンスで、Oracle Enterprise Manager (OEM) Cloud Control 13c の Management Agent のサポートがスタートされました。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。	2017 年 11 月 1 日
Microsoft SQL Server スナップショットのストレージ再設定	Microsoft SQL Server を実行中の Amazon RDS DB インスタンスにスナップショットを復元するときに、ストレージを再設定できるようになりました。詳細については、「DB インスタンスへの復元」を参照してください。	2017 年 10 月 26 日
Aurora MySQL 互換エディション の非同期キーのプリフェッチ	Asynchronous Key Prefetch (AKP) により、必要になる前にメモリ内でキーをプリフェッチすることで、非キャッシュのインデックス結合のパフォーマンスが向上します。詳細については、「Amazon Aurora での Asynchronous Key Prefetch の使用 」を参照してください。	2017 年 10 月 26 日
MySQL 5.7.19、5.6.37、および 5.5.57	MySQL バージョン 5.7.19、5.6.37、5.5.57 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS での MySQL のバージョン」を参照してください。	2017 年 10 月 25 日

変更	説明	変更日
PostgreSQL と互換性を持つ Amazon Aurora の一般提供スタート	PostgreSQL と互換性のある Amazon Aurora では、新規および既存の PostgreSQL のデプロイを簡単に、コスト効率よく設定、操作、スケーリングできるため、ユーザーは業務やアプリケーションに専念できます。詳細については、「Amazon Aurora PostgreSQL の使用」を参照してください。	2017 年 10 月 24 日
Amazon RDS for Oracle DB インスタンスで新しい DB インスタンスクラスをサポート	Amazon RDS for Oracle DB インスタンスでは、次世代のメモリ最適化 (db.r4) インスタンスクラスのサポートをスタートしました。また Amazon RDS for Oracle DB インスタンスでは、新しい現行世代インスタンスクラス (db.m4.16xlarge、db.t2.xlarge、db.t2.2xlarge) のサポートもスタートしました。詳細については、「 DB インスタンスクラス」および「RDS for Oracle DB インスタンスクラス」を参照してください。	2017 年 10 月 23 日
新機能	新規および既存のリザーブドインスタンスで、同じ DB インスタンスクラスの複数のサイズに対応できるようになりました。サイズに柔軟性のあるリザーブドインスタンスは、同じ AWS リージョン、データベースエンジン、およびインスタンスファミリーで、AZ 設定にまたがって DB インスタンスで利用できます。サイズに柔軟性のあるリザーブドインスタンスは、Amazon Aurora、MariaDB、MySQL、Oracle (Bring-Your-Own-License)、PostgreSQL の各データベースエンジンで利用できます。詳細については、「サイズ柔軟なリザーブド DB インスタンス」を参照してください。	2017 年 10 月 11 日
新機能	Oracle SQLT オプションを使用して、最適なパフォーマンスを得るために SQL ステートメントを調整できるようになりました。詳細については、「Oracle SQLT」を参照してください。	2017 年 9 月 22 日

変更	説明	変更日
新機能	Amazon RDS for Oracle DB インスタンスの既存の手動 DB スナップショットがある場合は、Oracle データベースエンジンの新しいバージョンにアップグレードできます。詳細については、「Oracle DB スナップショットのアップグレード」を参照してください。	2017 年 9 月 20 日
新機能	これで、Oracle を実行している Amazon RDS DB インスタンスで Oracle Spatial を使用して、spatial データを保存、取得、更新、クエリできます。詳細については、「Oracle Spatial」を参照してください。	2017 年 9 月 15 日
新機能	これで、Oracle Locator を使用して、Oracle を実行している Amazon RDS DB インスタンスでインターネットおよびサービスベースのアプリケーションとパートナーベースの GIS ソリューションをサポートできるようになります。詳細については、「Oracle Locator」を参照してください。	2017 年 9 月 15 日
新機能	これで、Oracle マルチメディアを使用して、Oracle を実行している Amazon RDS DB インスタンスでイメージ、音声、動画やそのほかのさまざまなメディアデータを保管、管理、取得することができるようになります。	2017 年 9 月 15 日
新機能	これで、Amazon Aurora MySQL DB クラスターから Amazon CloudWatch Logs に監査ログをエクスポートできるようになります。詳細については、「Aurora MySQL ログの Amazon CloudWatch Logs への発行」を参照してください。	2017 年 9 月 14 日
新機能	これで、Amazon RDS は Oracle を実行している DB インスタンスで Oracle Application Express (APEX) の複数のバージョンをサポートするようになります。詳細については、「Oracle Application Express (APEX)」を参照してください。	2017 年 9 月 13 日

変更	説明	変更日
新機能	これで、Amazon Aurora を使用して、非暗号化または暗号化された DB スナップショット、あるいは MySQL DB インスタンスを暗号化された Aurora MySQL DB クラスターに移行できるようになります。詳細については、「RDS for MySQL スナップショットの Aurora への移行」と「Aurora リードレプリカを使用した MySQL DB インスタンスから Amazon Aurora MySQL DB クラスターへのデータの移行」を参照してください。	2017 年 9 月 5 日
新機能	Amazon RDS for Microsoft SQL Server データベースを使用して、HIPAA 準拠アプリケーションを構築できます。詳細については、「Microsoft SQL Server DB インスタンス用のコンプライアンスプログラムサポート」を参照してください。	2017 年 8 月 31 日
新機能	Amazon RDS for MariaDB データベースを使用して、HIPAA 準拠アプリケーションを構築できるようになりました。詳細については、「Amazon RDS for MariaDB」を参照してください。	2017 年 8 月 31 日
新機能	16 TiB までの割り当てられたストレージ、および 1:1-50:1 のストレージ範囲のプロビジョンド IOPS で、Microsoft SQL Server を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS DB インスタンスストレージ」を参照してください。	2017 年 8 月 22 日
新機能	EU (フランクフルト) リージョンで Microsoft SQL Server を実行する DB インスタンス用のマルチ AZ 配置を使用できるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2017 年 8 月 3 日

変更	説明	変更日
新機能	MariaDB バージョン 10.1.23 および 10.0.31 を実行する Amazon RDS DB インスタンスを作成できるようになりました。詳細については、「Amazon RDS の MariaDB のバージョン」を参照してください。	2017 年 7 月 17 日
新機能	Amazon RDS は、すべての AWS リージョンで Microsoft SQL Server Enterprise Edition のライセンス込みのモデルをサポートするようになりました。詳細については、「Amazon RDS での Microsoft SQL Server のライセンス」を参照してください。	2017 年 7 月 13 日
新機能	Amazon RDS for Oracle は、データベースの拡張性を増大する Linux Kernel の huge pages をサポートするようになりました。Huge pages を使用すると、ページのテーブルを小さくし、メモリ管理の CPU 経過時間を減少することで、大規模なデータベースインスタンスのパフォーマンスを向上できます。Oracle バージョン 12.1.0.2 と 11.2.0.4 のすべての工ディションを実行する Amazon RDS DB インスタンスで huge pages を使用できます。詳細については、「サポートされている RDS for Oracle インスタンスで HugePages をオンにする」を参照してください。	2017 年 7 月 7 日
新機能	非 Aurora DB エンジンの db.t2.small と db.t2.medium DB インスタンスクラスの保存時暗号化 (EAR) をサポートするように更新しました。詳細については、「Amazon RDS の暗号化の可用性」を参照してください。	2017 年 6 月 27 日
新機能	欧州 (フランクフルト) リージョンで Amazon Aurora をサポートするように更新されました。アベイラビリティーゾーンの詳細については、「Amazon Aurora MySQL の使用可否」を参照してください。	2017 年 6 月 16 日

変更	説明	変更日
新機能	AWS リージョン間で DB スナップショットをコピーするときに、オプショングループを指定できるようになりました。詳細については、「オプショングループの考慮事項」を参照してください。	2017 年 6 月 12 日
新機能	AWS リージョン間で、特化された DB インスタンスから作成された DB スナップショットをコピーできるようになりました。Oracle TDE、Microsoft SQL Server TDE、および Microsoft SQL Server のミラーリングを使用したマルチ AZ 配置を使用する DB インスタンスからスナップショットをコピーできます。詳細については、「DB スナップショットのコピー」を参照してください。	2017 年 6 月 12 日
新機能	Amazon Aurora で、Amazon Aurora DB クラスターのすべてのデータベースを迅速にコスト効率よくコピーできるようになりました。詳細については、「Aurora クラスターでのデータベースのクローン作成」を参照してください。	2017 年 6 月 12 日
新機能	Amazon RDS で Microsoft SQL Server 2016 SP1 CU2 がサポートされるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server」を参照してください。	2017 年 6 月 7 日
プレビュー	PostgreSQL と互換性を持つ Amazon Aurora の発行プレビュー 詳細については、「Amazon Aurora PostgreSQL の使用」を参照してください。	2017 年 4 月 19 日

変更	説明	変更日
新機能	Amazon Aurora により、ALTER TABLE tbl_name ADD COLUMN col_name column_definition オペレーションをほぼ即時に実行できるようになりました。このオペレーションでは、テーブルをコピーする必要はありません。他の DML ステートメントに実質的な影響を及ぼすこともありません。詳細については、「高速 DDL を使用して Amazon Aurora でテーブルを変更する」を参照してください。	2017 年 4 月 5 日
新機能	新しいモニタリングコマンド、SHOW VOLUME STATUS が追加されました。ボリューム内のノードやディスクの数を表示するには、このコマンドを使用します。詳細については、「Aurora DB クラスターのボリュームステータスの表示」を参照してください。	2017 年 4 月 5 日
新機能	Amazon RDS での Oracle のカスタムパスワード検証関数において、独自のカスタムロジックを使用できるようになりました。詳細については、「パスワードを検証するためのカスタム関数の作成」を参照してください。	2017 年 3 月 21 日
新機能	Amazon RDS の Oracle DB インスタンスのオンライン REDO ログファイルやアーカイブ REDO ログファイルにアクセスできるようになりました。詳細については、「オンライン およびアーカイブ REDO ログへのアクセス」を参照してください。	2017 年 3 月 21 日
新機能	同じリージョンのアカウント間で、暗号化されている DB クラスタースナップショットと暗号化されていない DB クラスタースナップショットの両方をコピーできるようになりました。詳細については、「アカウント間での DB クラスタースナップショットのコピー」を参照してください。	2017 年 3 月 7 日

変更	説明	変更日
新機能	同じリージョンのアカウント間で、暗号化されている DB クラススナップショットを共有できるようになりました。詳細については、「DB クラスターのスナップショットの共有」を参照してください。	2017 年 3 月 7 日
新機能	暗号化されている Amazon Aurora MySQL DB クラスターをレプリケートして、クロスリージョン Aurora レプリカを作成できるようになりました。詳細については、「AWS リージョン間での Aurora MySQL DB クラスターのレプリケート」を参照してください。	2017 年 3 月 7 日
新機能	Microsoft SQL Server を実行する DB インスタンスへのすべての接続で Secure Sockets Layer (SSL) の使用を要求できるようになりました。詳細については、「Microsoft SQL Server DB インスタンスでの SSL の使用」を参照してください。	2017 年 2 月 27 日
新機能	ローカルのタイムゾーンを 15 の追加のタイムゾーンのいずれかに設定できるようになりました。詳細については、「サポートされているタイムゾーン」を参照してください。	2017 年 2 月 27 日
新機能	Amazon RDS プロシージャ `msdb.dbo.rds_shrink_tempdbfile` を使用して、Microsoft SQL Server を実行している DB インスタンスの tempdb データベースを圧縮できるようになりました。詳細については、「tempdb データベースの圧縮」を参照してください。	2017 年 2 月 17 日
新機能	Enterprise および Standard Edition の Microsoft SQL Server データベースを Amazon RDS DB インスタンスから Amazon S3 にエクスポートするとき、バックアップファイルを圧縮できるようになりました。詳細については、「バックアップファイルの圧縮」を参照してください。	2017 年 2 月 17 日

変更	説明	変更日
新機能	Oracle を実行している DB インスタンスでアウトバウンドネットワークアクセスに使われる DNS 名を解決できるようにするため、Amazon RDS がカスタム DNS サーバーをサポートするようになりました。詳細については、「カスタム DNS サーバーのセットアップ」を参照してください。	2017 年 1 月 26 日
新機能	Amazon RDS は、別のリージョンで暗号化されたリードレプリカを作成するようになりました。詳細については、「別の でのリードレプリカの作成AWS リージョン」および「CreateDBInstanceReadReplica」を参照してください。	2017 年 1 月 23 日
新機能	Amazon RDS が MySQL 5.1 から MySQL 5.5 への MySQL DB スナップショットアップグレードをサポートするようになりました。	2017 年 1 月 20 日
新機能	Amazon RDS は、MariaDB、MySQL、Oracle、PostgreSQL、Microsoft SQL Server データベースエンジンで、別のリージョンへの暗号化された DB スナップショットのコピーをサポートするようになりました。詳細については、DB スナップショットのコピー および CopyDBSnapshot を参照してください。	2016 年 12 月 20 日
新機能	Amazon Aurora MySQL で空間インデックスをサポートするようになりました。 空間インデックスでは、空間的データを使用するクエリにおける大きなデータセットのクエリパフォーマンスが向上します。詳細については、「Amazon Aurora MySQL と空間データ」を参照してください。	2016 年 14 月 12 日

変更	説明	変更日
新機能	Amazon RDS は、Oracle を実行している DB インスタンスでのアウトバウンドのネットワークアクセスをサポートするようになりました。DB インスタンスからネットワークへの接続に utl_http、utl_tcp および utl_smtp を使用できます。詳細については、「証明書と Oracle ウォレットを使用した、UTL_HTTP アクセスの設定」を参照してください。	2016 年 5 月 12 日
新機能	Amazon RDS は、MySQL バージョン 5.1 のサポートを終了しました。ただし、既存の MySQL 5.1 スナップショットは MySQL 5.5 インスタンスに復元できます。詳細については、「RDS for MySQL のサポートされているストレージエンジン」を参照してください。	2016 年 11 月 15 日
新機能	Amazon RDS で Microsoft SQL Server 2016 RTM CU2 がサポートされるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server」を参照してください。	2016 年 11 月 4 日
新機能	Amazon RDS が Oracle を実行している DB インスタンスのメジャーバージョンアップグレードをサポートしました。Oracle DB インスタンスを 11g から 12c にアップグレードできるようになりました。詳細については、「RDS for Oracle DB エンジンのアップグレード」を参照してください。	2016 年 11 月 2 日
新機能	Microsoft SQL Server 2014 Enterprise Edition を実行する DB インスタンスを作成できるようになりました。Amazon RDS では、すべてのエディションおよびリージョンで SQL Server 2014 SP2 がサポートされるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server」を参照してください。	2016 年 10 月 25 日

変更	説明	変更日
新機能	Amazon Aurora MySQL は 他の AWS のサービスと統合され、Amazon S3 バケットからテーブル内にテキストや XML データをロードしたり、データベースコードから AWS Lambda 関数を呼び出しできるようになりました。詳細については、「Aurora MySQL と AWS の他のサービスとの統合」を参照してください。	2016 年 10 月 18 日
新機能	Microsoft SQL Server を実行している Amazon RDS DB インスタンスで tempdb データベースにアクセスできるようになりました。tempdb データベースにアクセスするには、Microsoft SQL Server Management Studio (SSMS) 経由で Transact-SQL を使用するか、他のスタンダードの SQL クライアントアプリケーションを使用します。詳細については、「Amazon RDS で実行している Microsoft SQL Server DB インスタンスの tempdb データベースへのアクセス」を参照してください。	2016 年 9 月 29 日
新機能	Oracle で実行している Amazon RDS DB インスタンスで UTL_MAIL パッケージを使用できるようになりました。詳細については、「Oracle UTL_MAIL」を参照してください。	2016 年 9 月 20 日
新機能	新しい Microsoft SQL Server DB インスタンスのタイムゾーンを、アプリケーションのタイムゾーンに合わせて設定できるようになりました。詳細については、「Microsoft SQL Server DB インスタンスのローカルタイムゾーン」を参照してください。	2016 年 9 月 19 日

変更	説明	変更日
新機能	Oracle Label Security オプションを使用して、Oracle Database 12c を実行している Amazon RDS DB インスタンス内の個々のテーブルの行へのアクセスを制御できるようになりました。Oracle Label Security を使用すると、ポリシーベースの管理モデルへの規制コンプライアンスを適用し、機密データへのアクセスが適切なクリアランスレベルを持つユーザーのみに制限されていることを確認できます。詳細については、「Oracle Label Security」を参照してください。	2016 年 9 月 8 日
新機能	リーダーエンドポイントを使用して、Amazon Aurora DB クラスターに接続できるようになりました。これにより、DB クラスターで使用可能な Aurora レプリカ全体にわたる接続の負荷分散を行うことができます。クライアントがリーダーエンドポイントへの新規接続をリクエストすると、Aurora によって接続リクエストが DB クラスターの Aurora レプリカ間で配信されます。この機能は、DB クラスターの複数の Aurora レプリカ間の読み取りワークロードを分散させる役に立ちます。詳細については、「Amazon Aurora エンドポイント」を参照してください。	2016 年 9 月 8 日
新機能	Oracle を実行している Amazon RDS DB インスタンスの Oracle Enterprise Manager Cloud Control をサポートします。DB インスタンスの Management Agent を有効化して、Oracle Management Service (OMS) でデータを共有できます。詳細については、「Enterprise Manager Cloud Control 向け Oracle Management Agent」を参照してください。	2016 年 9 月 1 日
新機能	このリリースでは、リソースの ARN を取得するサポートを追加します。詳細については、「Amazon RDS の既存の ARN の取得」を参照してください。	2016 年 8 月 23 日

変更	説明	変更日
新機能	リソースを管理し、コストを追跡するため、各 Amazon RDS リソースに、最大 50 個のタグを割り当てることができます。詳細については、「Amazon RDS リソースのタグ付け」を参照してください。	2016 年 8 月 19 日
新機能	Amazon RDS では、Oracle Standard Edition Two の License Included モデルがサポートされるようになりました。詳細については、「Amazon RDS DB インスタンスの作成」を参照してください。 Microsoft SQL Server および Oracle を実行している Amazon RDS DB インスタンスのライセンスモデルを変更できるようになりました。詳細については、「Amazon RDS での Microsoft SQL Server のライセンス」および「RDS for Oracle のライセンスオプション」を参照してください。	2016 年 8 月 5 日
新機能	Amazon RDS では、完全バックアップファイル (.bak ファイル) を使用した Microsoft SQL Server データベースのネイティブ バックアップおよび復元がサポートされるようになりました。ストレージに Amazon S3、暗号化に AWS KMS を使用することで、SQL Server データベースを Amazon RDS に簡単に移行できるようになりました。また、移動可能な 1 つのファイル内でデータベースのインポートとエクスポートが行いやすくなりました。詳細については、「ネイティブバックアップと復元を使用した SQL Server データベースのインポートとエクスポート」を参照してください。	2016 年 7 月 27 日

変更	説明	変更日
新機能	これで、MySQL データベースから Amazon Simple Storage Service (Amazon S3) バケットにソースファイルをコピーし、これらのファイルから Amazon Aurora DB クラスターを復元できます。このオプションは、`mysqldump` を使用したデータの移行よりもかなり高速になる場合があります。詳細については、「外部の MySQL データベースから Aurora MySQL DB クラスターへのデータ移行」を参照してください。	2016 年 7 月 20 日
新機能	復元オペレーション中に AWS Key Management Service (AWS KMS) 暗号化キーを含めることにより、暗号化されていない Amazon Aurora DB クラスタースナップショットを復元して、暗号化された Amazon Aurora DB クラスターを作成できるようになりました。詳細については、「Amazon RDS リソースの暗号化」を参照してください。	2016 年 6 月 30 日
新機能	Oracle Repository Creation Utility (RCU) を使用して、Amazon RDS for Oracle にリポジトリを作成できます。詳細については、「Amazon RDS for Oracle での Oracle リポジトリ作成ユーティリティの使用」を参照してください。	2016 年 6 月 17 日
新機能	PostgreSQL クロスリージョンリードレプリカのサポートを追加します。詳細については、「別の でのリードレプリカの作成AWS リージョン」を参照してください。	2016 年 6 月 16 日
新機能	AWS マネジメントコンソール を使用して、ミラーリングによるマルチ AZ を簡単に Microsoft SQL Server DB インスタンスに追加できるようになりました。詳細については、「Microsoft SQL Server DB インスタンスへのマルチ AZ の追加」を参照してください。	2016 年 6 月 9 日

変更	説明	変更日
新機能	アジアパシフィック (シドニー)、アジアパシフィック (東京)、南米 (サンパウロ) の追加リージョンで SQL Server のミラーリングを使用したマルチ AZ 配置を使用できるようになりました。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2016 年 6 月 9 日
新機能	MariaDB バージョン 10.1 をサポートするために更新されました。詳細については、「Amazon RDS for MariaDB」を参照してください。	2016 年 6 月 1 日
新機能	リードレプリカとなる Amazon Aurora クロスリージョン DB クラスターをサポートするために更新されました。詳細については、「AWS リージョン間での Aurora MySQL DB クラスターのレプリケート」を参照してください。	2016 年 6 月 1 日
新機能	Oracle DB インスタンスで拡張モニタリングが利用できるようになりました。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」および「Amazon RDS DB インスタンスを変更する」を参照してください。	2016 年 5 月 27 日
新機能	Amazon Aurora DB クラススナップショットの手動スナップショット共有をサポートするために更新されました。詳細については、「DB クラスターのスナップショットの共有」を参照してください。	2016 年 5 月 18 日
新機能	MariaDB 監査プラグインを使用して MariaDB および MySQL データベースインスタンスのデータベースアクティビティを記録できるようになりました。詳細については、「MariaDB データベースエンジンのオプション」および「MySQL DB インスタンスのオプション」を参照してください。	2016 年 4 月 27 日

変更	説明	変更日
新機能	MySQL バージョン 5.6 からバージョン 5.7 へのアップグレードで、インプレースでのメジャーバージョンアップグレードが可能になりました。詳細については、「RDS for MySQL DB エンジンのアップグレード」を参照してください。	2016 年 4 月 26 日
新機能	Microsoft SQL Server DB インスタンスで拡張モニタリングが利用できるようになりました。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。	2016 年 4 月 22 日
新機能	Amazon RDS コンソールに Amazon Aurora Clusters ビューが表示されるよう更新されました。詳細については、「Aurora DB クラスターを表示する」を参照してください。	2016 年 4 月 1 日
新機能	アジアパシフィック (ソウル) リージョンで SQL Server マルチAZ がサポートされるよう更新されました。詳細については、「Amazon RDS for Microsoft SQL Server のマルチ AZ 配置」を参照してください。	2016 年 3 月 31 日
新機能	アジアパシフィック (ソウル) リージョンで Amazon Aurora マルチAZ がサポートされるよう更新されました。アベイラビリティーゾーンの詳細については、「Amazon Aurora MySQL の使用可否」を参照してください。	2016 年 3 月 31 日
新機能	PostgreSQL DB インスタンスで、接続に SSL を使用することを要求できます。詳細については、「PostgreSQL DB インスタンスで SSL を使用する」を参照してください。	2016 年 3 月 25 日
新機能	PostgreSQL の DB インスタンスで拡張モニタリングが利用できるようになりました。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。	2016 年 3 月 25 日

変更	説明	変更日
新機能	Microsoft SQL Server DB インスタンスで、ユーザー認証のために Windows 認証を使用できるようになりました。詳細については、「RDS for SQL Server による AWS Managed Active Directory の操作」を参照してください。	2016 年 3 月 23 日
新機能	アジアパシフィック (ソウル) リージョンで拡張モニタリングが使用できるようになりました。詳細については、「拡張モニタリングを使用した OS メトリクスのモニタリング」を参照してください。	2016 年 3 月 16 日
新機能	フェイルオーバー中にプライマリインスタンスに Aurora レプリカを昇格する順序をカスタマイズできるようになりました。詳細については、「Aurora DB クラスターの耐障害性」を参照してください。	2016 年 3 月 14 日
新機能	Aurora DB クラスターに移行するときに暗号化をサポートするように更新されました。詳細については、「Aurora DB クラスターへのデータの移行」を参照してください。	2016 年 3 月 2 日
新機能	Aurora DB 用クラスターのローカルタイムゾーンをサポートするように更新されました。詳細については、「Aurora DB クラスターのローカルタイムゾーン」を参照してください。	2016 年 3 月 1 日
新機能	現行世代の Amazon RDS DB インスタンスクラス用に MySQL バージョン 5.7 のサポートを追加するように更新されました。	2016 年 2 月 22 日
新機能	AWS GovCloud (US-West) リージョンで、db.r3 および db.t2 DB インスタンスクラスをサポートするように更新されました。	2016 年 2 月 11 日

変更	説明	変更日
新機能	DB スナップショットのコピーの暗号化および暗号化された DB スナップショットの共有をサポートするように更新されました。詳細については、「Amazon RDS の DB スナップショットのコピー」および「Amazon RDS の DB スナップショットを共有する」を参照してください。	2016 年 2 月 11 日
新機能	アジアパシフィック (シドニー) リージョンで Amazon Aurora をサポートするように更新されました。アベイラビリティーゾーンの詳細については、「Amazon Aurora MySQL の使用可否」を参照してください。	2016 年 2 月 11 日
新機能	Oracle DB インスタンスで SSL をサポートするように更新されました。詳細については、「RDS for Oracle DB インスタンスでの SSL の使用」を参照してください。	2016 年 2 月 9 日
新機能	MySQL および MariaDB DB インスタンスのローカルタイムゾーンをサポートするように更新されました。詳細については、「MySQL DB インスタンスのローカルタイムゾーン」および「MariaDB DB インスタンスのローカルタイムゾーン」を参照してください。	2015 年 12 月 21 日
新機能	MySQL および MariaDB インスタンスと Aurora の DB クラスターの OS メトリクスの拡張モニタリングをサポートするように更新されました。詳細については、「Amazon RDS コンソールでのメトリクスの表示」を参照してください。	2015 年 12 月 18 日
新機能	MySQL バージョン 5.5 の db.t2、db.r3、および db.m4 DB インスタンスクラスをサポートするように更新されました。詳細については、「 DB インスタンスクラス」を参照してください。	2015 年 12 月 4 日
新機能	既存の DB インスタンスのデータベースポートの変更をサポートするように更新されました。	2015 年 12 月 3 日

変更	説明	変更日
新機能	PostgreSQL インスタンスのデータベースエンジンの、メジャーバージョンのアップグレードをサポートするように変更されました。詳細については、「RDS for PostgreSQL DB エンジンのアップグレード」を参照してください。	2015 年 11 月 19 日
新機能	既存の DB インスタンスのパブリックアクセス可能性に関する変更をサポートするように更新されました。db.m4 スタンダード DB インスタンスクラスをサポートするように更新されました。	2015 年 11 月 11 日
新機能	手動 DB スナップショット共有をサポートするように更新されました。詳細については、「Amazon RDS の DB スナップショットを共有する」を参照してください。	2015 年 10 月 28 日
新機能	Microsoft SQL Server 2014 の Web、Express、および Standard の各エディションをサポートするように更新されました。	2015 年 10 月 26 日
新機能	MySQL ベースの MariaDB データベースエンジンをサポートするように更新されました。詳細については、「Amazon RDS for MariaDB」を参照してください。	2015 年 10 月 7 日
新機能	アジアパシフィック (東京) リージョンで Amazon Aurora をサポートするように更新されました。アベイラビリティーゾーンの詳細については、「Amazon Aurora MySQL の使用可否」を参照してください。	2015 年 10 月 7 日
新機能	db.t2.large DB インスタンスクラスのすべての DB エンジンで、バースト可能な db.t2 DB インスタンスクラスをサポートするように更新されました。詳細については、「 DB インスタンスクラス」を参照してください。	2015 年 9 月 25 日

変更	説明	変更日
新機能	R3 および T2 DB インスタンスクラスで Oracle DB インスタンスをサポートするように更新しました。詳細については、「DB インスタンスクラス」を参照してください。	2015 年 8 月 5 日
新機能	Microsoft SQL Server Enterprise Edition がライセンス込みのサービスモデルで利用できるようになりました。詳細については、「Amazon RDS での Microsoft SQL Server のライセンス」を参照してください。	2015 年 7 月 29 日
新機能	Amazon Aurora が正式リリースされました。Amazon Aurora は DB クラスターで複数の DB インスタンスをサポートする DB エンジンです。詳細については、「Amazon Aurora とは」を参照してください。	2015 年 7 月 27 日
新機能	DB スナップショットへのタグのコピーをサポートするように更新されました。	2015 年 7 月 20 日
新機能	すべての DB エンジン用ストレージサイズの増加および SQL Server 用 Provisioned IOPS の増加をサポートするために更新されました。	2015 年 6 月 18 日
新機能	リザーブド DB インスタンスのオプションが更新されました。	2015 年 6 月 15 日
新機能	TDE を使用した Oracle DB による Amazon CloudHSM の使用をサポートするように更新されました。	2015 年 1 月 8 日
新機能	保管時のデータの暗号化と新しい API バージョン 2014-10-31 をサポートするように更新されました。	2015 年 1 月 6 日

変更	説明	変更日
新機能	新しい Amazon DB エンジンである Aurora が含まれるように更新されました。Amazon Aurora は DB クラスターで複数の DB インスタンスをサポートする DB エンジンです。現在、Amazon Aurora はプレビューリリースであり、変更される可能性があります。詳細については、「Amazon Aurora とは」を参照してください。	2014 年 11 月 12 日
新機能	PostgreSQL のリードレプリカをサポートするように更新されました。	2014 年 11 月 10 日
新しい API と機能	GP2 タイプのストレージと新しい API バージョン 2014-09-01 をサポートするように更新されました。既存のオプションやパラメータグループをコピーして新しいオプションやパラメータグループを作成する機能をサポートするように更新されました。	2014 年 10 月 7 日
新機能	MySQL バージョン 5.6.19 以降を実行する DB インスタンス用の InnoDB キャッシュウォームアップをサポートするように更新されました。	2014 年 9 月 3 日
新機能	MySQL バージョン 5.6、SQL Server、および PostgreSQL のデータベースエンジン接続時に SSL 証明書認証をサポートするように更新されました。	2014 年 8 月 5 日
新機能	バースト可能な db.t2 DB インスタンスクラスをサポートするように更新されました。	2014 年 8 月 4 日
新機能	メモリ最適化が行われた db.r3 DB インスタンスクラスをサポートするように更新され、MySQL (バージョン 5.6)、SQL サーバー、PostgreSQL データベースエンジンと併用されます。	2014 年 5 月 28 日
新機能	SQL Server ミラーリングを使用する SQL Server マルチ AZ 配置をサポートするために更新されました。	2014 年 5 月 19 日

変更	説明	変更日
新機能	MySQL バージョン 5.5 からバージョン 5.6 へのアップグレードをサポートするために更新されました。	2014 年 4 月 23 日
新機能	Oracle GoldenGate をサポートするために更新されました。	2014 年 4 月 3 日
新機能	M3 DB インスタンスクラスをサポートするために更新されました。	2014 年 2 月 20 日
新機能	Oracle のタイムゾーンオプションをサポートするために更新されました。	2014 年 1 月 13 日
新機能	異なるリージョンにある MySQL DB インスタンス間でのレプリケーションをサポートするために更新されました。	2013 年 11 月 26 日
新機能	PostgreSQL DB エンジンをサポートするために更新されました。	2013 年 11 月 14 日
新機能	SQL Server の透過的なデータ暗号化 (TDE) をサポートするために更新されました。	2013 年 11 月 7 日
新しい API と新機能	クロスリージョン DB スナップショットのコピーをサポートするために更新されました。新しい API バージョン、2013-09-09。	2013 年 10 月 31 日
新機能	Oracle Statspack をサポートするために更新されました。	2013 年 9 月 26 日
新機能	レプリケーションを使用して MySQL のインスタンス間でデータのインポートまたはエクスポートをサポートするために更新されました。このデータのインポートとエクスポートは、Amazon RDS で実行される MySQL のインスタンスと、オンプレミスまたは Amazon EC2 で実行されるインスタンスの間で行われます。	2013 年 9 月 5 日

変更	説明	変更日
新機能	MySQL 5.6 用に db.cr1.8xlarge DB インスタンスクラスをサポートするために更新されました。	2013 年 9 月 4 日
新機能	リードレプリカのレプリケーションをサポートするために更新されました。	2013 年 8 月 28 日
新機能	パラレルリードレプリカの作成をサポートするために更新されました。	2013 年 7 月 22 日
新機能	すべての Amazon RDS リソースに対する詳細に調整されたアクセス許可とタグ付けをサポートするために更新されました。	2013 年 7 月 8 日
新機能	新しいインスタンスで MySQL 5.6 をサポートするために更新されました。MySQL 5.6 の memcached インターフェイスやバイナリログアクセスなどがサポートされています。	2013 年 7 月 1 日
新機能	MySQL 5.1 から MySQL 5.5 へのメジャーバージョンアップグレードをサポートするために更新されました。	2013 年 6 月 20 日
新機能	パラメータ値で表現を使用できるように DB パラメータグループを更新しました。	2013 年 6 月 20 日
新しい API と新機能	リードレプリカのステータスをサポートするために更新されました。新しい API バージョン、2013-05-15。	2013 年 5 月 23 日
新機能	ネイティブのネットワーク暗号化に関する Oracle Advanced Security 機能と、Oracle Transparent Data Encryption をサポートするために更新されました。	2013 年 4 月 18 日
新機能	SQL Server のメジャーバージョンアップグレードと、プロビジョンド IOPS の追加機能をサポートするために更新されました。	2013 年 3 月 13 日

変更	説明	変更日
新機能	RDS で VPC をデフォルトでサポートするために更新されました。	2013 年 3 月 11 日
新しい API と新機能	ログアクセスをサポートするために更新されました。新しい API バージョン、2013-02-12。	2013 年 3 月 4 日
新機能	RDS イベント通知サブスクリプションをサポートするために更新されました。	2013 年 2 月 4 日
新しい API と新機能	DB インスタンスの名前変更のサポート、および VPC の DB セキュリティグループのメンバーを VPC セキュリティグループへ移行することをサポートするために更新されました。	2013 年 1 月 14 日
新機能	AWS GovCloud (US-West) のサポートが更新されました。	2012 年 12 月 17 日
新機能	m1.medium DB インスタンスクラスと m1.xlarge DB インスタンスクラスをサポートするために更新されました。	2012 年 11 月 6 日
新機能	リードレプリカの昇格をサポートするために更新されました。	2012 年 10 月 11 日
新機能	Microsoft SQL Server DB インスタンスの SSL をサポートするために更新されました。	2012 年 10 月 10 日
新機能	Oracle マイクロ DB インスタンスをサポートするために更新されました。	2012 年 9 月 27 日
新機能	SQL Server 2012 をサポートするために更新されました。	2012 年 9 月 26 日
新しい API と新機能	プロビジョンド IOPS をサポートするために更新されました。API バージョン: 2012-09-17。	2012 年 9 月 25 日

変更	説明	変更日
新機能	VPC の DB インスタンスでの SQL Server サポート、および Data Pump での Oracle サポートに対応するために更新されました。	2012 年 9 月 13 日
新機能	SQL Server エージェントをサポートするために更新されました。	2012 年 8 月 22 日
新機能	DB インスタンスのタグ付けをサポートするために更新されました。	2012 年 8 月 21 日
新機能	Oracle APEX、XML DB、Oracle のタイムゾーン、および VPC での Oracle DB インスタンスをサポートするために更新されました。	2012 年 8 月 16 日
新機能	SQL Server データベースエンジンチューニングアドバイザーおよび VPC での Oracle DB インスタンスをサポートするために更新されました。	2012 年 7 月 18 日
新機能	オプショングループをサポートするために更新されました。初期のオプションは Oracle Enterprise Manager Database Control です。	2012 年 5 月 29 日
新機能	Amazon Virtual Private Cloud において、リードレプリカをサポートするように更新されました。	2012 年 5 月 17 日
新機能	Microsoft SQL Server をサポートするために更新されました。	2012 年 5 月 8 日
新機能	強制フェイルオーバー、Oracle DB インスタンスのマルチ AZ 配置、および Oracle DB インスタンスのデフォルト以外の文字セットをサポートするために更新されました。	2012 年 5 月 2 日
新機能	Amazon Virtual Private Cloud (VPC) サポートを更新しました。	2012 年 2 月 13 日

変更	説明	変更日
更新された内容	新しいリザーブドインスタンスタイプに対応するために更新されました。	2011 年 12 月 19 日
新機能	Oracle エンジンをサポートするために更新されました。	2011 年 5 月 23 日
更新された内容	コンソールが更新されました。	2011 年 5 月 13 日
更新された内容	短縮されたバックアップとメンテナンスの時間に関する内容を編集しました。	2011 年 2 月 28 日
新機能	MySQL 5.5 のサポートを追加しました。	2011 年 1 月 31 日
新機能	リードレプリカのサポートを追加しました。	2010 年 10 月 4 日
新機能	AWS Identity and Access Management (IAM) のサポートが追加されました。	2010 年 9 月 2 日
新機能	DB エンジンのバージョン管理を追加しました。	2010 年 8 月 16 日
新機能	リザーブド DB インスタンスを追加しました。	2010 年 8 月 16 日
新機能	Amazon RDS では、DB インスタンスへの SSL 接続が可能になりました。	2010 年 6 月 28 日
新規ガイド	これは Amazon RDS ユーザーガイド の初期のリリースです。	2010 年 6 月 7 日

AWS 用語集

最新の AWS の用語については、「AWS の用語集 リファレンス」の「AWS 用語集」を参照してください。